People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University Echahid Cheikh Larbi Tebessi - Tebessa

Faculty of Exact Sciences and Natural and Life Sciences

Department : Computer Science

Thesis

For Obtaining the MASTER's Degree

Specialty: Computer Networks and Security (RSI)

Flood Detection in Satellite Images Using Deep Learning

Theme Submitted by:
Rabah Manar

Before the jury:

Dr. Boualleg Yaakoub	*Larbi Tebessi University*	*President*
Dr. Gahmousse Abdellatif	*Larbi Tebessi University*	*Examiner*
Dr. Nouioua Tarek	*Larbi Tebessi University*	*Supervisor*
Dr. Zeggari Ahmed	*Larbi Tebessi University*	*Co-Supervisor*

Date: 10/06/2025

Acknowledgment

First of all, I would like to thank ALLAH for giving me the strength, patience, and courage to finish this work.

I am very thankful to my supervisor, Dr. Tarek Nouioua, for his help, advice, and support during this project.

I also thank my co-supervisor, Dr. Ahmed Zaggari, for his useful guidance and support.

I would also like to sincerely thank the jury members, the President Dr. Boualleg Yaakoub and the Examiner Dr. Gahmousse Abdellatif for accepting to evaluate and discuss my thesis. I really appreciate their time and effort.

Finally, I express my gratitude to everyone who contributed, directly or indirectly, to the completion of this work. Your support and the knowledge you shared were essential to my success.

Dedication

I dedicate this work to the person I was, who faced many challenges but never gave up.

To the person I am today, who has worked hard to reach this point.

And to the person I will become, who will continue to grow, improve, and follow new dreams.

I also dedicate this thesis to my dear parents for their love, prayers, and constant support.

To my brothers and sisters, for always standing by my side.

To my colleague Belghit Abderrazak, thank you for your help, valuable advice, and continuous support during this journey.

And finally, to all my friends and my whole family, your kindness, help, and motivation meant a lot to me.

Abstract

Floods are one of the most common and dangerous natural disasters. They can damage homes, roads, and important services, and they often happen without warning. To reduce the damage, it is important to detect floods quickly and accurately. This study uses deep learning methods to detect floods in satellite images.

A total of 380 images were used, divided equally into flood and non-flood classes. Several deep learning models were tested, including VGG, ResNet, DenseNet, MobileNet, and Inception. These models were improved using transfer learning, which helps save time and improves results. The best models, such as ResNet50V2 and DenseNet, achieved high accuracy (up to 97%) in classifying the images.

The study also tested a new idea by adding a quantum layer to the models using the PennyLane library. These hybrid models showed promising results.

This research shows that deep learning and transfer learning are powerful tools for flood detection. In the future, this work can be improved by using more data and trying new ideas to make it even more accurate and useful.

Keywords: Flood Detection, Satellite Images, Deep Learning, Transfer Learning, CNN, ResNet, DenseNet, Quantum Layer, PennyLane, Image Classification.

Résumé

Les inondations sont l'une des catastrophes naturelles les plus fréquentes et les plus dangereuses. Elles peuvent endommager les maisons, les routes et les services importants, et elles arrivent souvent sans avertissement. Pour réduire les dégâts, il est important de détecter les inondations rapidement et avec précision. Cette étude utilise des méthodes d'apprentissage profond (deep learning) pour détecter les inondations dans les images satellites.

Un total de 380 images a été utilisé, réparti de manière égale entre les classes "inondation" et "pas d'inondation". Plusieurs modèles de deep learning ont été testés, comme VGG, ResNet, DenseNet, MobileNet et Inception. Ces modèles ont été améliorés grâce à l'apprentissage par transfert (transfer learning), ce qui permet de gagner du temps et d'obtenir de meilleurs résultats. Les meilleurs modèles, comme ResNet50V2 et DenseNet, ont atteint une grande précision (jusqu'à 97 %) pour classer les images.

L'étude a aussi testé une nouvelle idée en ajoutant une couche quantique aux modèles à l'aide de la bibliothèque PennyLane. Ces modèles hybrides ont donné des résultats prometteurs.

Cette recherche montre que le deep learning et le transfer learning sont des outils puissants pour détecter les inondations. À l'avenir, ce travail peut être amélioré en utilisant plus de données et de nouvelles idées pour le rendre encore plus précis et utile.

Mots-clés: Détection d'inondation, Images satellites, Apprentissage profond, Apprentissage par transfert, CNN, ResNet, DenseNet, Couche quantique, PennyLane, Classification d'images.

ملخص

تُعدّ الفيضانات من أكثر الكوارث الطبيعية شيوعًا وخطورة. فهي قد تتسبب في تدمير المنازل والطرقات والخدمات الأساسية، وغالبًا ما تحدث دون سابق إنذار. للحد من الأضرار، من المهم اكتشاف الفيضانات بسرعة وبدقة. تعتمد هذه الدراسة على تقنيات التعلم العميق لاكتشاف الفيضانات باستخدام صور الأقمار الصناعية.

تم استخدام 380 صورة في هذه الدراسة، مقسمة بالتساوي إلى فئات "فيضانات" و"بدون فيضانات". تم اختبار عدة نماذج من التعلم العميق، منها VGG وResNet وDenseNet وMobileNet وInception. وقد تم تحسين أداء هذه النماذج باستخدام تقنية "التعلم بالنقل" (Transfer Learning)، مما يساعد على توفير الوقت وتحقيق نتائج أفضل. من بين النماذج الأفضل أداءً كان ResNet50V2 وDenseNet، حيث وصلت دقتها إلى 97٪ في تصنيف الصور.

كما اختبرت الدراسة فكرة جديدة من خلال إضافة طبقة كمية (Quantum Layer) إلى النماذج باستخدام مكتبة PennyLane، وقد أظهرت هذه النماذج الهجينة نتائج واعدة.

تُبيّن هذه الدراسة أن التعلم العميق والتعلم بالنقل أدوات فعالة في اكتشاف الفيضانات. ويمكن تحسين هذا العمل مستقبلًا باستخدام بيانات أكثر وتجربة أفكار جديدة لجعل النظام أكثر دقة وفائدة.

الكلمات المفتاحية: اكتشاف الفيضانات، صور الأقمار الصناعية، التعلم العميق، التعلم بالنقل، الشبكات العصبية الالتفافية (CNN)، ResNet، DenseNet)، الطبقة الكمية، PennyLane، تصنيف الصور.

Table of content

Acknowledgment **1**
Dedication **2**
Abstract **3**
Résumé **4**
ملخص **5**
Table of content **6**
List of Figures **9**
List of Tables **11**
List of abbreviations **12**
General Introduction **12**
Chapter 1: **Theoretical Background** **1**
1. Introduction 2
2. Satellite Image Basics 2
2.1. Optical Images 2
2.2. Radar Images 2
2.3. Comparison of Optical and Radar Images 3
2.4. Image Resolution 3
Spatial Resolution 3
Temporal Resolution 4
2.5. Revisit Time 4
2.6. Basic Flood Detection Methods 4
Normalized Difference Water Index 4
Modified Normalized Difference Water Index 5
Normalized Difference Vegetation Index 5
3. Artificial Intelligence 5
4. Machine Learning 6
4.1. Definition 6
4.2. Types of Machine Learning 6
4.3. Artificial Neural Networks 9
4.4. Multilayer Perceptron 10
4.5. Activation Functions 10
5. Deep Learning 12
5.1. Convolutional Neural Networks 12
5.2. Deep Learning for Remote Sensing 13
5.3. Transfert Learning 14
6. Quantum Computing 14
6.1. Definition 15
6.2. Qubit 15
6.3. Principles of Quantum Computing 15
6.4. Quantum Circuits 16
6.5. Quantum Gates 17
7. Evaluation Metrics 18
8. Conclusion 20

Chapter 2: Review of existing Approaches for Flood Detection.....21
1. Introduction.....22
2. Previous Work.....22
3. Conclusion.....29
Chapter 3: Methodology.....30
1. Introduction.....31
2. Tools and Libraries.....31
Python.....31
Google Colaboratory.....31
Google Drive.....31
TensorFlow.....32
Keras.....32
Numpy.....32
Matplotlib.....32
Scikit-Learn.....33
PennyLane.....33
3. Data Collection.....33
4. Data Preprocessing.....34
5. Models Used.....34
6. Model Architecture.....38
7. Model Training.....40
8. Conclusion.....41
Chapter 4: Results and Analysis.....42
1. Introduction.....43
2. Results.....43
2.1 VGG.....43
VGG16.....43
VGG19.....44
2.2 ResNet.....45
ResNet50V2.....45
ResNet101V2.....46
ResNet152V2.....48
2.3 DenseNet.....49
DenseNet121.....49
Densenet169.....50
DenseNet201.....51
2.4 MobileNet.....52
MobileNet.....52
MobileNetV2.....53
2.5 Inception.....54
InceptionV3.....54
InceptionResNetV2.....55
3. Comparison between models with and without Data Augmentation.....56
4. Results of Hybrid Models.....56

5. Comparison of Results..... 59
5.1 Comparison of Classical Models with the state of the art..... 59
5.2 Comparison of Hybrid Models with the state of the art..... 59
6. Conclusion..... 60
General Conclusion.....61

List of Figures

Figure 1.1: Impact of Spatial Resolution 3
Figure 1.2: Temporal Resolution 4
Figure 1.3: Key Components of Artificial Intelligence 5
Figure 1.4: Types of Machine Learning Algorithms 6
Figure 1.5: Supervised Machine Learning schema 7
Figure 1.6: Unsupervised Machine Learning schema 7
Figure 1.7: Reinforcement Learning schema 8
Figure 1.8: Architecture of Artificial Neural Networks 9
Figure 1.9: MultiLayer Perceptron schema 10
Figure 1.10: Convolutional Neural Networks Architecture 13
Figure 1.11: Classical Bit and Quantum Qubit States in Superposition 14
Figure 1.12: Measurement of Maximally Entangled Qubit Pair Between Two Parties 15
Figure 1.13: Creating Superposition with the Hadamard Gate 15
Figure 1.14: Effect of the Pauli-X Gate on Qubit States 16
Figure 1.15: Matrix Representation of Pauli-Y and Pauli-Z Gates 16
Figure 1.16: Entanglement via the Controlled-NOT Gate 16
Figure 3.1: VGG Neural Network Architecture 34
Figure 3.2: Residual connections 35
Figure 3.3: Dense connections 36
Figure 3.4: Depthwise separable convolution block 36
Figure 3.5: Inverted residual block and linear bottleneck block 37
Figure 3.6: An example of an Inception module 38
Figure 3.7: Model Architecture 39
Figure 3.8: Hybrid Model Architecture 40
Figure 4.1: VGG16 Model results 43
Figure 4.2: VGG19 Model results 44
Figure 4.3: ResNet50V2 Model results 45
Figure 4.4: ResNet101V2 Model results 46
Figure 4.5: ResNet152V2 Model results 47
Figure 4.6: DenseNet121 Model results 48

Figure 4.7: DenseNet169 Model results 49

Figure 4.8: DnseNet201 Model results 50

Figure 4.9: MobileNet Model results 51

Figure 4.10: MobileNetV2 Model results 52

Figure 4.11: InceptionV3 Model results 53

Figure 4.12: InceptionResNetV2 Model results 54

Figure 4.13: Comparison of the models Accuracy with and without Data Augmentation 55

Figure 4.14: Comparison between the classic model and the hybrid model 57

List of Tables

Table 1.1: Comparison of Optical and Radar Images 3
Table 2.1: Summary of related work on flood detection 28
Table 4.1: Classification report of VGG16 42
Table 4.2: Classification report of VGG19 44
Table 4.3: Classification report of ResNet50V2 45
Table 4.4: Classification report of ResNet101V2 46
Table 4.5: Classification report of ResNet152V2 47
Table 4.6: Classification report of DenseNet121 48
Table 4.7: Classification report of DenseNet169 49
Table 4.8: Classification report of DenseNet201 50
Table 4.9: Classification report of MobileNet 51
Table 4.10: Classification report of MobileNetV2 52
Table 4.11: Classification report of InceptionV3 53
Table 4.12: Classification report of InceptionResNetV2 53
Table 4.13: Evaluation Results of Hybrid Deep Learning Models for Flood Detection 56
Table 4.14: Comparison of classical models with the state of the art 58
Table 4.15: Comparison of Hybrid models with the state of the art 59

List of abbreviations

AI: Artificial Intelligence

ML: Machine Learning

RL: Reinforcement Learning

ANN: Artificial Neural Network

MLP: Multilayer Perceptron

ReLU: Rectified Linear Unit

DL: Deep Learning

CNN: Convolutional Neural Network

TL: Transfer Learning

TP : True positive

TN : True negative

FP : False positive

FN : False negative

NDWI: Normalized Difference Water Index

MNDWI: Modified Normalized Difference Water Index

NDVI: Normalized Difference Vegetation Index

SWI: Standardized based Water-Level Index

VGG: Visual Geometry Group

ResNet: Residual Network

DenseNet: Densely Connected Convolutional Network

General Introduction

Floods are one of the most common and damaging natural disasters. They can happen for many reasons, like heavy rain, rivers overflowing, melting snow, or the failure of human-made systems like dams and drainage infrastructure. In cities, the risk of flooding often gets worse due to poor water management and quick, unplanned construction. This prevents proper drainage and makes flooding more likely.

Floods can cause serious damage to homes, roads, and important services like electricity and water. They can also stop transportation and force people to leave their homes. The effects are even worse in developing countries, where there are fewer resources and weaker infrastructure to handle such disasters.

To reduce these risks, there is a growing need for smart and efficient systems that can detect floods at an early stage. Early detection helps emergency services respond faster, minimizing damage and saving lives.

Satellite and aerial imagery have become essential tools for monitoring floods. These images offer wide and updated views of affected areas, especially where access is difficult or dangerous. However, traditional image analysis methods are often too slow and not suitable for real-time decision-making. They also face challenges in complex environments like urban or forest areas.

Recent research has turned to Artificial Intelligence, especially Deep Learning, to address these limitations. Convolutional Neural Networks (CNNs), a type of deep learning model, have shown great success in analyzing images automatically and accurately. However, one common limitation of CNNs is their need for large labeled datasets, which are often unavailable during flood events.

Transfer Learning offers a useful solution by letting us reuse models that were already trained on large datasets. This approach saves time, requires fewer labeled images, and can lead to better performance under real-world conditions.

This study proposes a flood detection system based on CNNs and Transfer Learning. It evaluates and compares several pre-trained models to assess their performance on the task. Additionally, the study explores the integration of a quantum layer to see if it can help the models learn better and make more accurate predictions, even with limited data.

The main goal is to develop an automatic, fast, and reliable system that can support emergency responders and decision-makers in managing flood situations more effectively.

This thesis is structured as follows:

- General Introduction: Introduces the research topic and explains why it is important.
- Chapter 1: Covers the theoretical background, including satellite imagery, deep learning techniques, quantum concepts, and evaluation metrics.
- Chapter 2: Reviews related studies in the field.
- Chapter 3: Describes the methodology, including data preparation and model development.
- Chapter 4: Discusses experimental results and model comparisons..
- General Conclusion: Summarizes the study and suggests directions for future research.

Chapter 1

Theoretical Background

1. Introduction

This chapter presents the essential concepts required to understand flood detection using satellite imagery and deep learning. It begins by introducing different types of satellite images and important terms such as spatial resolution and revisit time. Then, it covers the basics of artificial intelligence, machine learning, and deep learning. It also discusses the concept of transfer learning, provides a brief overview of quantum computing, and describes common evaluation metrics used to assess model performance.

2. Satellite Image Basics

2.1. Optical Images

Optical images are captured using sensors that detect visible light reflected from the Earth's surface. These images resemble photographs and are useful for observing features like vegetation, urban areas, and water bodies. However, capturing these images is limited by daylight and weather conditions, such as cloud cover [1].

2.2. Radar Images

Radar images are acquired using active sensors that transmit radar signals and receive the reflected responses from the Earth’s surface. These sensors can operate effectively in any weather conditions or lighting, allowing them to capture images both day and night even through clouds and fog. This capability makes radar imagery highly valuable for observing environmental changes, natural disasters, and terrain structures [2].

2.3. Comparison of Optical and Radar Images

Feature	Optical Images	Radar Images
Sensor Type	Passive (relies on sunlight)	Active (emits its own signals)
Weather Dependence	Affected by clouds and weather conditions	Works through and operates in all weather
Day/Night Capability	Limited to daylight	Operates both day and night

Table 1.1: Comparison of Optical and Radar Images.

2.4. Image Resolution

Spatial Resolution

Spatial resolution refers to the smallest object that can be detected by a sensor, typically represented by the pixel size in an image. A higher spatial resolution means the image shows more detail, making it easier to see small features on the Earth's surface [3].



Figure 1.1: Impact of Spatial Resolution [3].

Temporal Resolution

Temporal resolution refers to how often a sensor captures images of the same place High temporal resolution is crucial for monitoring dynamic phenomena such as vegetation growth, urban development, or natural disasters [4].

