MASTER'S THESIS

Systems and Multimedia speciality

Deep learning-enhanced satellite-based Flood detection for early Warning and prevention.

Prepared by:

Cherif Elarbi

Supervised by:

Dr. Nouioua Tarek

Before the jury:

Dr. Benour Akrem

Dr. Achouri Mounir

*University Of Larbi Tebessi Tebessa
Faculty Of Exact Sciences And Science Nature and life
Department Of Mathematics And computer science*

2023/2024

Dedication

To my mother and father,
to my family and my friends,
to my teachers,
to my colleagues,
to candles that burn to light up for others,
to everyone who taught me characters,
I dedicate this humble research to the Lord Almighty
to find acceptance and success.

Thanks

Thanks, and gratitude. As we take our last steps in university life, we look back on the years we've spent here, filled with learning and growth. Our distinguished professors have played a pivotal role in our journey, dedicating themselves to shaping the future generation and revitalizing our nation.

Before moving forward, we must express our profound thanks, gratitude, appreciation, and love to those who have carried the sacred torch of knowledge. They have guided us on our path to wisdom and understanding.

Special thanks and sincere appreciation to Dr. Tarek Nouioua for his support, guidance, encouragement, and all his efforts throughout this project. I would also like to thank the examiners for dedicating their time to evaluate this work.

Abstract

The integration of deep learning techniques into satellite-based flood detection systems aims to improve early warning and prevention measures. Flooding poses significant risks to human lives and infrastructure, necessitating effective monitoring and mitigation strategies.

This project leverages deep learning techniques, specifically transfer learning, to enhance satellite-based flood detection systems. By utilizing pre-trained models such as ResNet-50, VGG-16, and MobileNet-V2, we analyze satellite imagery to detect and map flood-prone areas with high precision. CNNs have the ability to automatically learn complex patterns, which significantly improves detection accuracy.

Our approach demonstrated that custom CNN architectures could achieve superior accuracy, with our custom model reaching 96.7% accuracy. Transfer learning also proved effective, with models like VGG-16 and MobileNet-V2 achieving 95% accuracy. However, the hybrid model showed high accuracy of 98% with the pretrained DenseNet201 model, demonstrating its efficacy for classification.

The application of quantum computing in flood detection systems could offer substantial advantages, including enhanced computational power and speed. These advancements could lead to more accurate and efficient flood prediction and response strategies in the future.

Abstract

L'intégration des techniques d'apprentissage profond dans les systèmes de détection des inondations par satellite vise à améliorer les mesures de prévention et d'alerte précoce. Les inondations posent des risques significatifs pour les vies humaines et les infrastructures, nécessitant des stratégies efficaces de surveillance et de mitigation.

Ce projet exploite les techniques d'apprentissage profond, spécifiquement l'apprentissage par transfert, pour améliorer les systèmes de détection des inondations par satellite. En utilisant des modèles pré-entraînés tels que ResNet-50, VGG-16 et MobileNet-V2, nous analysons les images satellites pour détecter et cartographier les zones sujettes aux inondations avec une grande précision. Les réseaux de neurones convolutifs (CNN) ont la capacité d'apprendre automatiquement des motifs complexes, ce qui améliore considérablement la précision de la détection.

Notre approche a démontré que les architectures CNN personnalisées pouvaient atteindre une précision supérieure, notre modèle personnalisé atteignant une précision de 96,7%. L'apprentissage par transfert s'est également révélé efficace, avec des modèles tels que VGG-16 et MobileNet-V2 atteignant une précision de 95%. Cependant, le modèle hybride a montré une haute précision de 98% avec le modèle DenseNet201 pré-entraîné, démontrant son efficacité pour la classification.

L'application de l'informatique quantique dans les systèmes de détection des inondations pourrait offrir des avantages substantiels, y compris une puissance de calcul et une vitesse accrues. Ces avancées pourraient conduire à des stratégies de prévision et de réponse aux inondations plus précises et efficaces à l'avenir.