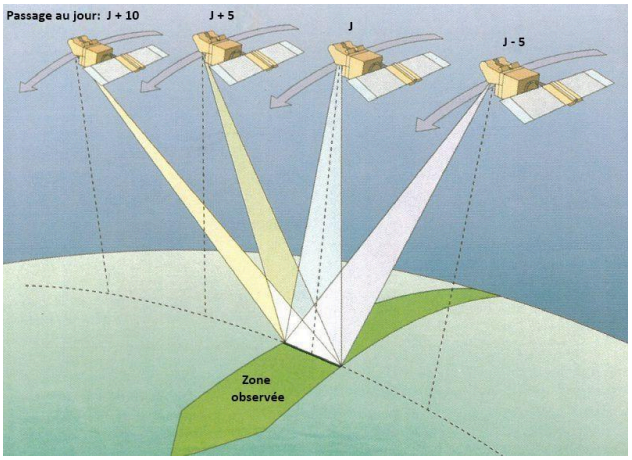


Figure 1.2: Temporal Resolution [57].

2.5. Revisit Time

Revisit time is how long it takes for a satellite to pass over the same area again. Satellites with shorter revisit times can provide more timely data, which is essential for applications requiring frequent updates [5].

2.6. Basic Flood Detection Methods

In the past, people used simple band-based methods to find water in satellite images. These methods compare two image bands to spot water. Here are the main indices:

Normalized Difference Water Index

$$NDWI = \frac{G-NIR}{G+NIR}$$

- G = green band
- NIR = near-infrared band

Water reflects green light and absorbs NIR light, so water pixels get high NDWI values [27].

Modified Normalized Difference Water Index

$$MNDWI = \frac{G-MIR}{G+MIR}$$

- G = green band
- MIR = middle infrared band

MIR shows water more clearly than NIR, so MNDWI better separates water from land and buildings [28].

Normalized Difference Vegetation Index

$$NDVI = \frac{NIR-R}{NIR+R}$$

- NIR = near-infrared band
- R = red band

NDVI was made for plants, but flooded plants lose their normal signal. Low NDVI values can point to water [29].

These three techniques were easy to use and helped find water quickly. However, because they can mix up water with other surfaces, researchers moved on to smarter methods like deep learning.

3. Artificial Intelligence

Artificial Intelligence (AI) is a field of computer science that focuses on developing systems capable of performing tasks that usually require human intelligence. These tasks include learning from experience, recognizing patterns, understanding language, and making decisions. AI combines techniques from machine learning, data analysis, and natural language processing to create smart systems that can adapt to different situations and solve problems efficiently [6].

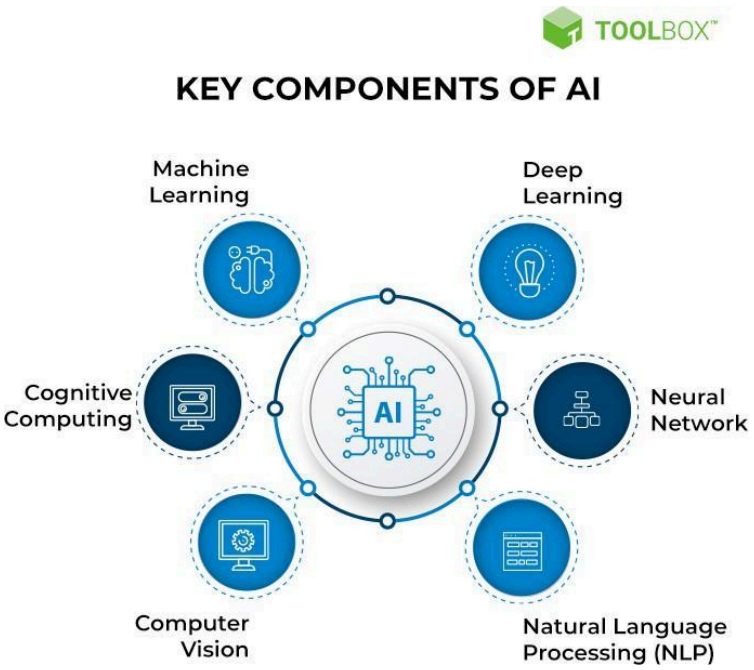


Figure 1.3: Key Components of Artificial Intelligence [7].

4. Machine Learning

4.1. Definition

Machine Learning (ML) is a subset of AI focused on developing algorithms that allow computers to learn from data and improve their performance over time, without needing explicit programming for each task. ML algorithms enable systems to recognize patterns and make predictions or decisions based on past data [8].

4.2. Types of Machine Learning

Machine learning can be categorized into three primary types, each defined by the nature of the data used for training and the problem-solving approach. These types are: **Supervised Learning**, **Unsupervised Learning**, and **Reinforcement Learning**.

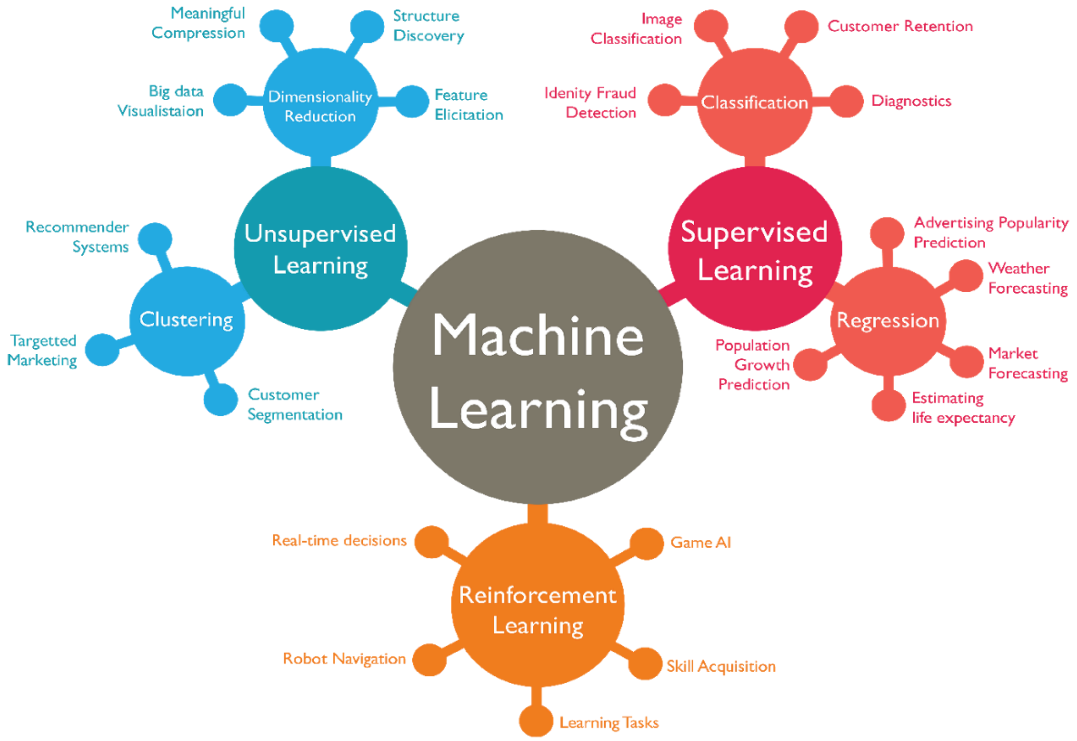


Figure 1.4: Types of Machine Learning Algorithms [9].

- **Supervised Learning**

Supervised learning is a type of machine learning where the algorithm is trained using data that already has known outcomes. In simple terms, this means that the system learns by being shown examples of the right answers (called labels). For instance, in a classification task like detecting spam emails, the system is trained with a set of emails that are already labeled as "spam" or "not spam". The system then learns how to predict whether new or unseen emails are spam or not. There are two main subtypes within supervised learning:

Classification: The goal is to assign labels to data based on input features. For example, determining if an email is spam or not.

Regression: This is used when the output is a continuous value. For example, predicting the price of a house based on features like location, size, and condition [10].



Figure 1.5: Supervised Machine Learning schema [11].

- **Unsupervised Learning**

Unsupervised learning is used when the data doesn't have labels, so the system has to find patterns or groups on its own. The algorithm tries to understand how data points are related without knowing the correct answers. For example, it can group customers based on their shopping habits (this is called clustering), or it can simplify complex data to make it easier to study (this is called dimensionality reduction).

The main tasks in unsupervised learning include:

Clustering: Grouping similar data points together. For example, organizing customers into different categories based on their behavior.

Association: Finding relationships between items in large datasets, like discovering which products are often bought together [10].

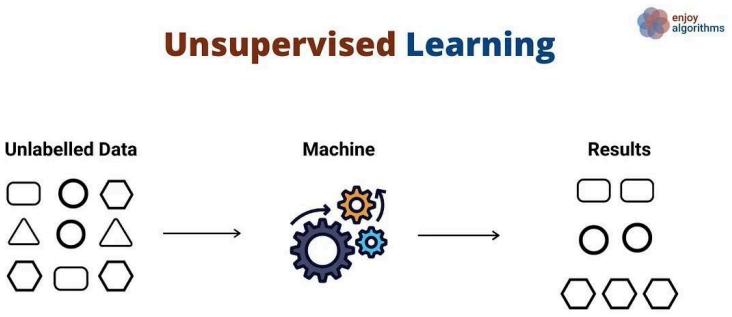


Figure 1.6: Unsupervised Machine Learning schema [11].

- **Reinforcement Learning**

Reinforcement Learning (RL) is a type of machine learning where a program, called an agent, learns by trying things out and getting feedback from its environment. The agent's goal is to earn as much reward as possible by choosing good actions and avoiding bad ones. Unlike other types of learning, it doesn't use labeled data or examples it learns through trial and error, like how people learn by doing something and seeing what happens. Over time, the agent figures out which actions lead to better results.

For example, RL is used in robotics where a robot learns to navigate a room by trial and error, and in video games where an AI learns how to win by playing the game multiple times [10].

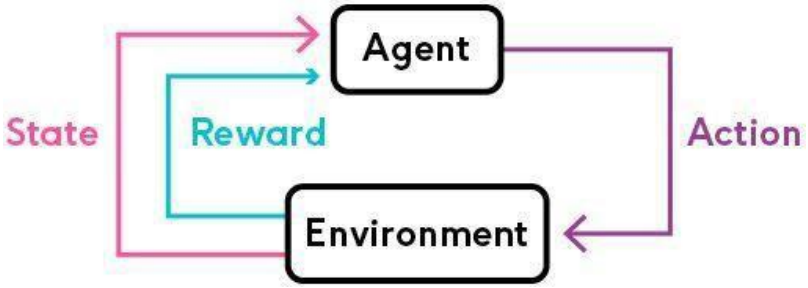


Figure 1.7: Reinforcement Learning schema [12].

4.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of the human brain. They consist of multiple layers of interconnected nodes, named neurons, which work collaboratively to process data. These networks are designed to learn patterns within data by adjusting the weights of connections between the neurons. This adaptive learning process allows the network to enhance its performance as more data is processed. A basic neural network typically includes an input layer that receives the data, one or more hidden layers where the data is processed, and an output layer that produces the final result. ANNs are widely utilized in diverse fields such as image recognition, speech processing, and medical diagnostics due to their ability to identify intricate patterns and provide accurate predictions [13].

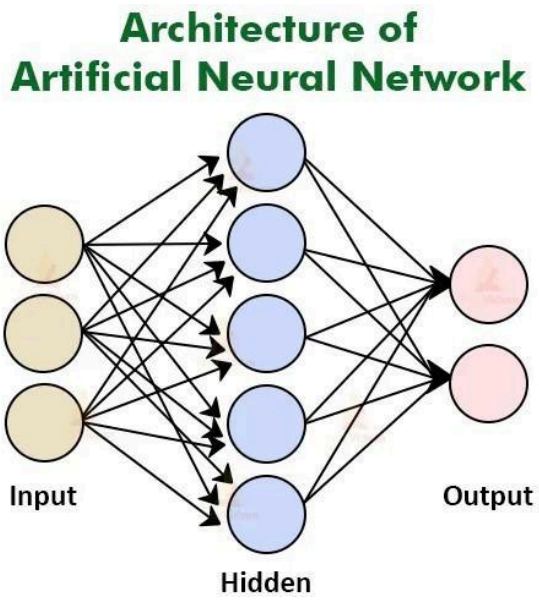


Figure 1.8: Architecture of Artificial Neural Networks [14].

4.4. Multilayer Perceptron

The Multilayer Perceptron (MLP) is one of the most common and basic types of artificial neural networks. It consists of three main parts: an input layer, one or more hidden layers, and an output layer. Each neuron in one layer is fully connected to the neurons in the next layer. The neurons receive inputs, apply weights to them, and then pass the result through an activation function such as ReLU or sigmoid to introduce non-linearity into the model.

What makes MLP powerful is its ability to learn complex relationships in data, especially non-linear patterns. This learning process is done using an algorithm called backpropagation, which calculates the error between the predicted output and the actual result, and then adjusts the weights to minimize this error. It often uses gradient descent to update the weights efficiently.

MLPs are used in many real-world tasks. Although newer deep learning models exist today, MLP remains important as it laid the foundation for more advanced networks like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [15].

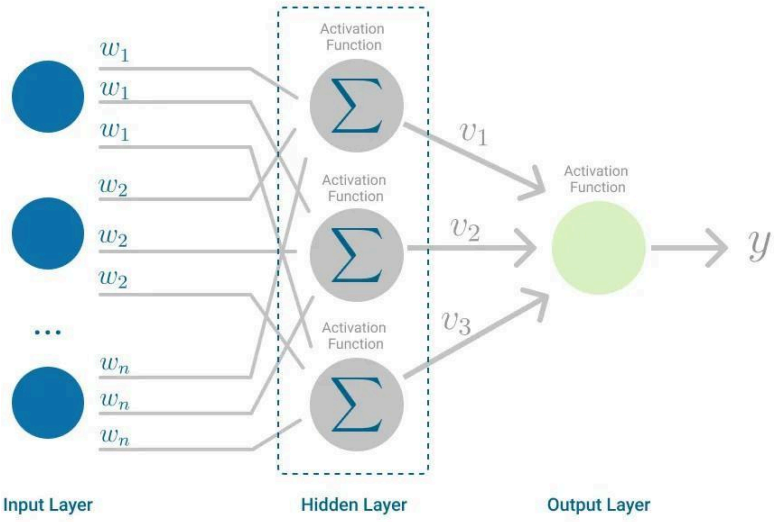


Figure 1.9: MultiLayer Perceptron schema [16].

4.5. Activation Functions

Activation functions are essential components of artificial neural networks, responsible for introducing non-linearity into the model. This non-linearity allows the network to learn complex patterns and relationships between inputs and outputs. Several activation functions have been developed over time, each with specific properties and use cases. Early functions like the binary step and linear functions are rarely used in modern deep learning due to their limited representational power. In contrast, functions such as the Rectified Linear Unit (ReLU) have become standard due to their computational simplicity and ability to mitigate the vanishing gradient problem. Similarly, the sigmoid function, which maps input values to a range between 0 and 1, remains useful in binary classification tasks, though it also suffers from gradient-related limitations in deeper networks. The choice of activation function can significantly influence the performance and training efficiency of neural models [17].

5. Deep Learning

Deep Learning (DL) is an important branch of machine learning that is based on artificial neural networks with multiple layers. The main goal of deep learning is to enable computers to learn from large amounts of data in a way that mimics how the human brain works. In these models, each layer learns to extract features from the data step by step. The early layers focus on simple details, like edges in an image, while the deeper layers can recognize complex patterns, such as faces or objects. These networks are trained using an algorithm called backpropagation, which updates the weights to reduce errors. Today, deep learning is widely used in applications such as image recognition, machine translation, and virtual assistants because of its strong ability to understand complex data accurately. Deep learning includes several popular architectures, such as Convolutional Neural Networks (CNNs), which are widely used for image and video processing, and Recurrent Neural Networks (RNNs), which are suitable for sequential data like speech and text. Each type has unique features that make it suitable for specific tasks [18].

5.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of deep learning model and artificial neural network designed mainly for working with visual data, such as images and videos. This type of network is inspired by the way the human brain processes visual information. Instead of analyzing the whole image at once, CNNs look at small parts of the image, which helps them understand the content more effectively.

Unlike traditional neural networks, where every neuron is connected to all neurons in the previous layer, CNNs use a smarter method called **convolution**. This technique allows the network to focus on specific regions of the image, reducing the number of calculations needed. As a result, CNNs are faster and easier to train.

CNNs consist of several key components that work together to analyze and transform input data into useful outputs. These components include:

Convolutional Layers: These are the core layers of CNNs. In each convolutional layer, the network uses small filters, to scan across the image.

After applying the filter, the network produces a new image called a feature map,

which highlights where the detected feature appears in the original image.

Pooling Layers: After the convolutional layer, CNNs use pooling layers to simplify the data.

There are two common types of pooling:

- **Max Pooling**: selects the highest value from each small region.
- **Average Pooling**: calculates the average of the values in each region.

Fully Connected Layers: At the end of the CNN, the network flattens the data into a single list of values. These values are then passed to fully connected layers, where each node is linked to all previous nodes. These layers use the extracted features to make final decisions, such as:

- Is the image of a cat or a dog?
- Does this area in the image show flooding or not?

CNNs learn by training. During training, the network compares its predictions with the correct answers and calculates the error. It then uses a method called backpropagation to adjust its internal weights to reduce the error. This process is repeated many times until the network becomes accurate at recognizing patterns in new images [19].

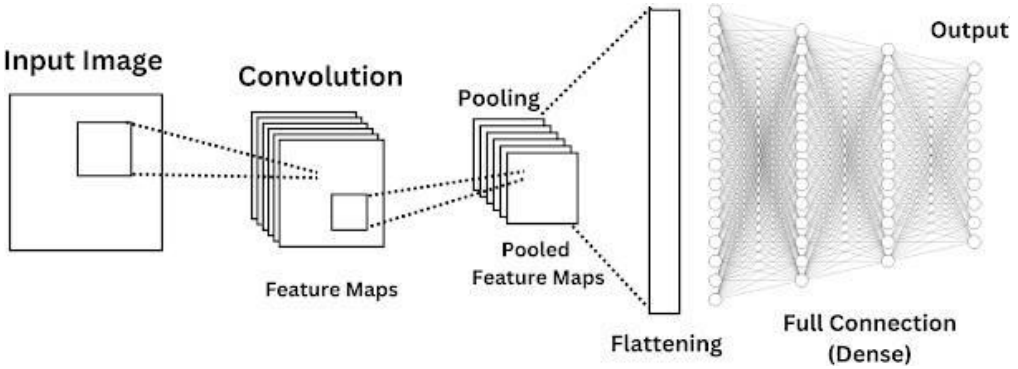


Figure 1.10: Convolutional Neural Networks Architecture [20].

5.2. Deep Learning for Remote Sensing

Deep learning has become central to remote sensing in recent years because it can learn complex patterns in satellite images without manual rules. Convolutional Neural Networks (CNNs) are the most common models: they use

layers of convolution and pooling to extract both spectral and spatial features from input images [30]. With the public release of Sentinel-1 (SAR) and Sentinel-2 (multispectral) data, researchers can train CNNs on large, high-resolution datasets [31][32]. These CNN-based methods outperform traditional spectral indices by automatically adapting to shadows, clouds, and mixed land covers. Therefore, deep learning now leads in flood detection, land-use mapping, and vegetation analysis. However, challenges such as cloud interference, seasonal variability, and model generalization across different regions still motivate ongoing research into improved architectures and data-augmentation techniques [30].

5.3. Transfert Learning

Transfer Learning (TL) is an important technique in deep learning where a model that has already learned from one task is reused for a different task. Instead of training a model from scratch, a pre-trained model usually trained on large datasets such as ImageNet is reused to solve a new task. This approach reduces the need for large amounts of data and computational resources, which is especially useful when data is limited [21].

There are two main strategies in transfer learning: feature extraction and fine-tuning. In feature extraction, the early layers of the pre-trained model, which learn general features, are frozen, and only the final layers are trained for the new task. Fine-tuning involves unfreezing some or all layers of the pre-trained model to adapt more closely to the specific task, but this requires more computation and data [22].

Transfer learning offers many advantages, including shorter training times, better performance with limited data, and reduced computational cost. However, challenges such as negative transfer when knowledge from the original task does not help or even harms performance on the new task must be considered. Choosing an appropriate pre-trained model and deciding which layers to train or freeze are crucial steps for success [23].

6. Quantum Computing

6.1. Definition

Quantum computing is a field at the intersection of computer science and quantum physics, focused on designing computational systems based on the principles of quantum mechanics. Unlike classical computing, which represents information using bits (0 or 1), quantum computing uses quantum bits, or **qubits**, as the fundamental unit of data. Due to quantum phenomena like **superposition** and **entanglement**, qubits can represent multiple states simultaneously and influence each other instantaneously. This enables quantum computers to perform certain complex computations much more efficiently than classical machines [24].

6.2. Qubit

A **qubit** (quantum bit) is the basic unit of quantum information. Unlike a classical bit that can only be in a definite state of either 0 or 1, a qubit can be in a **superposition** of both 0 and 1. Mathematically, this is expressed as:

$$|\psi\rangle=\alpha_0|0\rangle+\alpha_1|1\rangle$$

where α_0 and α_1 are complex probability amplitudes, and the squares of their magnitudes ($|\alpha_0|^2$ and $|\alpha_1|^2$) represent the probabilities of the qubit being measured as 0 or 1, respectively. In addition to superposition, qubits can also exhibit entanglement, a property where the state of one qubit is dependent on the state of another, even across large distances. This correlation allows quantum systems to process and share information in fundamentally new ways [24].

6.3. Principles of Quantum Computing

Quantum computing relies on three core principles derived from quantum mechanics:

- **Superposition**: A qubit can exist in a combination of the basis states $|0\rangle$ and $|1\rangle$ at the same time. This allows quantum processors to handle multiple potential outcomes simultaneously, vastly increasing parallelism.

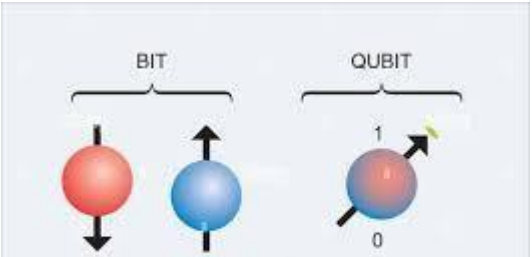


Figure 1.11: Classical Bit and Quantum Qubit States in Superposition [58].

- **Entanglement**: When qubits are entangled, the state of one qubit instantly affects the state of another, regardless of the distance between them. This phenomenon enables highly coordinated operations across many qubits.

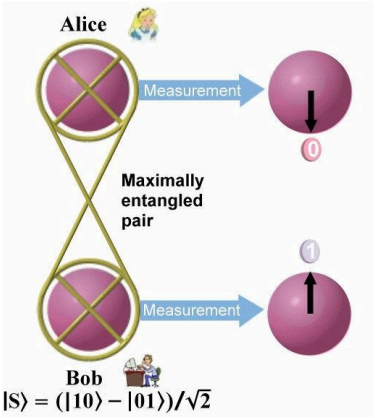


Figure 1.12: Measurement of Maximally Entangled Qubit Pair Between Two Parties [59].

These principles allow quantum computers to solve certain problems much faster than classical computers [24].

6.4. Quantum Circuits

Quantum circuits are a model used in quantum computing to represent and run quantum algorithms in practice. These circuits are made up of a series of quantum gates that are applied to qubits to perform calculations. The circuits are usually drawn as horizontal lines, each representing a qubit, and the gates are placed along these lines to show the order of operations.

Each quantum gate performs a reversible mathematical operation called a unitary operation. These operations are represented by matrices that change the state of one or more qubits. For example, when a Hadamard gate is applied to a qubit in the state $|0\rangle$, it changes the qubit into a superposition of $|0\rangle$ and $|1\rangle$.

Quantum circuits are the foundation for building and running quantum algorithms.

Understanding how to construct and use these circuits is essential for developing practical applications in quantum computing [24].

6.5. Quantum Gates

Quantum logic gates are the basic building blocks of quantum circuits, similar to how classical logic gates are used in traditional computing. These gates manipulate qubits using quantum operations that are reversible and represented by unitary matrices. Unlike classical gates, which operate on bits (0 or 1), quantum gates operate on qubits that can exist in superposition and entanglement.

Some of the most important quantum gates include:

- **Hadamard Gate (H):** This gate puts a qubit into superposition. When applied to a state $|0\rangle$ or $|1\rangle$, it creates an equal probability of being in $|0\rangle$ and $|1\rangle$.

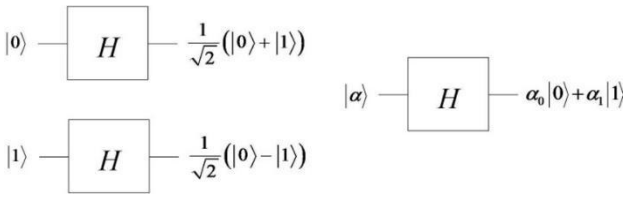


Figure 1.13: Creating Superposition with the Hadamard Gate [60].

- **Pauli-X Gate:** Similar to a classical NOT gate, it flips the state of a qubit (i.e., turns $|0\rangle$ into $|1\rangle$ and vice versa).

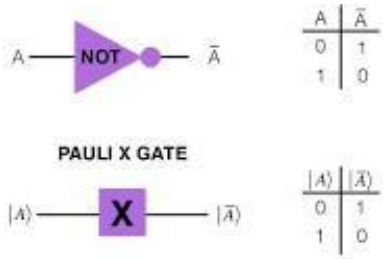


Figure 1.14: Effect of the Pauli-X Gate on Qubit States [61].

- **Pauli-Y and Pauli-Z Gates:** These gates perform more complex rotations on the Bloch sphere, useful in various quantum algorithms.

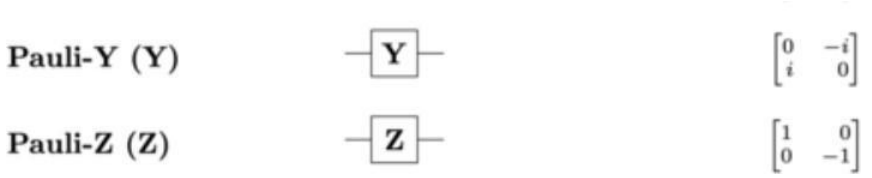


Figure 1.15: Matrix Representation of Pauli-Y and Pauli-Z Gates [62].

- **CNOT Gate (Controlled-NOT):** A two-qubit gate that flips the second qubit (target) only if the first qubit (control) is $|1\rangle$. It's essential for creating entanglement between qubits.



Control	Target	Result
$\|0\rangle$	$\|0\rangle$	$\|00\rangle$
$\|0\rangle$	$\|1\rangle$	$\|01\rangle$
$\|1\rangle$	$\|0\rangle$	$\|11\rangle$
$\|1\rangle$	$\|1\rangle$	$\|10\rangle$

Figure 1.16: Entanglement via the Controlled-NOT Gate [63].

Quantum circuits are built by applying sequences of these gates to a set of qubits. For example, to create a Bell state, a Hadamard gate is applied to one qubit, followed by a CNOT gate using that qubit as the control and another qubit as the target. These circuits form the foundation for running quantum algorithms on quantum computers [24].

7. Evaluation Metrics

Evaluation metrics are tools used to check how good a classification model is at making predictions. There are several common metrics used to measure the performance of a model.

Accuracy: is the ratio of correct predictions (both true positives and true negatives) to the total number of predictions. Although it is widely used, accuracy alone may not

be reliable when the dataset is imbalanced [25].

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$$

TP : True positive.

TN : True negative.

FP : False positive.

FN : False negative.

Recall: refers to the model's ability to correctly identify all actual positive cases. It is especially important in applications like disease detection, where missing positive cases can be critical [25].

$$Recall = \frac{TP}{TP+FN}$$

Precision: measures how many of the predicted positive cases are actually correct. High precision means the model makes fewer false positive errors [25].

$$Precision = \frac{TP}{TP+FP}$$

F1-score: is a harmonic mean of precision and recall. It is useful when we need to balance both precision and recall, especially when the data classes are not evenly distributed [25].

$$F1\ Score = 2\times\left(\frac{Precision\times Recall}{Precision+Recall}\right)$$

Intersection over Union (IoU): is commonly used in image classification and segmentation tasks. It measures the overlap between the predicted region and the actual region. A higher IoU indicates better model performance [26].

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

These metrics together provide a comprehensive view of how well a classification model works, especially in real-world situations where accuracy alone is not enough.

8. Conclusion

This chapter laid the groundwork for understanding satellite image types, image resolution, and the essential principles of deep learning, including neural networks and machine learning. The knowledge gained here is crucial for implementing and optimizing deep learning models in the context of satellite-based flood detection. With this theoretical foundation, we are now equipped to explore how these concepts are applied in real-world scenarios and how they can be leveraged to improve flood detection accuracy.

Chapter 2
Review of existing Approaches for Flood Detection

1. Introduction

In this chapter, we review several recent studies and explain what they did, how they did it, and what results they found, to help show the current progress in flood detection.

2. Previous Work

In [33], the researchers worked on detecting flood-related content by combining information from social media images and text. First, they used a tool called DeepSentiBank to extract visual features from user-uploaded images. Then, they turned user-generated hashtags into word vectors using Word2Vec and gave them weights with TF-IDF. These visual and textual features were combined and used in an SVM classifier with an RBF kernel. This method reached an Average Precision of 90.45% (AP@480) when using both images and text, compared to 86.64% with only images and 63.41% with only hashtags.

For satellite images, the team trained convolutional neural networks (CNNs), specifically FCN and SegNet models, first using only RGB channels. Later, they added infrared (IR) data to better tell apart water and land. Using IR helped improve the Intersection over Union (IoU) score by over 10% on familiar locations. The best result was 84.36% IoU using SegNet with RGB+IR, while RGB alone gave 73.56%. The model also performed well on new, unseen areas with 74.13% IoU. These results show the importance of combining visual and textual data and the strong role of infrared channels in improving flood detection accuracy. This approach helps support early warning systems and emergency response planning.

In [34], the researchers used data from 16 flood events in the Yangtze River Basin to build a dataset of 5,296 SAR image tiles with strong labels, plus two separate test sets. They trained four Convolutional Neural Networks (CNNs): FCN-8, SegNet, UNet, and DeepResUNet, and compared their results to a simple global threshold method.

Using only the VH polarization band from Sentinel-1 SAR, the top CNNs (UNet and DeepResUNet) achieved an Overall Accuracy (OA) of 98.6% and an F1-score of 97.6% on the first test set. The threshold method scored 95.8% OA and 87.7% F1. DeepResU-Net maintained 97.9% OA and 94.8% F1 on the second test set. Incorporating Digital Elevation Model (DEM) data added only a small gain.

Finally, the team applied a near-real-time flood mapping approach using a weakly labeled dataset for eight new events. Even with weak labels, UNet still reached 95.8% OA and 92.0% F1. The Intersection over Union (IoU) metric also confirmed high accuracy in segmentation. These findings show that CNNs with VH polarization SAR data can detect floods quickly and accurately, making them valuable for early warning and emergency response systems.

In [35], the team introduced a hybrid deep learning model (DHMFP–CHHSSO) to detect floods using satellite images. This model combines Convolutional Neural Networks (CNNs) and Deep ResNet, with weight optimization done by the CHHSSO algorithm. The process starts with removing noise from images using a median filter, followed by segmenting them with a chaotic-map weighted k-means method. After segmentation, the model extracts vegetation-based features like NDVI and SAVI.

The researchers trained the model using different splits of data (60%, 70%, 80%, and 90%), set a batch size of 32, and ran training for 50 epochs. The optimized model achieved 94.98% accuracy, 93.48% sensitivity, and 98.29% specificity. In an ablation study, adding CHHSSO optimization increased precision to 93.65% and lowered the false positive rate to 3.6%. For segmentation quality, the improved k-means gave a Dice score of 0.863 and a Jaccard index of 0.889.

These findings show that the DHMFP–CHHSSO model can quickly and accurately identify flooded areas, making it a promising tool for early flood warning and disaster response.

In the study by Tarek Nouioua [36], Convolutional Neural Networks (CNNs) combined with Transfer Learning were used to detect floods in satellite images. The dataset included 102 high-resolution images (82 for training and 20 for testing). Pretrained models were tested: VGG16, InceptionV3, and ResNet50.

The ResNet50 model showed strong performance on the test data, achieving an overall accuracy of 75%. For the flood class, the model reached a precision of 0.75, a recall of 0.43, and an F1-score of 0.55. For the non-flood class, the F1-score was 0.83. The author applied image preprocessing, data augmentation, and froze early layers to retrain only the top layers. Despite the limited data, these techniques helped the model

adapt well to the task. This study highlights the effectiveness of transfer learning in flood detection tasks using CNNs.

In [37], Lemenkova introduces a seamless, GIS-driven deep learning pipeline for tracking flood shifts over time in Bangladesh's Ganges Delta. She works with Landsat 8/9 OLI-TIRS scenes taken at flood peak (March) and post-flood (November) moments using 2014 images to train the model and 2021–2023 images to test it.

Instead of crafting rigid flood-detection rules, the system learns from the satellites themselves: It groups pixels into "flood" and "non-flood" clusters using k-means and maximum likelihood, then feeds those examples into a multilayer perceptron embedded directly within GRASS GIS via Scikit-Learn. Once trained, the network instantly tags every pixel in new images as underwater or dry land.

These pixel-level predictions are then woven into seasonal flood maps that don't just outline water bodies but also pinpoint wetlands, mangrove forests, croplands, towns, and aquaculture sites. The workflow shows which areas get submerged when the river swells by comparing maps before and after floods.

By baking deep learning right into an open-source GIS, Lemenkova offers a scalable, budget-friendly, and transparent tool for near-real-time flood monitoring in data-poor, high-risk deltas and a template that can be dropped into other flood-prone regions worldwide.