Abstract

إدماج تقنيات التعلم العميق في أنظمة الكشف عن الفيضانات المعتمدة على الأقمار الصناعية يهدف إلى تحسين التدابير الإنذارية المبكرة والوقائية. الفيضانات تشكل مخاطر كبيرة على حياة البشر والبنية التحتية، مما يتطلب استراتيجيات فعالة للرصد والتخفيف.

يعتمد هذا المشروع على تقنيات التعلم العميق، وبالتحديد التعلم بالنقل، لتعزيز أنظمة الكشف عن الفيضانات المعتمدة على الأقمار الصناعية. من خلال استخدام نماذج مُدربة مسبقًا مثل ResNet-50 وVGG-16 وMobileNet-V2، نقوم بتحليل الصور الساتلية لاكتشاف ورسم خرائط المناطق المعرضة للفيضانات بدقة عالية.

تمتلك الشبكات العصبية الالتفافية (CNNs) القدرة على تعلم الأنماط المعقدة تلقائيًا، مما يحسن دقة الكشف بشكل كبير.

أثبتت مقاربتنا أن البنى المخصصة للشبكات العصبية الالتفافية يمكن أن تحقق دقة فائقة، حيث وصل النموذج المخصص لدينا إلى دقة 96.7%. كما ثبتت فعالية التعلم بالنقل، حيث حققت نماذج مثل VGG-16 وMobileNet-V2 دقة 95%. ومع ذلك، أظهر النموذج الهجين دقة عالية بلغت 98% مع النموذج المدرب مسبقًا DenseNet201، مما يثبت فعاليته في التصنيف.

قد يوفر تطبيق الحوسبة الكمومية في أنظمة الكشف عن الفيضانات مزايا كبيرة، بما في ذلك تعزيز القدرة الحاسوبية والسرعة. هذه التطورات قد تؤدي إلى استراتيجيات أكثر دقة وكفاءة للتنبؤ بالفيضانات والاستجابة لها في المستقبل.

Contents

List of Figures

List of Tables

General introduction

In recent years, the increasing frequency and severity of flooding events have highlighted the critical importance of robust early warning and prevention systems. Flooding poses significant risks to human lives, property, and ecosystems, underscoring the urgent need for advanced technologies and methodologies to mitigate its impact. Traditional flood detection methods often rely on manual interpretation of satellite imagery or simplistic algorithms, which are limited in accuracy, scalability, and timeliness, particularly in remote or densely populated regions.

Addressing these limitations, this thesis focuses on the development and application of deep learning-enhanced satellite-based flood detection systems for early warning and prevention purposes. Deep learning, a subfield of artificial intelligence, has demonstrated remarkable capabilities in pattern recognition and feature extraction from large-scale datasets, making it a promising tool for analyzing satellite imagery and identifying flood extents with unprecedented accuracy and efficiency.

The integration of AI techniques with existing methodologies is explored in this study. It investigates the potential of a deep learning-based approach for flood detection in satellite images, employing convolutional neural networks (CNNs) and transfer learning techniques to enhance disaster response and management. By leveraging CNNs' ability to automatically learn meaningful features from imagery and adapting pre-trained models through transfer learning, the proposed methodology aims to revolutionize flood detection systems and reduce vulnerability in flood-prone areas. The work contributes to advancing flood monitoring systems by combining the power of CNNs and transfer learning to develop an accurate and robust flood detection model. [1]

Chapter 1

Theoretical Study about Flood detection for early Warning and prevention

1.1 Introduction

Flooding events demand quick response , posing a big challenge and threat to communities worldwide , Flood detection plays a crucial role in early warning systems and prevention strategies for effective disaster response and management, enabling timely warnings, evacuation planning, and resource allocation.

While traditional flood detection methods face challenges due to their limited efficacy in real-time monitoring and detection capabilities compared to advanced techniques based on deep learning. These limitations restrict timely response and early warning systems for flood events. By leveraging advanced technologies like deep learning and satellite imagery, we can enhance our capabilities in detecting floods accurately and efficiently, ultimately saving lives and reducing the devastating effects of floods.