In [38], Elarbi built and tested two deep-learning models, classic CNNs and hybrid quantum–classical networks, to spot floods from satellite pictures. They started with 102 high-resolution images (80 for training, 20 for validation) and used simple tricks like random flips, rotations, and normalization to make the models tougher. Then, they applied transfer learning to five Pretrained models(ResNet50, VGG16, MobileNetV2, DenseNet201, InceptionV3) plus a small custom CNN. The custom CNN hit 96.7% accuracy; MobileNetV2 and VGG16 reached about 95%.

Next, they experimented with "quantum layers" using PennyLane, hoping quantum effects would boost performance. The standout was the hybrid DenseNet201, which climbed to 98% accuracy better than its purely classical version while other hybrids gave

mixed results.

They judged each model by its precision, recall, and F1-score, along with how training and validation losses behaved. The hybrid DenseNet201 struck the best balance, detecting nearly every flood while keeping false alarms low. This result shows that blending deep learning, transfer learning, and early quantum layers can markedly improve satellite-based flood detection, though these hybrid models still require further tuning and real-world testing before they're ready for deployment.

Wedajo et al. [39] show that combining multiple geospatial inputs with modern machine learning methods significantly improves flood prediction and mapping in Ethiopia's Awash Basin. They first identified the most critical factors elevation, rainfall, wetness index, aspect, and slope and then compared four models: Random Forest, Linear Regression, Support Vector Machine (SVM), and LSTM. Random Forest performed best, reaching 91% accuracy, 0.90 precision, 0.94 recall, and 0.94 AUC. Linear regression followed closely,

while SVM fared poorly and LSTM achieved moderate results. The study highlights that selecting the right features and model for each region is essential, since outcomes depend on data quality and parameter choices. These findings provide a strong foundation for building operational flood monitoring and early-warning systems in Amibara and similar flood-prone areas.

In [40], Mohamadiazar and colleagues designed a deep learning-based model using the U-Net architecture to predict urban flooding. The model integrated Sentinel-1 satellite imagery with various geospatial and environmental data, including rainfall, elevation (DEM), hydrologic soil group (HSG), and surface imperviousness. Miami-Dade County, Florida, was used as the test site, with flood event records from 2014 to 2023 used for validation.

The study evaluated the model's performance using different combinations of input data. The best results were achieved using the combination of Slope + HSG + Imperviousness + Rainfall, which gave a precision of 94.07%, a recall of 86.00%, an F1-score of 89.85%, and an overall accuracy of 93.73%. The weakest performance was seen when using only rainfall data, with an F1-score of 86.52% and accuracy of 91.57%.

This shows that combining topographic and land use information significantly improves flood prediction performance.

Overall, the model was efficient and adaptable, suitable for near real-time (NRT) flood mapping applications, although some uncertainty remains due to the temporal mismatch between rainfall and satellite observations. The results suggest that such a system could support urban flood risk management and emergency response planning.

In [41], More et al. present an end-to-end deep-learning framework for automatically mapping flood extent and estimating flood volume from Sentinel-1 and Sentinel-2 imagery over the 2019 Kolhapur event in India. They first preprocess multispectral and radar data (including indices such as Normalized Difference based Water Index (NDWI), Modified based Normalized Difference Water Index (MNDWI), and Standardized based Water-Level Index(SWI)) and manually label flood and non-flood pixels. Three advanced segmentation networks Nested U-Net (U-Net++), MA-Nets, and DeepLabV3+ are then trained on this data. On the Sen1Floods11 test set, MA-Nets achieved the highest accuracy with an F1-score of 0.94 and IoU of 0.853, while U-Net++ reached 0.919/0.849 and DeepLabV3+ 0.893/0.805. When applied to real Kolhapur images, the models attained an average F1-score of 0.85 (rising from 0.708 before the flood to 1.01 during peak inundation). Finally, the segmented flood masks were combined with a digital elevation model to calculate a flood volume of 0.0010 km^3. This work demonstrates that modern deep-learning architectures can deliver high-accuracy flood maps and reliable volume estimates using Sentinel data, offering a scalable tool for rapid flood assessment.

In [42], Doan and Le-Thi introduce a Siamese deep-learning model using a Swin-Transformer backbone to find floods in pairs of Sentinel-1 SAR images (before and after a flood). The system learns features from both images, calculates differences at multiple levels, and then uses a pyramid decoder with residual connections to segment flood areas accurately.

They trained this model on the S1GFloods dataset (4,830 image pairs, 90% training, 10% testing). It reached 96.9% precision, 94.6% recall, and a 95.7% F1-score while needing only 32.3 GFLOPs of computation. This outperforms other methods like FC-Siam-Diff, DTCDSCN, Siam-NestedUNet, and BIT. The study shows that combining a Siamese network with hierarchical vision transformers offers both high accuracy and

efficiency, making it ideal for wide-scale, near-real-time flood detection.

In [43], Manaf et al. built two aerial datasets one from benchmark sources (FloodNet, AIDER, Kaggle) and one that adds Google-downloaded images and used data augmentation (rotations, flips, mixed resolutions) to expand both. They fine-tuned five CNN backbones (Inception V3, VGG-16, ResNet50, Xception, MobileNet) on ImageNet weights for a binary flood vs. non-flood task. MobileNet and Xception led the pack with about 98% accuracy and a 92% F1-score on the main dataset. When trained on the mixed dataset, the best model stayed above 85% accuracy across all test splits, showing strong generalization to different regions and resolutions. Using LIME heatmaps, they confirmed the networks focus on water-like textures and identified common mistakes (like occluded water or odd angles), highlighting how explainable AI can guide model improvement.

Approach Used	**Year**	**Data Type**	**Geographical Area**	**Model**	**Best Performance**
Deep neural networks combining visual and textual features[33]	2017	Social media images and text	Not specified	SVM with RBF kernel	90.45% Average Precision (AP@480) combining images and text
		satellite images		FCN and SegNet for satellite images	84.36% IoU with SegNet (RGB+IR)
Deep learning with CNNs[34]	2023	Sentinel-1 SAR images	Yangtze River Basin, China	FCN-8, SegNet, UNet, DeepResUNet	98.6% OA, 97.6% F1-score (UNet, DeepResUNet)
Hybrid deep learning with CHHSSO optimization[35]	2023	Satellite images	Kerala, India	CNNs and Deep ResNet	94.98% accuracy

Transfer learning with CNNs[36]	2023	satellite images	Not specified	VGG16, InceptionV3, ResNet50	75% accuracy (ResNet50)
Baking deep learning into GIS[37]	2024	Landsat 8/9 OLI-TIRS satellite images	Ganges Delta, Bangladesh	Multilayer perceptron in GRASS GIS	Not specified
Deep learning with transfer learning and quantum layers[38]	2024	satellite images	Not specified	Custom CNN, ResNet50, VGG16, MobileNetV2, InceptionV3, DenseNet201	98% accuracy (ClassicalQuantic DenseNet201)
Machine learning[39]	2024	Satellite images and geospatial data	Awash Basin, Ethiopia	Random Forest, Linear Regression, SVM, LSTM	91% accuracy, 0.90 precision, 0.94 recall (Random Forest)
Deep learning integrating spatial-temporal analysis[40]	2024	Sentinel-1 images and environmental data	Miami-Dade County, Florida, USA	U-Net	93.73% accuracy, 94.07% precision, 89.85% F1-score
Deep learning[41]	2024	Sentinel-1 and Sentinel-2 images	Kolhapur, India	U-Net++, MA-Nets, DeepLabV3+	F1-score of 0.94, IoU of 0.853 (MA-Nets)
Siamese deep learning with Swin-Transformer[42]	2025	Sentinel-1 SAR image pairs	Not specified	Siamese network with Swin-Transformer backbone	95.7% F1-score, 96.9% precision, 94.6% recall
Deep learning with explainable AI[43]	2025	Aerial images	Multiple regions	Inception V3, VGG-16, ResNet50, Xception,	~98% accuracy, 92% F1-score (MobileNet, Xception)

				MobileNet	

Table 3.1: Summary of related work on flood detection.

3. Conclusion

In this chapter, we looked at different techniques used to detect floods in satellite images. Deep learning techniques, especially Convolutional Neural Networks (CNNs), can detect floods more quickly and accurately. CNNs can learn patterns from satellite images without needing manual steps. Using data from satellites like Sentinel-1 and Sentinel-2 helps improve the results. Some researchers also used hybrid techniques and transfer learning to make the models better. Even with these new techniques, there are still problems like not enough labeled data, and the models may not work well in every flood situation. In the future, better techniques should be developed to solve these problems and make flood detection faster and more reliable.

Chapter 3

Methodology

1. Introduction

This chapter explains the steps followed to build and test the deep learning model used for flood detection. It describes the tools and libraries used, how the data was collected and prepared, and how different models were selected and trained. Each part of the process is described clearly to show how the final model was created.

2. Tools and Libraries

This study used several Python-based tools and libraries to implement and evaluate the flood detection model. These tools were critical in data preprocessing, model building, training, testing, and visualization.

Python

In this project, we used **Python**, which is a popular and easy-to-use programming language. It was created by Guido van Rossum in the late 1980s. Python is known for its simple and readable code, which looks similar to English. This makes it a good choice for both beginners and experts. It supports object-oriented programming, which helps in building and managing complex systems. Python also has many useful libraries and tools that can be used for data analysis, machine learning, and scientific computing. Because of these features, Python was the main language used in this research [44].

Google Colaboratory

We utilized Google Colaboratory (Colab) as the main development platform. Google Colab is a free cloud-based platform provided by Google that allows users to write and run Python code through a web browser. It offers access to powerful computing resources like GPUs and TPUs without needing to install anything on a local machine. This makes it very useful for training deep learning models, especially when working with large datasets. Colab also supports easy sharing and collaboration, making it a practical tool for research and development [45].

Google Drive

In this project, Google Drive was used to store and manage the dataset and notebook files. Google Drive is a free cloud storage service that allows users to upload, organize, and access their files from anywhere using an internet connection. It integrates seamlessly with Google Colab, making it easy to load and save data during code execution. This cloud-based system also supports file sharing and collaboration, which is helpful in research environments where multiple users work on the same project [46].

TensorFlow

We employed TensorFlow, an open-source library developed by Google, designed for large-scale machine learning tasks. TensorFlow is widely used for building and training deep learning models, particularly neural networks. It offers a flexible architecture that allows for deployment across various platforms, including desktops, servers, and mobile devices. Its comprehensive ecosystem supports a range of functionalities essential for machine learning applications [44].

Keras

Keras is a simple and easy-to-use deep learning library written in Python. It helps developers build and train deep learning models quickly. It has a clean and user-friendly interface, which makes it a good choice for beginners. Keras works with powerful backends like TensorFlow and makes it easier to create complex neural networks. Because of its simplicity and flexibility, Keras was used in this research to develop and test our deep learning model [47].

Numpy

NumPy (Numerical Python) is a core library for numerical and scientific computing in Python. It supports efficient operations on large, multi-dimensional arrays and matrices, along with a wide range of high-performance mathematical functions. NumPy is widely used in data analysis, linear algebra, and statistical modeling. Its ability to perform vectorized operations significantly enhances computational speed and memory efficiency [44].

Matplotlib

Matplotlib, a widely used Python library for creating static, animated, and interactive visualizations. Matplotlib provides a comprehensive API for embedding plots into applications and is particularly effective for generating high-quality graphs and charts. It supports various plot types, including line, bar, scatter, and histogram, making it a versatile tool for data visualization in scientific and analytical applications [44].

Scikit-Learn

Scikit-learn is a popular Python library that provides many tools for machine learning, including evaluation metrics. Scikit-learn offers various functions to measure how well models perform, such as accuracy, precision, recall, F1-score, and more. These metrics help assess and compare different models to select the best one [44].

These tools worked together to provide an effective and efficient framework for flood detection through deep learning methods.

PennyLane

PennyLane is an open-source Python library made by Xanadu that helps connect quantum computers with regular machine learning tools like TensorFlow and PyTorch. It lets users build and train models that use both quantum and classical parts. The main feature of PennyLane is the QNode, which makes it easy to run and improve quantum circuits as part of a bigger machine learning system [64].

3. Data Collection

In this study, 380 optical images were collected, which contain real-world images captured during various flood events. The dataset was prepared to include an equal number of flood and non-flood images. The final dataset was divided into two subsets: 80% of the data (304 images) were used for training the model, consisting of 152 flood images and 152 non-flood images. The remaining 20% (76 images) were used for validation, including 38 flood and 38 non-flood images. This balanced distribution ensures fair training and evaluation of the deep learning model.

4. Data Preprocessing

Before training the model, several preprocessing steps were applied to prepare the image dataset. These included image cleaning by removing corrupted images and eliminating duplicate entries to ensure data quality, resizing all images to 224 × 224 pixels, and normalizing pixel values by rescaling them to a range of 0 to 1. Additionally, data augmentation techniques were employed to increase the diversity of the training data and reduce overfitting. This included applying random transformations such as rotation, width and height shifts, shear, zoom, and horizontal flips using the ImageDataGenerator class in Keras.

5. Models Used

In this study, several well-known convolutional neural network (CNN) models were evaluated to choose the best one for this task. The tested models included different versions from the following families:

- **VGG**: The Visual Geometry Group network, introduced by Simonyan and Zisserman in 2014, is a deep convolutional neural network known for its simplicity and uniform architecture. The main idea behind VGG is the use of small 3×3 convolution filters applied sequentially, which allows the network to have deeper architectures while maintaining manageable complexity. VGG models are structured in blocks, each containing multiple convolutional layers followed by max pooling, and ending with fully connected layers.

  Despite their simplicity, VGG networks are powerful in extracting hierarchical features from images. They have been widely adopted in many computer vision tasks due to their strong performance and ease of implementation. However, they are computationally expensive in terms of memory and processing time, especially with deeper versions [48].

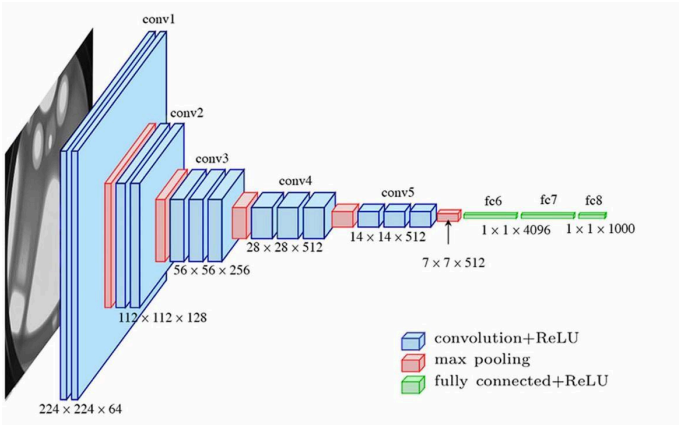


Figure 3.1: VGG Neural Network Architecture [49].

In this study, we tested two versions of the VGG family: VGG16 and VGG19, to assess their performance in classifying flood and non-flood images.

- **ResNet**: The Residual Network, introduced by He et al. (2015), brought a major advancement in deep learning by effectively training very deep neural networks. Traditional deep networks often face the **vanishing gradient problem**, where the gradient becomes too small during backpropagation, making it hard for the model to learn. ResNet solves this issue using **residual connections** (also called **skip connections**), which allow the network to learn the difference (residual) between the input and output of a layer. Instead of learning a full transformation H(x), the model learns F(x) = H(x) − x, and then adds the input back to get the final output H(x) = F(x) + x. This simple change helps the model to train deeper architectures, such as ResNet-50, ResNet-101, and ResNet-152, without performance degradation [50].

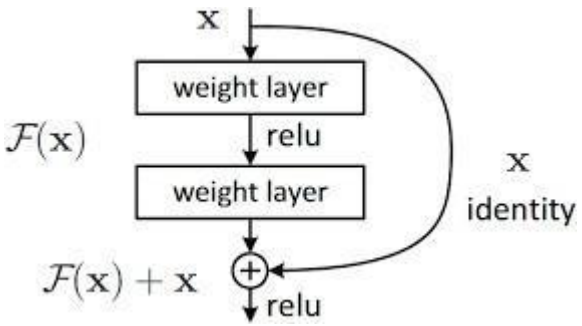


Figure 3.2: Residual connections [50].

In our work, we tested multiple versions of ResNet, including ResNet-50, ResNet-101, ResNet-101V2, ResNet-152, and ResNet-50V2. These models are widely used in image classification tasks and have achieved excellent results due to their ability to maintain accuracy even as network depth increases.

- **DenseNet**: The Densely Connected Convolutional Network, proposed by Huang et al. (2017), is another important deep learning architecture designed to improve feature reuse and alleviate the vanishing gradient problem. In DenseNet, each layer is directly connected to all previous layers, meaning that the input to each layer includes the feature maps from all earlier layers. This design improves

information flow and encourages the reuse of features across the network, making the model more efficient in terms of parameters and computation.

DenseNet replaces the traditional sequential connection of layers with **dense connections**, where the output of each layer is concatenated (not added) to the inputs of all subsequent layers. This results in improved gradient propagation and stronger feature propagation, especially in very deep networks. DenseNet models have been shown to require fewer parameters while achieving competitive or better performance compared to traditional networks [51].

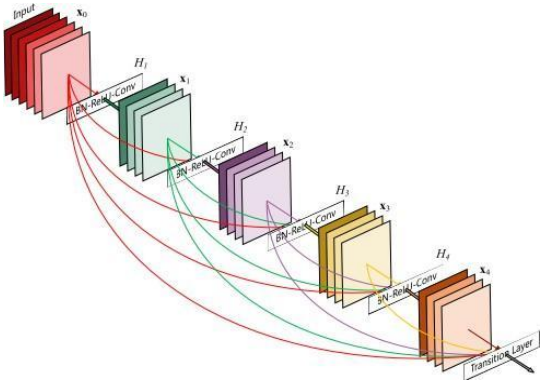


Figure 3.3: Dense connections [51].

In our experiments, we tested multiple versions of DenseNet, including DenseNet-121, DenseNet-169, and DenseNet-201, to evaluate their performance in flood detection.

- **MobileNet**: MobileNet is a family of lightweight deep neural networks developed by Google to run efficiently on mobile and embedded devices. The original MobileNet architecture (Howard et al., 2017) uses **depthwise separable convolutions**, which significantly reduce the number of parameters and computational cost compared to traditional convolutional layers. This design makes MobileNet models fast and suitable for real-time applications [52].

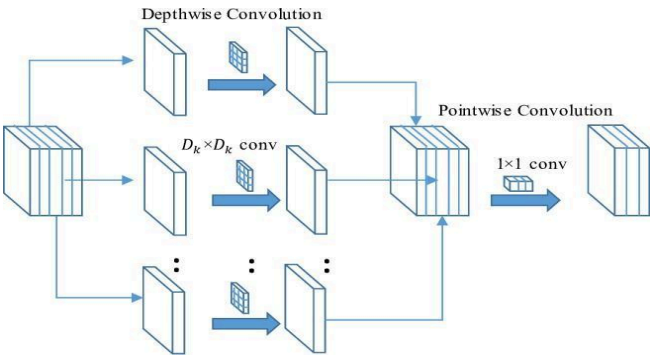


Figure 3.4: Depthwise separable convolution block [53].

The MobileNet family includes several improved versions: **MobileNetV2** introduces **inverted residuals and linear bottlenecks** to further enhance efficiency and performance, while **MobileNetV3** (in both Small and Large variants) combines architecture search techniques with the MobileNetV2 structure to achieve a better trade-off between accuracy and speed [52].

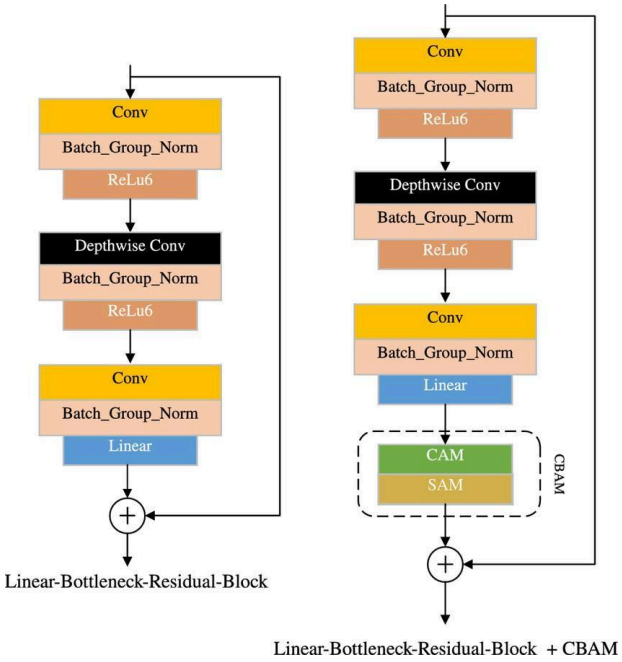


Figure 3.5: Inverted residual block and linear bottleneck block [54].

In our study, we evaluated the performance of multiple MobileNet versions including MobileNet, MobileNetV2, MobileNetV3Small, and MobileNetV3Large to determine the most effective model for flood detection.

- **Inception**: The Inception architecture, first introduced as part of the GoogleNet model, is known for its ability to extract rich features from images using an

efficient design. The key innovation in this family is the **Inception module**, which applies multiple convolution filters (1×1, 3×3, and 5×5) in parallel within the same layer. This allows the network to capture different types of information at various scales without significantly increasing the computational cost [55].

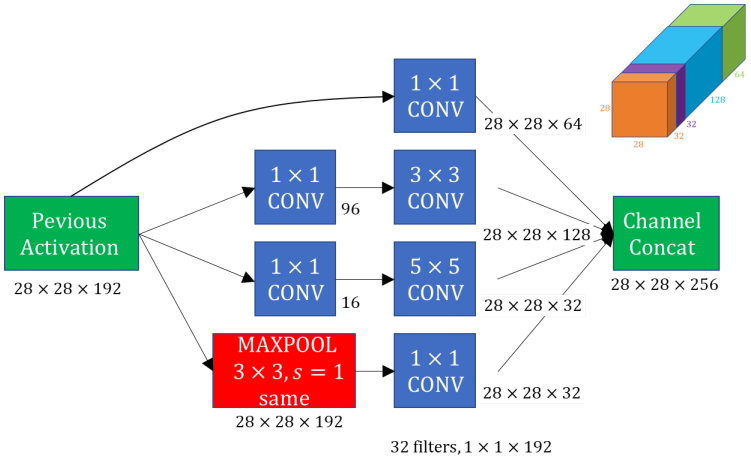


Figure 3.6: An example of an Inception module [56].

Later versions, such as **InceptionV3**, introduced several improvements, including factorized convolutions, dimensionality reduction using 1×1 convolutions, and better training techniques. Another important version, **Inception-ResNet-v2**, combines the Inception structure with **residual connections**, which help in training deeper networks by reducing the problem of vanishing gradients [55].

In our study, we tested both InceptionV3 and Inception-ResNet-v2 models using transfer learning to evaluate their performance in detecting floods.

All these models were chosen because they are widely used in image classification. Each one has different architectural features. We compared their performance using evaluation metrics to find the most accurate and efficient model for our task.

6. Model Architecture

The models tested in this study pre-trained on the large ImageNet dataset, which helps them recognize general features such as edges, textures, and shapes.

To adapt these models for flood detection, we used a transfer learning method. In this

approach, we freeze the original layers of the pre-trained model. Then, we add new layers on top for our specific task. In our case, we added:

- A GlobalAveragePooling2D layer to reduce the size of the output from the convolutional part.
- A final Dense layer with a sigmoid activation function to classify the image as either flooded or not flooded.

This setup was used to compare the performance of each model using standard transfer learning.

After this, we developed an enhanced version by adding a Quantum Layer to the model. This layer is based on concepts from quantum computing and was simulated using the Pennylane library. The steps of the enhanced model are:

- The output from the base model was reduced to 8 values (for 8 qubits) using a dense layer.
- These values were passed to the QuantumLayer, where quantum operations like rotation and entanglement were applied.
- Several additional Dense layers were added to further process the output.
- Finally, a Dense layer with a sigmoid activation function was used to make the binary prediction.

This quantum-enhanced architecture was applied to all the tested models.

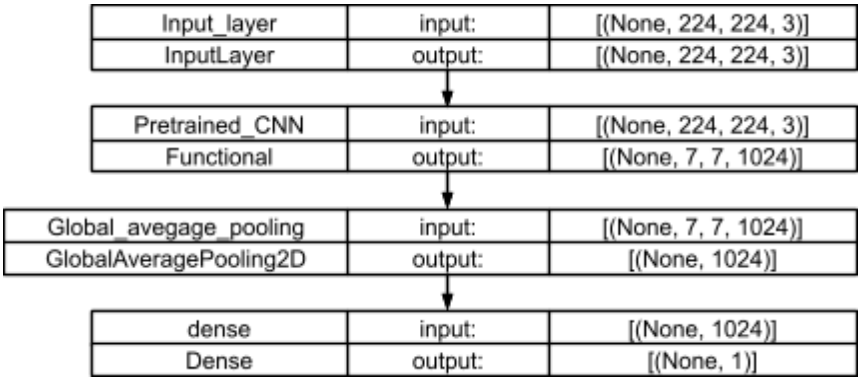


Figure 3.7: Model Architecture.

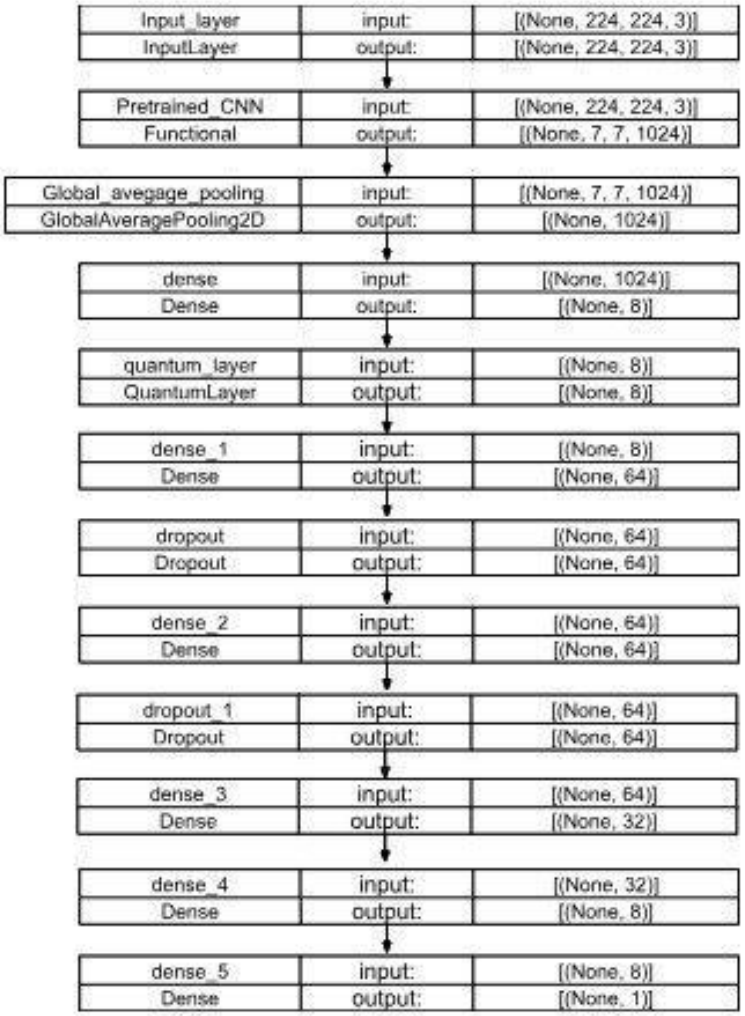


Figure 3.8: Hybrid Model Architecture.

7. Model Training

To train the selected deep learning models for flood detection, several important choices were made to ensure good learning and performance. A batch size of 32 was used because it provides a good balance between training speed and model accuracy. The models were trained for 25 epochs, which is enough to let the model learn from the data without overfitting. The Adam optimizer was chosen because it adjusts the learning rate during training, making it faster and more efficient than basic methods. Since the task is to classify images into two classes (flood or non-flood), the binary cross-entropy loss function was used, as it is standard for binary classification problems. A custom learning rate scheduler was also added to reduce the learning rate by a factor of 0.1 every 10 epochs to help the model escape local minima and learn better. The sigmoid activation

function was used in the final layer to output probabilities between 0 and 1, which is suitable for binary classification. During training, the model's performance was measured using accuracy on both training and validation sets. In addition, evaluation metrics such as F1-score, recall, precision, and Intersection over Union (IoU) were used to better understand the model's ability to correctly detect flooded areas. These choices follow best practices in deep learning for image classification tasks.

8. Conclusion

In this chapter, we presented the full methodology used to develop a deep learning model for flood detection. We explained the tools and libraries employed, how the data was collected and preprocessed, and the process for selecting and training several CNN-based models using transfer learning. Each step was designed to ensure the model could learn effectively from the dataset and achieve accurate results. The next chapter will focus on evaluating the model's performance using suitable metrics and presenting the results of the experiments.

Chapter 4

Results and Analysis

1. Introduction

In this chapter, we share and discuss the results of our flood detection experiments. After training several pre-trained deep learning models, we evaluated their performance using standard metrics such as accuracy, precision, recall, and F1-score. We also compared the performance of models trained with and without data augmentation and explored the impact of integrating a quantum layer. Through this analysis, we aim to identify which models perform best and highlight the challenges that still need to be addressed.

2. Results

2.1 VGG

VGG16

The performance results of the VGG16 model are provided below:

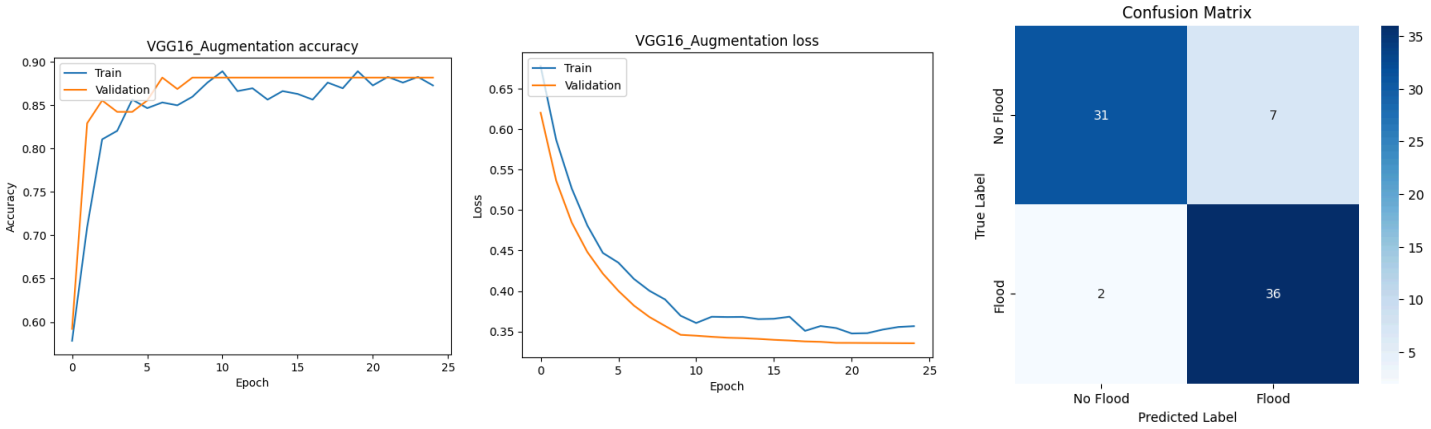


Figure 4.1: VGG16 Model results.

	precision	recall	f1-score	support
No Flood	0.94	0.82	0.87	38
Flood	0.84	0.95	0.89	38
accuracy	0.88			
macro avg	0.89	0.88	0.88	76
weighted avg	0.89	0.88	0.88	76

Table 4.1: Classification report of VGG16.

The VGG16 model performed well, reaching around 88% accuracy on the test set. Its good F1-score and IoU values show that it was able to classify flood and non-flood images quite reliably. One reason for this result is the use of data augmentation, which helped the model learn from more diverse examples and avoid overfitting. Also, the VGG16 architecture, with its deep layers, was able to capture important visual features. However, a few images were still misclassified, which could be due to confusing or unclear image content or because some features of flood and non-flood scenes looked too similar.

VGG19

Shown here are the outcomes from the VGG19 model:

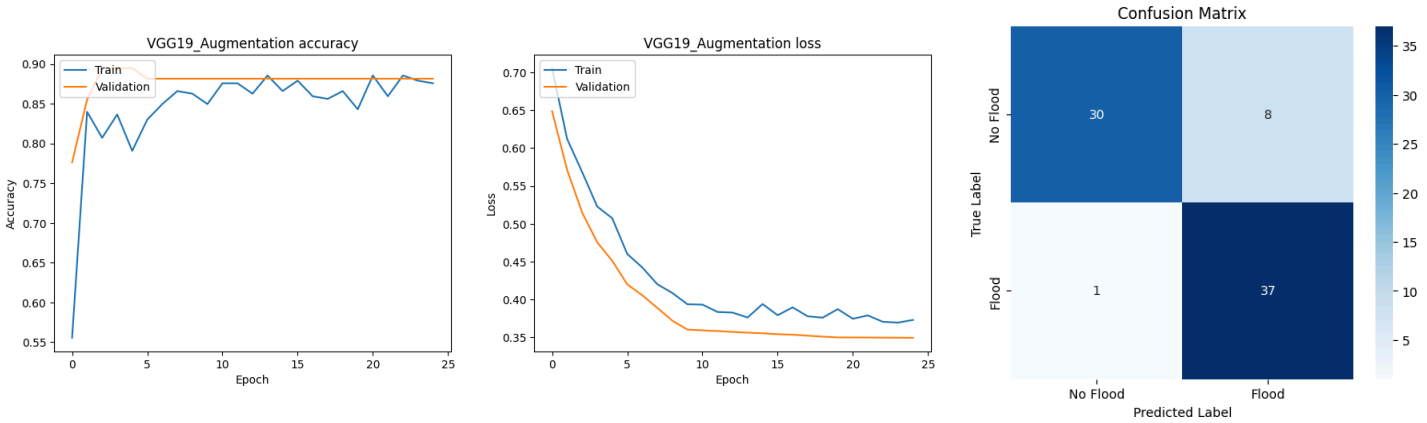


Figure 4.2: VGG19 Model results.

	precision	recall	f1-score	support
No Flood	0.97	0.79	0.87	38
Flood	0.82	0.97	0.89	38
accuracy	0.88			
macro avg	0.89	0.88	0.88	76
weighted avg	0.89	0.88	0.88	76

Table 4.2: Classification report of VGG19.