1.2 Overview about Flood Detection in Satellite Images

Floods are a natural disaster causing a lot of damage and loss of life , using satellite images can provide wide coverage and high resolution for monitoring large areas and can help us to know flood or non-flood pattern using Deep learning techniques, especially Convolutional Neural Networks (CNNs) prove to be highly effective ,in analyzing images CNNs have the capability to recognize flooded regions by processing sets of satellite images.[2]

1.2.1 Limitations of traditional flood detection methods

Traditional methods for flood detection in satellite imagery face several challenges that limit their effectiveness.

1.Spectral Similarity: Traditional methods often rely on analyzing the spectral reflectance properties of water bodies in satellite images. However, flooded areas can have spectral signatures similar to other features like shadows, snow, or wet soil. This similarity can lead to false positives (areas identified as flooded when they are not) and false negatives (flooded areas missed by the method).

2.Complexity of Flood Patterns: Floods can exhibit a wide range of appearances in satellite images. Factors like water depth, vegetation cover, debris, and sediment load can significantly alter the spectral signature of flooded areas. Traditional methods may struggle to capture this variability and accurately identify floods across diverse scenarios. Spatial Resolution Limitations: The spatial resolution of some satellite sensors may not be fine-grained enough to capture smaller floods or detailed flood boundaries. This can lead to underestimation of the flood extent or missing critical information for early warning systems.

3.Data Dependency: Traditional methods may require specific weather data or pre-flood baseline images for comparison. This can limit their applicability in situations where such data is unavailable or unreliable.

4.Computational Cost: Some traditional methods can be computationally expensive to implement, especially when dealing with large datasets of satellite imagery. This can hinder real-time monitoring capabilities[3]

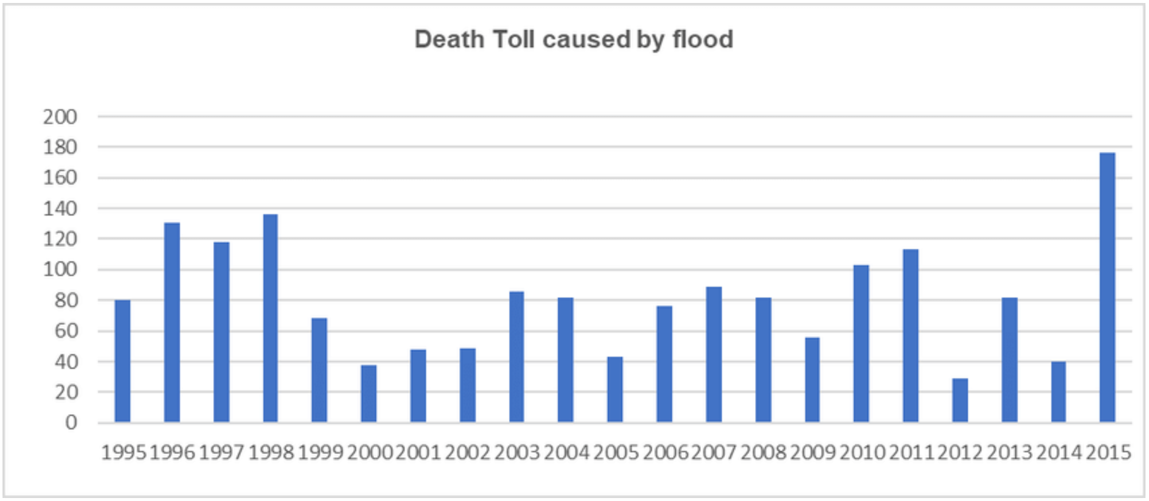


Figure 1.1: this figure represents flood fatality statistics in US by The U.S. Natural Hazard Statistics

[4]

1.2.2 Importance of flood detection and early warning systems

As we can see in figure 1.2 flood is a huge risk that poses threats to ecosystems worldwide.

Why early warning systems?

1.Risk Reduction and Preparedness : Early warning systems help reduce flood-related risks by providing timely information about possible flood events. This allows

Figure 1.2: this figure represents flood risk in Malaysia
[?]

authorities and individuals to take necessary precautions and evacuate areas at risk and minimize loss of life and property.

2.Public Safety: The primary purpose of flood detection and early warning systems is to ensure public safety. These systems can help save lives and prevent flood damage by providing accurate and timely information about flood risk.