The VGG19 model gave a solid performance, reaching 88% accuracy on the test set, which is similar to VGG16. The F1-score (0.89) and IoU (0.80) confirm that the model was quite effective in distinguishing between flood and non-flood

images. One of the main reasons for this performance is the data augmentation, which helped the model see more diverse examples during training, improving its generalization.

However, there was still a noticeable difference between how the model handled each class. It was very good at detecting flood images (Recall = 0.97), but less accurate with non-flood images (Recall = 0.79). This could mean that the model sometimes gets "too careful" and plays it safe by labeling uncertain images as floods, just to avoid missing a real flood. That might explain why 8 non-flood images were misclassified as floods.

This kind of result is common in real-world problems where one class (like floods) is more critical, and the model learns to prioritize it, even if that means making some mistakes on the other class. Overall, VGG19 showed strong learning ability, especially when supported with data augmentation and careful training.

2.2 ResNet

ResNet50V2

The ResNet50V2 model produced the following results:

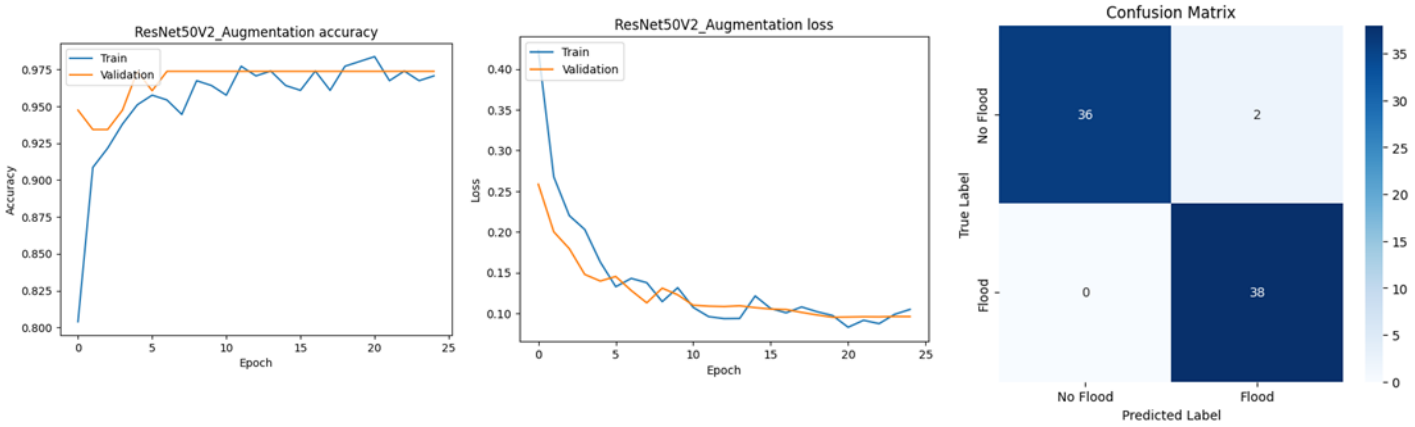


Figure 4.3: ResNet50V2 Model results.

	precision	recall	f1-score	support
No Flood	1	0.95	0.97	38

	precision	recall	f1-score	support
Flood	0.95	1	0.97	38
accuracy	0.97			
macro avg	0.97	0.97	0.97	76
weighted avg	0.97	0.97	0.97	76

Table 4.3: Classification report of ResNet50V2.

The model trained smoothly, improving steadily over the epochs. It reached 97.4% validation accuracy and performed equally well on the test set. It correctly identified all flood images (100% recall) and only misclassified 2 non-flood images, giving it a precision of 95%. The F1-score (0.97) and IoU (0.95) show that the model is highly reliable in detecting floods. Overall, the model is accurate, consistent, and shows great potential for real-world flood detection tasks.

ResNet101V2

Presented below is the evaluation of the ResNet101V2 model:

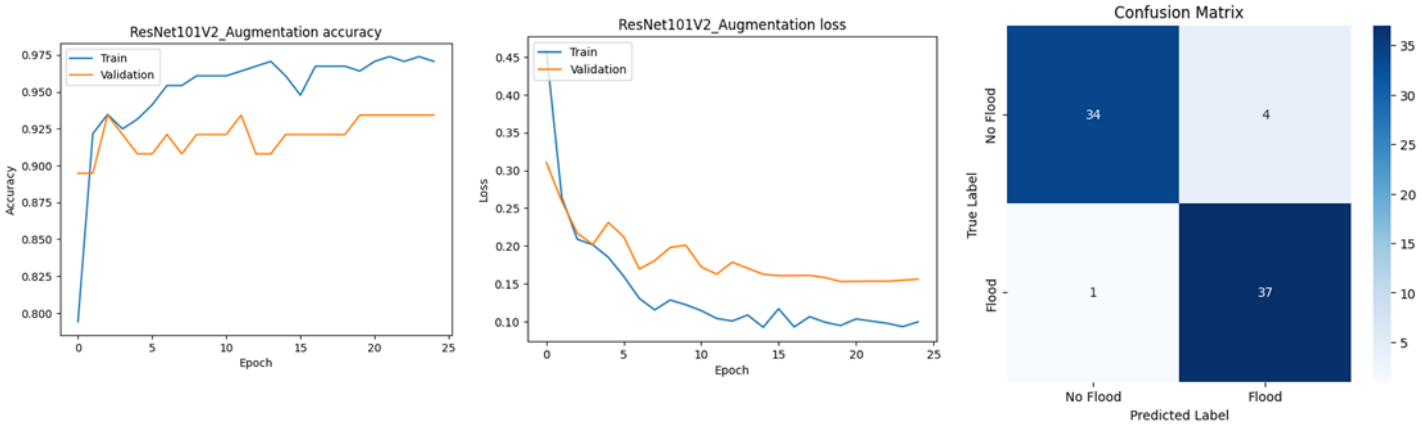


Figure 4.4: ResNet101V2 Model results.

	precision	recall	f1-score	support
No Flood	0.97	0.89	0.93	38
Flood	0.90	0.97	0.94	38
accuracy	0.93			
macro avg	0.94	0.93	0.93	76

	precision	**recall**	**f1-score**	**support**
weighted avg	0.94	0.93	0.93	76

Table 4.4: Classification report of ResNet101V2.

The ResNet101V2 model achieved 93.42% test accuracy, showing strong performance in classifying flood and non-flood images. It detected most flood cases correctly with a high recall of 97.37%, while maintaining a solid precision of 90.24%, leading to an F1-score of 0.94 and an IoU of 0.88. These results indicate that the model was both accurate and reliable. Most errors came from non-flood images being misclassified as floods, likely due to visual similarities like wet surfaces. Overall, the model performed consistently and shows strong potential for real-world flood detection.

ResNet152V2

The ResNet152V2 model generated the results listed below:

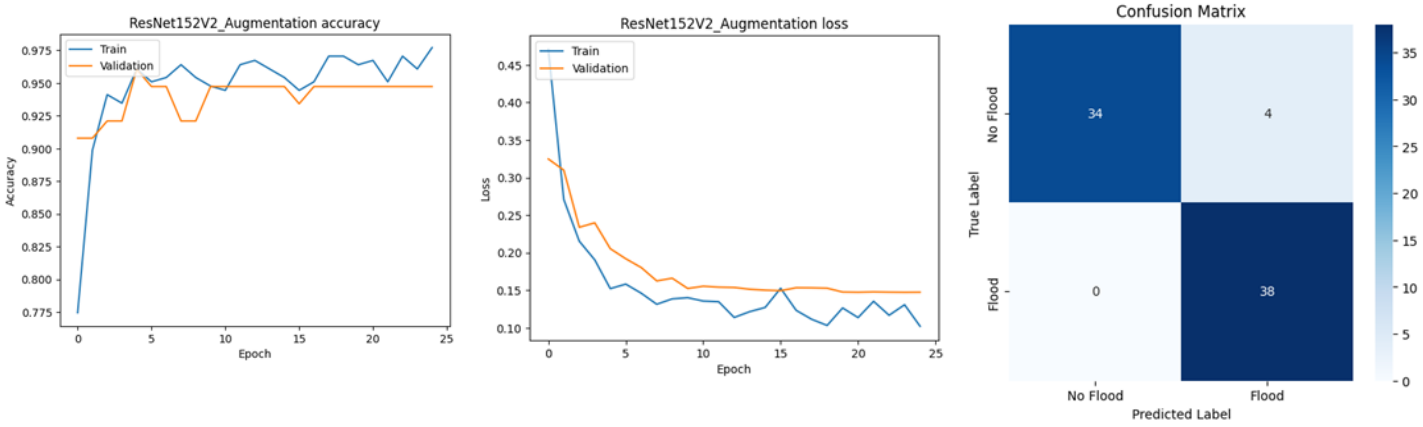


Figure 4.5: ResNet152V2 Model results.

	precision	**recall**	**f1-score**	**support**
No Flood	1	0.89	0.94	38
Flood	0.90	1	0.95	38
accuracy	0.95			
macro avg	0.95	0.95	0.95	76
weighted avg	0.95	0.95	0.95	76

Table 4.5: Classification report of ResNet152V2.

The ResNet152V2 model achieved a 95% test accuracy, demonstrating excellent performance in classifying flood and non-flood images. It reached 100% recall for flood images, meaning it detected all flood cases correctly, and had a precision of 90.48%, indicating a few false positives. The F1-score was 0.95 and the IoU reached 0.90, confirming a strong balance between precision and recall. Misclassifications were minimal, with only 4 non-flood images wrongly labeled as floods. Overall, the model showed robust detection capability and is highly suitable for flood monitoring tasks.

2.3 DenseNet

DenseNet121

The following output was obtained using the DenseNet121 model:

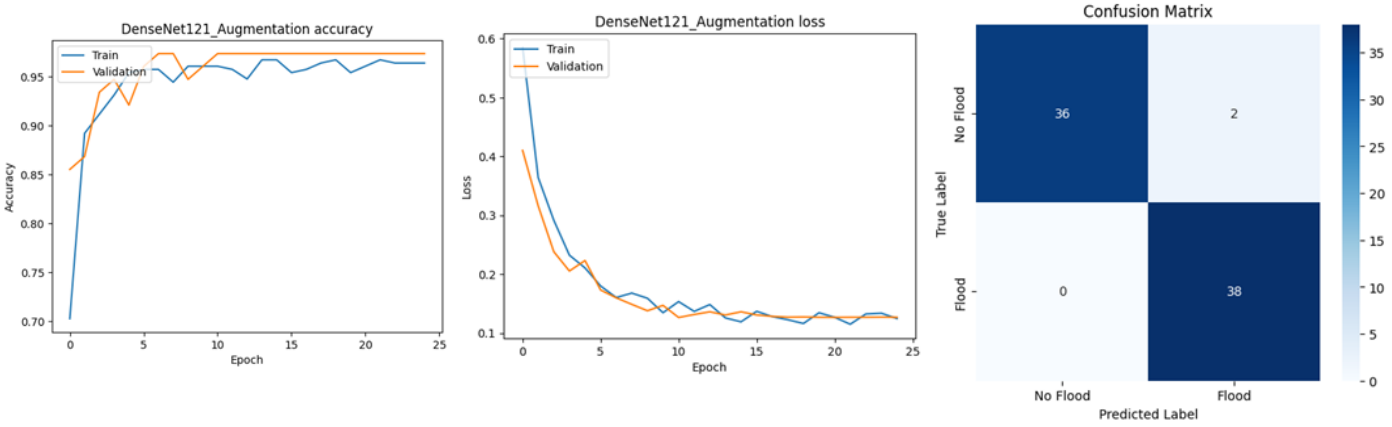


Figure 4.6: DenseNet121 Model results.

	precision	**recall**	**f1-score**	**support**
No Flood	1	0.95	0.97	38
Flood	0.95	1	0.97	38
accuracy	0.97			
macro avg	0.97	0.97	0.97	76
weighted avg	0.97	0.97	0.97	76

Table 4.6: Classification report of DenseNet121.

The DenseNet121 model gave perfect results in detecting flood images. During training, the accuracy improved steadily and reached a strong validation accuracy of 97.37%, without overfitting. On the test set, the model achieved high scores: 95% precision, 100% recall, and a strong F1-score of 97.44%. This means it correctly found all the real flood images, which is very important in real-life flood situations. The confusion matrix showed only two mistakes where the model thought there was a flood but there wasn't a small error that's acceptable if it helps avoid missing actual floods. Overall, DenseNet121's ability to reuse features seems to have helped it perform well in this task.

Densenet169

Here are the findings from the DenseNet169 model:

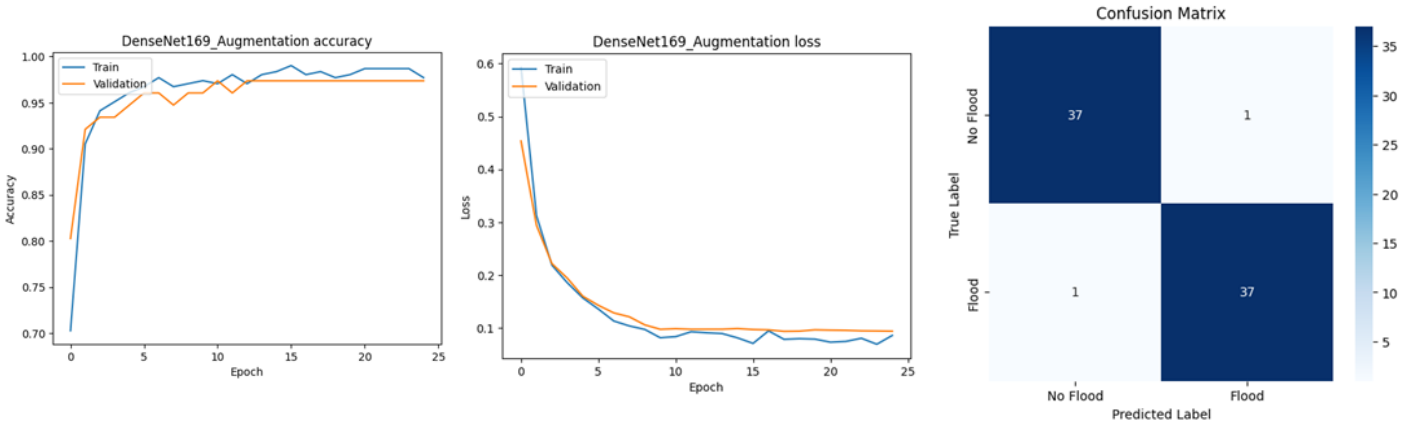


Figure 4.7: DenseNet169 Model results.

	precision	recall	f1-score	support
No Flood	0.97	0.97	0.97	38
Flood	0.97	0.97	0.97	38
accuracy	0.97			
macro avg	0.97	0.97	0.97	76
weighted avg	0.97	0.97	0.97	76

Table 4.7: Classification report of DenseNet169.

The DenseNet169 model demonstrated excellent performance in flood detection, with strong accuracy, precision, recall, and F1-score all reaching 97.37%. Throughout training, the model showed steady improvement in both accuracy and loss, with no signs of overfitting. The validation accuracy remained high and stable after epoch 10, and the final test results confirmed the model's robustness. The confusion matrix revealed only two misclassifications one flood image wrongly predicted as no flood and one no-flood image incorrectly classified as flood indicating that the model is highly reliable for real-world flood detection tasks. DenseNet169's deep connectivity and feature reuse likely contributed to its consistent and balanced performance.

DenseNet201

The performance results of the DenseNet201 model are provided below:

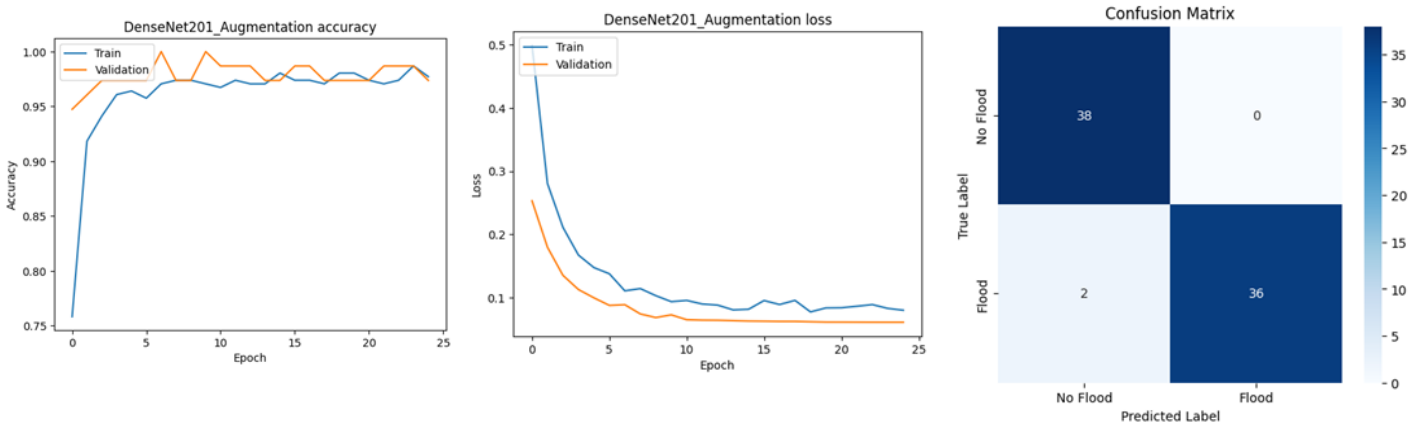


Figure 4.8: DenseNet201 Model results.

	precision	**recall**	**f1-score**	**support**
No Flood	0.95	1	0.97	38
Flood	1	0.95	0.97	38
accuracy	0.97			
macro avg	0.97	0.97	0.97	76
weighted avg	0.97	0.97	0.97	76

Table 4.8: Classification report of DenseNet201.

The DenseNet201 model performed well in detecting floods. It correctly identified nearly all flood and non-flood cases, showing it's both smart and careful, especially since it made no false flood alarms. This kind of precision is important in real-world situations where a wrong alert can waste resources. Even though it missed just 2 actual flood images, its overall performance stayed strong and consistent throughout training. These results suggest the model is not just accurate, but also reliable and ready to be used in practical flood monitoring systems.

2.4 MobileNet

MobileNet

The results from the MobileNet model are presented here:

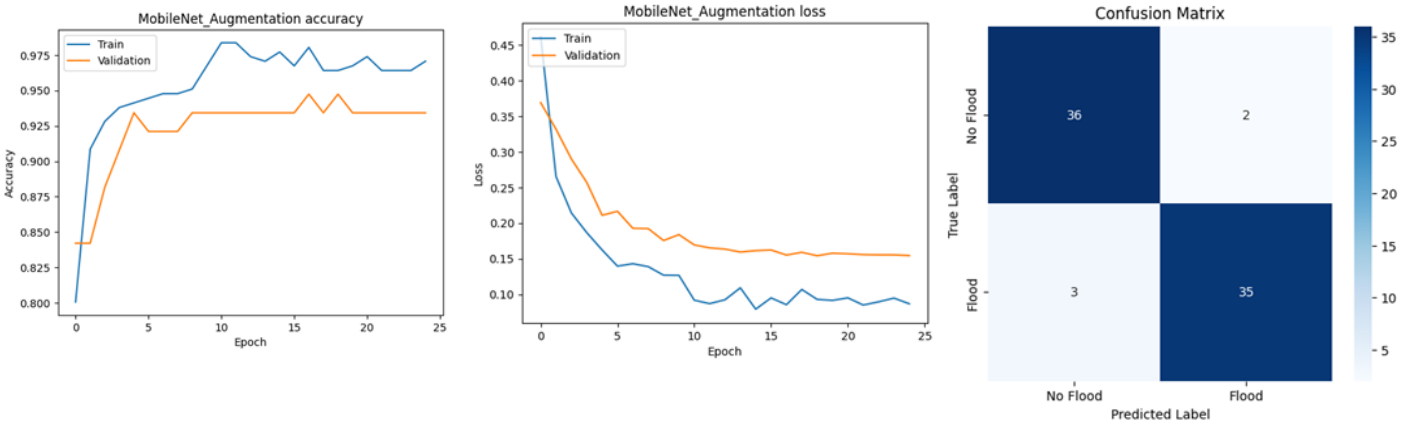


Figure 4.9: MobileNet Model results.

	precision	**recall**	**f1-score**	**support**
No Flood	0.92	0.95	0.94	38
Flood	0.95	0.92	0.93	38
accuracy	0.97			
macro avg	0.93	0.93	0.93	76
weighted avg	0.93	0.93	0.93	76

Table 4.9: Classification report of MobileNet.

MobileNet did a great job spotting flood and non-flood areas in the images, with strong and balanced results across all metrics. It was especially accurate and consistent, making only a handful of mistakes just three flood images were missed, and two non-flood ones were wrongly flagged. The training process went smoothly, with the model getting better over time and staying stable in later rounds. Overall, it proves that even a lightweight model like MobileNet can be both fast and reliable, which is great news for real-world flood detection, where quick decisions really matter.

MobileNetV2

Below are the outcomes produced by the MobileNetV2 model:

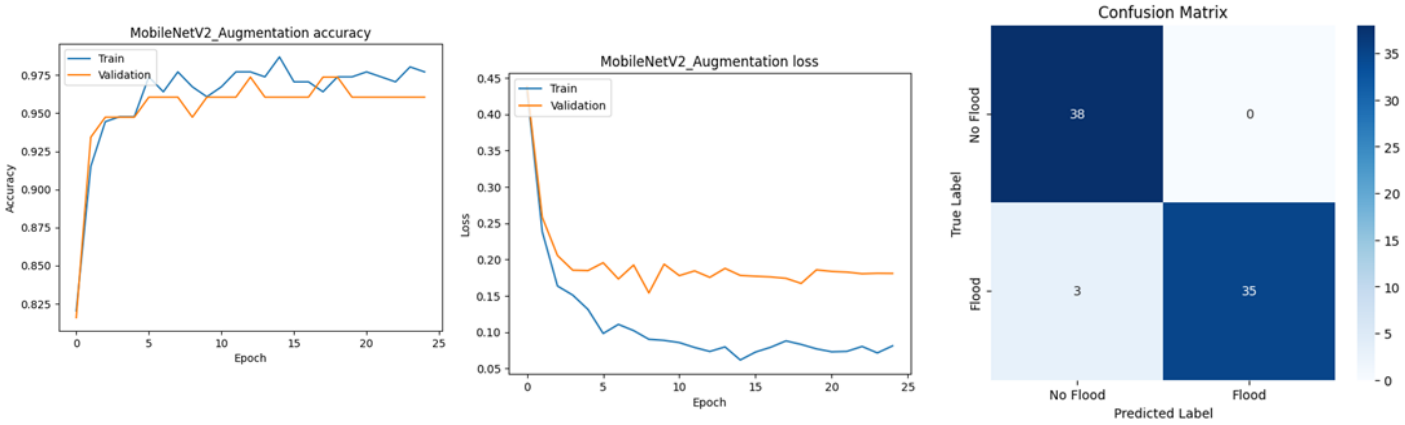


Figure 4.10: MobileNetV2 Model results.

	precision	**recall**	**f1-score**	**support**
No Flood	0.93	1	0.96	38
Flood	1	0.92	0.96	38
accuracy	0.96			
macro avg	0.96	0.96	0.96	76
weighted avg	0.96	0.96	0.96	76

Table 4.10: Classification report of MobileNetV2.

The MobileNetV2 model gave excellent results in detecting floods. After 25 training epochs, it reached a high validation accuracy of 96.05%, showing that it learned to recognize flood and non-flood images very well. On the test set, it achieved perfect precision (1.00), which means it correctly identified all the real flood cases it predicted, and a recall of 92.11%, meaning it found most of the actual flood images. The F1-score was 95.89%, and the Intersection over Union (IoU) score was also high at 92.11%. The confusion matrix showed that the model made no false flood predictions and only missed 3 actual flood cases. These results suggest that MobileNetV2 is not only accurate but also efficient, making it a good choice for practical and real-time flood detection tasks.

2.5 Inception

InceptionV3

The results from the InceptionV3 model are presented here:



Figure 4.11: InceptinoV3 Model results.

	precision	recall	f1-score	support
No Flood	0.88	0.92	0.90	38
Flood	0.92	0.87	0.89	38
accuracy	0.89			
macro avg	0.90	0.89	0.89	76
weighted avg	0.90	0.89	0.89	76

Table 4.11: Classification report of InceptionV3.

The InceptionV3 model performed quite well in detecting flood and non-flood images. It achieved a test accuracy of 89.47%, which shows that it was able to generalize effectively to new, unseen images. Its precision was 91.67%, meaning it correctly identified most flood cases without many false alarms, while its recall was 86.84%, showing it managed to detect most of the actual flood events. The F1-score, which balances both precision and recall, was 89.19%, and the IoU score reached 80.49%, confirming a strong overlap between predicted and actual flood regions. According to the confusion matrix, the model correctly classified 33 out of 38 flood images and 35 out of 38 non-flood images, only confusing a few cases. Overall, InceptionV3 performed reliably, with high accuracy and

generalization, making it a good option for flood detection, even if it is slightly less accurate than some other models tested.

InceptionResNetV2

The InceptionResNetV2 model produced the results shown below:

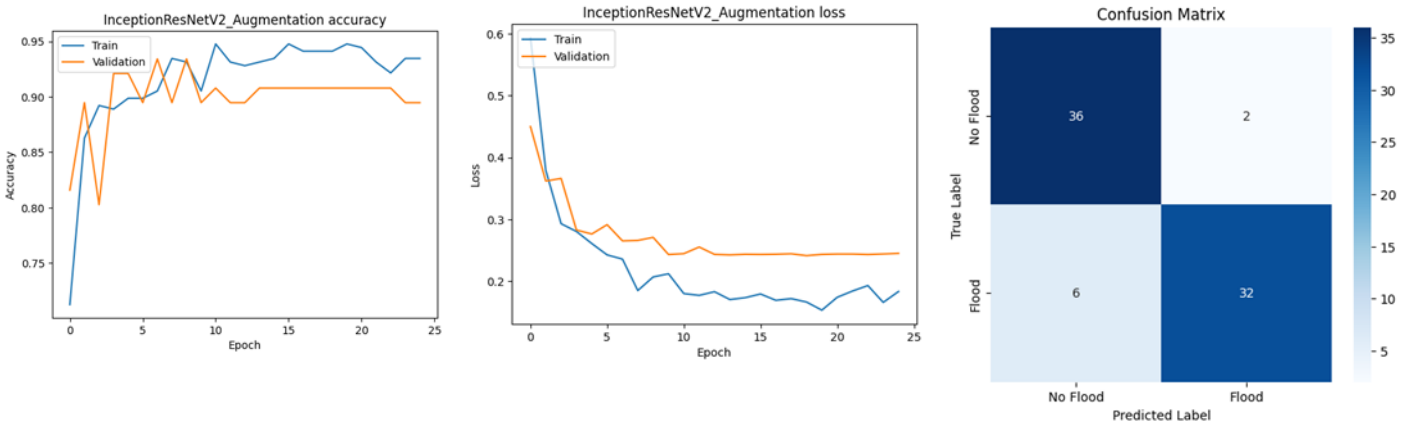


Figure 4.12:InceptionResNetV2 Model results.

	precision	recall	f1-score	support
No Flood	0.86	0.95	0.90	38
Flood	0.94	0.84	0.89	38
accuracy	0.89			
macro avg	0.90	0.89	0.89	76
weighted avg	0.90	0.89	0.89	76

Table 4.12: Classification report of InceptionResNetV2.

The InceptionResNetV2 model did a good job in telling the difference between flood and non-flood images, reaching an accuracy of 89%. It was very good at correctly finding flood images (94% precision) and also found most of them (84% recall). Out of 76 test images, it only made a few mistakes 6 flood images were marked as non-flood, and 2 non-flood images were marked as flood. The model's F1-score was 88.9%, and it got an IoU score of 80%, which means it was quite strong and reliable for this kind of flood detection work.

3. Comparison between models with and without Data Augmentation

This study compared the accuracy of several pre-trained CNN models for flood image classification, both with and without data augmentation. In most cases, data augmentation helped improve the models' accuracy by adding more variety to the training images. For example, DenseNet121, DenseNet169, and DenseNet201 all reached 97.37% accuracy with augmentation, which is higher than their results without augmentation. MobileNetV2 also showed improvement, increasing from 94.74% to 96.05%. These results suggest that models with dense or lightweight architectures benefit from having more diverse training data. However, some deep models, such as ResNet50V2 and ResNet101V2, performed better without augmentation, achieving accuracies of 98.68% and 97.37%, respectively. This may be because these models already have strong regularization, and the extra data variation may have confused them. In contrast, MobileNetV3Small performed worse with augmentation, dropping from 77.63% to 67.11%, which could mean it struggled with the added complexity. Overall, data augmentation was beneficial for most models, but its effectiveness depends on the model's structure and ability to learn from more diverse data.

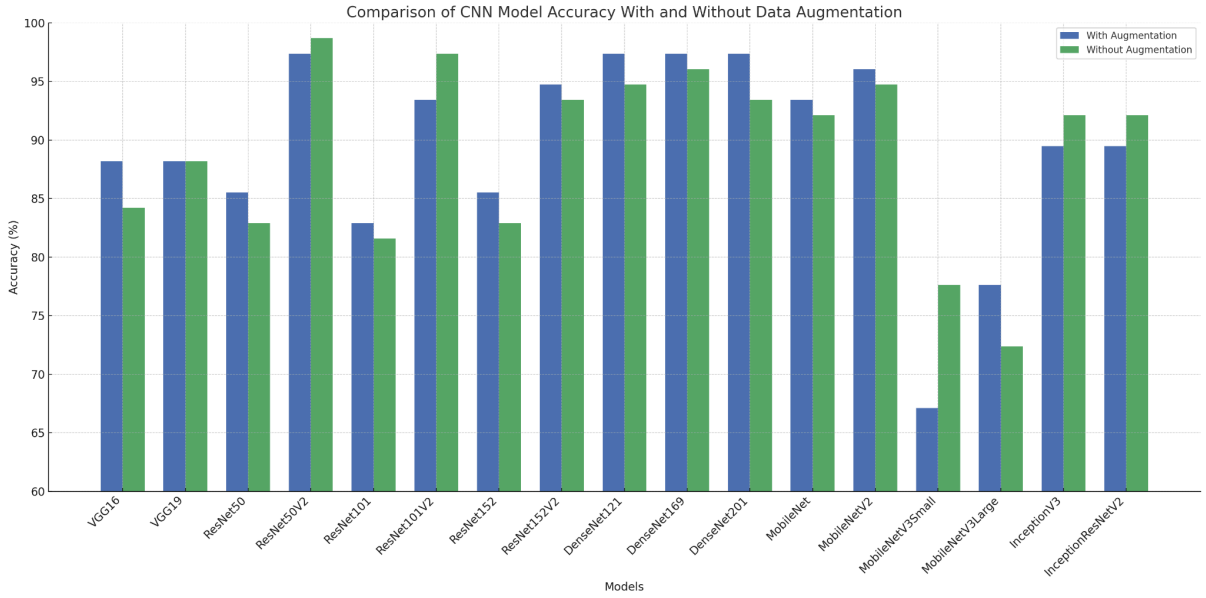


Figure 4.13: Comparison of the models Accuracy with and without Data Augmentation.

4. Results of Hybrid Models

The following table presents the evaluation results of various hybrid deep learning models used for flood detection. Each model was assessed using key performance metrics, including Accuracy, Precision, Recall, F1-score, and Intersection over Union

(IoU). These results highlight the effectiveness of different models when combined with the proposed hybrid approach.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	IoU (%)
VGG16	88.16	83.72	94.74	88.89	80.00
VGG19	89.47	84.09	97.37	90.24	82.22
ResNet50V2	98.68	97.44	100.00	98.70	97.44
ResNet101V2	93.42	90.00	94.74	92.31	85.71
ResNet152V2	96.05	92.68	100.00	96.20	92.68
DenseNet121	97.37	95.00	100.00	97.44	95.00
DenseNet169	96.05	94.87	97.37	96.10	92.50
DenseNet201	97.37	95.00	100.00	97.44	95.00
MobileNet	97.37	94.44	89.47	91.89	85.00
MobileNetV2	96.05	97.30	94.74	96.00	92.31
InceptionV3	92.11	92.11	92.11	92.11	85.37

Table 4.13: Evaluation Results of Hybrid Deep Learning Models for Flood Detection.

Among all the hybrid deep learning models tested for flood detection, the ResNet50V2 model is the best, with the highest accuracy of 98.68%, perfect recall of 100%, and a very high IoU of 97.44%, which means it can accurately mark flood locations. The DenseNet121 and DenseNet201 models also perform very well, both with 97.37% accuracy, perfect recall (100%), and IoU of 95.00%, showing they are good at detecting floods without missing much. The deeper ResNet152V2 model is also strong, with 96.05% accuracy and perfect recall, and an IoU of 92.68%. The ResNet101V2 model is a bit lower in accuracy (93.42%) and recall (94.74%), but still does a good job. Among MobileNet models, MobileNetV2 is balanced with 96.05% accuracy, very high precision (97.30%), and recall of 94.74%, while MobileNet has similar accuracy

(97.37%) but lower recall (89.47%), meaning it missed some flooded areas. The older VGG models, VGG16 and VGG19, have lower accuracy, around 88–89%. They find most flooded spots, but are less precise and accurate in marking the flood areas correctly. Lastly, The InceptionV3 model also performs well, achieving 92.11% accuracy, precision, recall, and F1-score, with IoU of 85.37%.

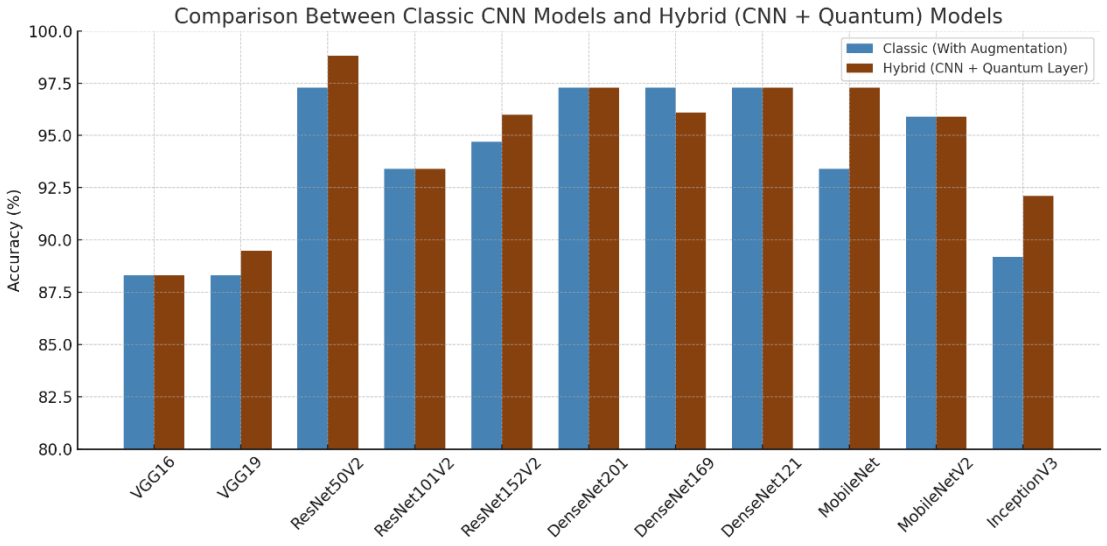


Figure 4.14: Comparison between the classic model and the hybrid model.

The chart comparing classic CNN models (with data augmentation) and hybrid models that include a quantum layer shows that adding quantum parts can sometimes improve accuracy, but not always. Models like ResNet50V2, ResNet152V2, and MobileNet performed better when combined with quantum layers, which means these models can benefit from the special abilities of quantum computing, like handling complex patterns. DenseNet models stayed strong in both versions, showing they are stable no matter the setup. However, some models like DenseNet169 actually did worse with the quantum layer, which suggests that quantum parts don't always work well with every model, While InceptionV3 showed a clear improvement, reaching 92.11 % accuracy in its hybrid version. In short, using quantum layers in CNNs can help in some cases, but it depends on the model and its architecture.

5. Comparison of Results

5.1 Comparison of Classical Models with the state of the art

The table below is a summary for comparing results (F1-score) of our models with other related works:

model	Nouioua Tarek [36]	Cherif Elarbi [38]	Our Work
VGG16	60	94.31	88.89
ResNet50	68.66	88.10	86.08
InceptionV3	58.33	93.67	89.19
MobileNetV2	-	94.39	95.89
DenseNet201	-	88.89	97.30

Table 4.12: Comparison of classical models with the state of the art.

The table shows that our models perform better than Nouioua Tarek's models in all tested architectures, especially VGG16 and InceptionV3. Compared to Cherif Elarbi's models, our results are a little lower for some models, like VGG16 and ResNet50, but better for others, especially MobileNetV2 and DenseNet201, where we achieved the highest scores.

5.2 Comparison of Hybrid Models with the state of the art

The table below is a summary for comparing the results (Accuracy%) of our models with other related work:

model	Cherif Elarbi [38]	Our Work
VGG16	81	88.16
InceptionV3	90	92.11
MobileNetV2	95	96.05
DenseNet201	98	97.37
DenseNet169	97	96.05
DenseNet121	95	97.37

Table 4.12: Comparison of Hybrid models with the state of the art.

The table compares the accuracy of our hybrid models with the models of Cherif Elarbi [38]. Our models perform better in most cases, with higher accuracy for VGG16, InceptionV3, and MobileNetV2. For DenseNet models, Cherif Elarbi's DenseNet201 and DenseNet169 have slightly higher accuracy, while our DenseNet121 model performs better. Overall, our hybrid models show competitive or improved accuracy compared to previous work.

6. Conclusion

In this chapter, we discussed the evaluation of our models' results and compared their performance with previous work. The analysis highlights the strengths and limitations of our approach in classifying flood and non-flood images.

General Conclusion

This thesis studies the use of deep learning and transfer learning methods to detect floods. Because floods are dangerous and occur frequently, it is crucial to locate them quickly and accurately. This work tested different deep learning models, such as VGG, ResNet, DenseNet, MobileNet, and Inception, using a balanced dataset of flood and non-flood images.

The models were trained using transfer learning, which allowed us to use already-trained models to save time and improve results. The results showed that models like ResNet50V2 and DenseNet gave very good accuracy (up to 97%). These models were able to detect flooded areas with high precision and few mistakes.

The thesis also tried using a quantum-inspired layer inside some models to see if it could improve performance. The hybrid models gave encouraging results, especially when the amount of training data was small.

However, the study had some limitations. The dataset was small and only included certain types of images. Also, the quantum part is still new and needs more testing and research.

In the future, the system can be improved in many ways:

- By using larger and more diverse datasets that include different types of floods and various image sources.
- By expanding the method to detect other types of natural disasters.
- By predicting floods before they happen, using extra information such as landforms, geography, rainfall, soil type, and environmental data.
- By using Explainable AI to better understand why the model makes certain decisions, we can help improve the system and make it more trustworthy.

In short, this research shows that deep learning and transfer learning are useful for flood detection. With more development, such systems can support early warning, help emergency responders, and reduce the impact of natural disasters on people and the environment.

Bibliography

[1] National Institute of Biomedical Imaging and Bioengineering. *Optical Imaging*. Available at: https://www.nibib.nih.gov/science-education/science-topics/optical-imaging (accessed April 13, 2025).

[2] NASA Earthdata. *Synthetic Aperture Radar (SAR)*. Available at: https://www.earthdata.nasa.gov/learn/earth-observation-data-basics/sar (accessed April 13, 2025).

[3] GISGeography. *Spatial Resolution vs Spectral Resolution*. Available at: https://gisgeography.com/spatial-resolution-vs-spectral-resolution/ (accessed April 13, 2025).

[4] OnGeo Intelligence. *The Role of Temporal Resolution in Satellite Observation*. Available at: https://ongeo-intelligence.com/blog/temporal-resolution-in-satellite-earth-observation (accessed April 13, 2025).

[5] Wikipedia. *Satellite Revisit Period*. Available at: https://en.wikipedia.org/wiki/Satellite_revisit_period#cite_note-:0-1 (accessed April 13, 2025).

[6]IBM. *What is Artificial Intelligence?* Available at: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence (accessed April 11, 2025).

[7] Spiceworks. *What is Artificial Intelligence (AI)?*. Available at: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ai/?utm_source=chatgpt.com (accessed April 11, 2025).

[8] Mahesh Batta. *Machine Learning Algorithms – A Review*. International Journal of Science and Research (IJSR), vol. 9, no. 1, pp. 381–386, 2020.

[9] 7wData. *Types of Machine Learning Algorithms*. Available at: https://7wdata.be/visualization/types-of-machine-learning-algorithms-2/ (accessed April 11, 2025).

[10] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 4th edn. Pearson (2020).

[11] Kozan, M. *Supervised and Unsupervised Learning (An Intuitive Approach)*. Available at: https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644 (accessed April 11, 2025).

[12] Patil, S.S. *Reinforcement Learning*. Available at: https://www.linkedin.com/pulse/reinforcement-learning-sumit-s-patil (accessed April 11, 2025).

[13] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553),

436–444. https://doi.org/10.1038/nature14539

[14]Knoldus. *Architecture of Artificial Neural Network*. Available at: https://blog.knoldus.com/architecture-of-artificial-neural-network/ (accessed April 11, 2025).

[15] Murtagh, F.: Multilayer perceptrons for classification and regression. *Neurocomputing* **2**(5–6), 183–197 (1991). https://doi.org/10.1016/0925-2312(91)90023-5

[16] Indrawan, M.R. *Understanding Multilayer Perceptron: The Foundation of Modern Neural Networks*. Available at: https://medium.com/@muhammadraflyindrawan/understanding-multilayer-perceptron-the-foundation-of-modern-neural-networks-5b8bf757db99 (accessed April 11, 2025).

[17] Akgül, İ.: Activation Functions Used in Artificial Neural Networks. In: Academic Studies in Engineering, pp. 41–58. Gece Kitaplığı (2023).

[18] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[19] O'Shea, K., Nash, R.: An Introduction to Convolutional Neural Networks. arXiv preprint arXiv:1511.08458 (2015).

[20] QuarkML. *Convolutional Neural Network (CNN): Architecture Explained*. Available at: https://www.quarkml.com/2023/06/introduction-to-convolutional-neural-networks.html (accessed April 11, 2025).

[21] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: *International Conference on Artificial Neural Networks*, pp. 270–279. Springer, Cham (2018).

[22] Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. *Journal of Big Data* **3**(1), 9 (2016).

[23] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. *Proceedings of the IEEE* **109**(1), 43–76 (2021).

[24] Research Computing Center, University of North Carolina at Chapel Hill: Introduction to Quantum Computers. (2022).

[25] Hossin, M., Sulaiman, M.N.: A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, vol. 5, no. 2, pp. 1–11 (2015).

[26] Padilla, R., Netto, S.L., da Silva, E.A.: A Survey on Performance Metrics for Object-Detection Algorithms. In: *Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237–242. IEEE (2020).

[27] McFeeters, S.K.: The use of the Normalized Difference Water Index (NDWI) in the

delineation of open water features. *International Journal of Remote Sensing* 17(7), 1425–1432 (1996).

[28] Xu, H.: Modification of Normalized Difference Water Index (NDWI) to enhance open water features in remotely sensed imagery. *International Journal of Remote Sensing* 27(14), 3025–3033 (2006).

[29] Gessesse, A.A., Melesse, A.M.: Temporal relationships between time series CHIRPS-rainfall estimation and eMODIS-NDVI satellite images in Amhara Region, Ethiopia. In: Melesse, A.M., Abtew, W., Senay, G.B. (eds.) *Extreme Hydrology and Climate Variability*, pp. 81–92. Elsevier, Amsterdam (2019).

[30] Zhu, X., Tuia, D., Mou, L., Xia, G.S., Zhang, L., Xu, F., Fraundorfer, F.: Deep learning in remote sensing: A comprehensive review. *IEEE Geoscience and Remote Sensing Magazine* 5(4), 8–36 (2017).

[31] Torres, R., Snoeij, P., Geudtner, D., Bibby, D., Davidson, M., Attema, E., et al.: GMES Sentinel-1 mission. *Remote Sensing of Environment* 120, 9–24 (2012).

[32] Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., et al.: Sentinel-2: ESA's optical high-resolution mission for GMES operational services. *Remote Sensing of Environment* 120, 25–36 (2012).

[33] Bischke, B., Bhardwaj, P., Gautam, A., Helber, P., Borth, D., Dengel, A.: Detection of Flooding Events in Social Multimedia and Satellite Imagery using Deep Neural Networks. In: MediaEval (September 2017).

[34] Wu, X., Zhang, Z., Xiong, S., Zhang, W., Tang, J., Li, Z., An, B., Li, R.: A near-real-time flood detection method based on deep learning and SAR images. Remote Sens. 15(8), 2046 (2023).

[35] Stateczny, A., Praveena, H. D., Krishnappa, R. H., Chythanya, K. R., Babysarojam, B. B.: Optimized deep learning model for flood detection using satellite images. Remote Sens. 15(20), 5037 (2023).

[36] Nouioua, T.: Flood Detection in Satellite Images using Deep Learning. (2023)

[37] Lemenkova, P.: Deep Learning Methods of Satellite Image Processing for Monitoring of

Flood Dynamics in the Ganges Delta, Bangladesh. Water 16(8), 1141 (2024)

[38] Cherif, E.: Deep learning-enhanced satellite-based flood detection for early warning and prevention. Master thesis, University Larbi Tébessi–Tébessa (2024)

[39] Wedajo, G. K., Lemma, T. D., Fufa, T., Gamba, P.: Integrating Satellite Images and Machine Learning for Flood Prediction and Susceptibility Mapping for the Case of Amibara, Awash Basin, Ethiopia. Remote Sens. 16(12), 2163 (2024)

[40] Mohamadiazar, N., Ebrahimian, A., Hosseiny, H.: Integrating deep learning, satellite image processing, and spatial-temporal analysis for urban flood prediction. J. Hydrology. 639, 131508 (2024)

[41] More, N., Pawar, A., Kanthe, A., Gangane, J., Padiya, P.: Automatic Detection of Flood Extent and Volume Estimation from Sentinel-2 Satellite Images using Deep Learning Techniques in India. *Panamerican Mathematical Journal* 34(4), 416 (2024).

[42] Doan, T. N., Le-Thi, D. N.: A novel deep learning model for flood detection from synthetic aperture radar images. Journal of Advances in Information Technology. 16(1) (2025)

[43] Manaf, A., Mughal, N., Talpur, K. R., Talpur, B. A., Mujtaba, G., Talpur, S. R.: Aerial Image Classification in Post-Flood Scenarios Using Robust Deep Learning and Explainable Artificial Intelligence. IEEE Access (2025)

[44] A.L. Sayeth Saabith, T. Vinothraj, and M.M.M. Fareez. Popular Python Libraries and Their Application Domains. *International Journal of Advance Engineering and Research Development*, vol. 7, no. 11, Nov. 2020..

[45] Google. *Colaboratory: Frequently Asked Questions*. https://research.google.com/colaboratory/faq.html. Accessed May 17, 2025.

[46] Google. *How to use Google Drive*. https://support.google.com/drive/answer/2424384. Accessed May 17, 2025.

[47] Manaswi, N.K.: Understanding and Working with Keras. In: *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition with TensorFlow and Keras*, pp. 31–43. Apress, Berkeley, CA (2018).

[48] Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale

Image Recognition. arXiv preprint arXiv:1409.1556 (2014).

[49] Boesch, G.: Very Deep Convolutional Networks (VGG) Essential Guide. Viso.ai. October 6, 2021. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/. Accessed May 14, 2025.

[50] He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, Las Vegas, NV, USA (2016).

[51] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely Connected Convolutional Networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708. IEEE, Honolulu, HI, USA (2017).

[52] Howard, A.G.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017).

[53] Hossain, D., Imtiaz, M. H., Ghosh, T., Bhaskar, V., Sazonov, E.: Real-Time Food Intake Monitoring Using Wearable Egocentric Camera. In: *Proceedings of the 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4191–4195. IEEE, Montreal, QC, Canada (2020).

[54] Chen, M., Dai, H., Wei, S., Hu, Z.: Linear-ResNet GAN-based Anime Style Transfer of Face Images. *Signal, Image and Video Processing* **17**(6), 3237–3245 (March 2023).

[55] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (AAAI 2017), pp. 4278–4284. AAAI Press, San Francisco, CA, USA (February 2017).

[56] DataHacker RS: Inception Module. *DataHacker.RS Blog*, 14 November 2018. Available at: https://media5.datahacker.rs/2018/11/inception_module.png. Accessed May 16, 2025.

[57] Belgian Science Policy Office. *III.2.3 Résolution temporelle*. Available at: https://eo.belspo.be/fr/iii23-resolution-temporelle

[58] Alamy. *Quantum Superposition States*. https://www.alamy.com/stock-photo/quantum-superposition-states.html

[59] Universe Review. *Quantum Computing*. Available at: https://universe-review.ca/R13-11-QuantumComputing.htm (Accessed: 27 May 2025)

[60] BlueQubit. *Everything You Need to Know About the Quantum Circuit*. Available at: https://www.bluequbit.io/quantum-circuit (Accessed: 27 May 2025)

[61] Roell, J. *Demystifying Quantum Gates One Qubit At A Time*. Available at: https://jasonroell.com/2018/02/27/demystifying-quantum-gates%E2%80%8A-%E2%80%8A

one-qubit-at-a-time/ (Accessed: 27 May 2025)

[62] NOREA. *Quantum Computing Explained*. Available at: https://www.norea.nl/uploads/bfile/b90d6290-5f15-4736-9cb8-4a001d1539a8 (Accessed: 27 May 2025)

[63] Medium. *Quantum Gates & Circuits*. Available at: https://medium.com/codex/quantum-gates-circuits-8cfdf293615b (Accessed: 27 May 2025)

[64] Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M.S., Alonso-Linaje, G., AkashNarayanan, B., Asadi, A., Arrazola, J.M.: PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations. arXiv preprint arXiv:1811.04968 (2018)