3. Environmental protection: Flood warning systems also work to protect the environment by minimizing pollution and pollution from floods. These systems support efforts to protect natural ecosystems by alerting authorities to potential environmental hazards.

4. Economic benefits: Early warning systems can lead to significant cost savings by reducing flood damage. These systems contribute to economic stability in flood-prone areas by minimizing the impact on commerce, agriculture and public services. [5]

1.2.3 Importance of incorporating deep learning techniques

Incorporating deep learning techniques, such as Convolutional Neural Networks (CNNs) and transfer learning, for flood detection offers significant advantages in terms of accuracy, robustness, and real-time applicability. By leveraging pre-trained deep learning models like ResNet50 and VGG19 to build a new performance model, we can improve the generalization capability of the system.[?]

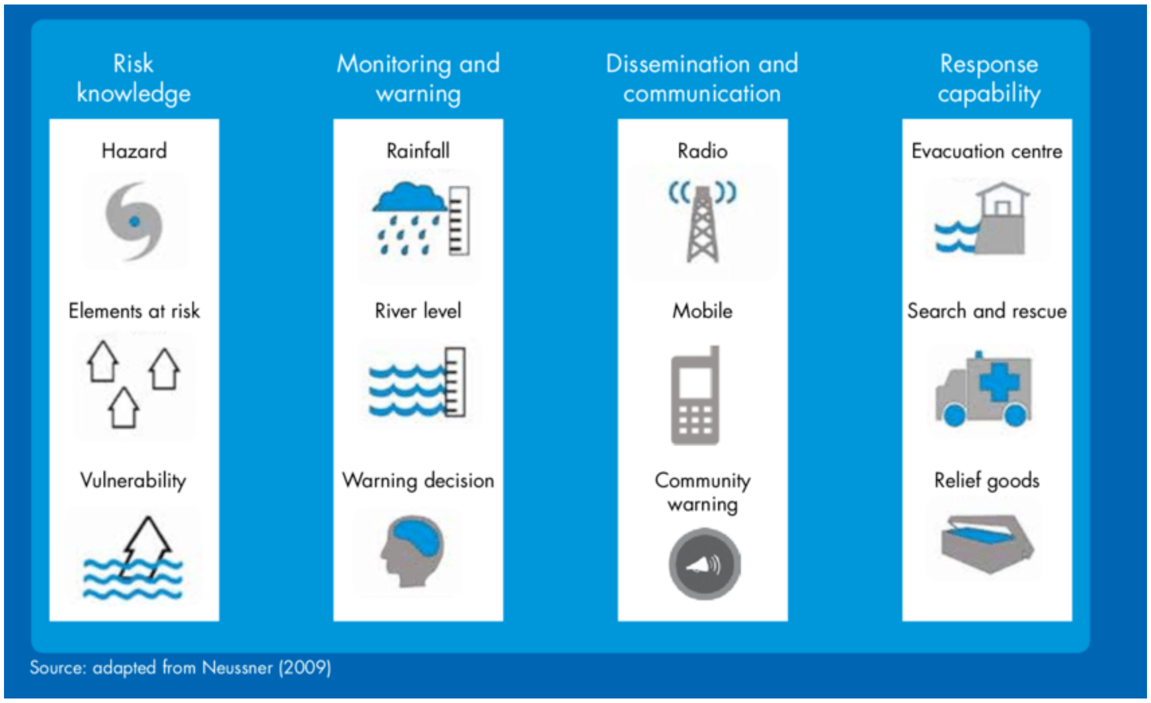


Figure 1.3: This figure represents the components of a flood early warning system [?]

Transfer Learning Definition :

Transfer learning is a powerful technique used in deep learning that improves the learning of a new task By leveraging the ability to reuse existing models and their knowledge of new problems, it has facilitated the training of deep neural networks, even when data is limited.[6]

The Powerful Partnership: CNNs and Transfer Learning in Deep Learning:

Training a CNN from scratch on a new task can be expensive and requires a huge dataset. Transfer learning enables us to adapt pre-trained CNN models, such as VGG19 and ResNet50, trained on large datasets like ImageNet, transfer learning is our main approach.

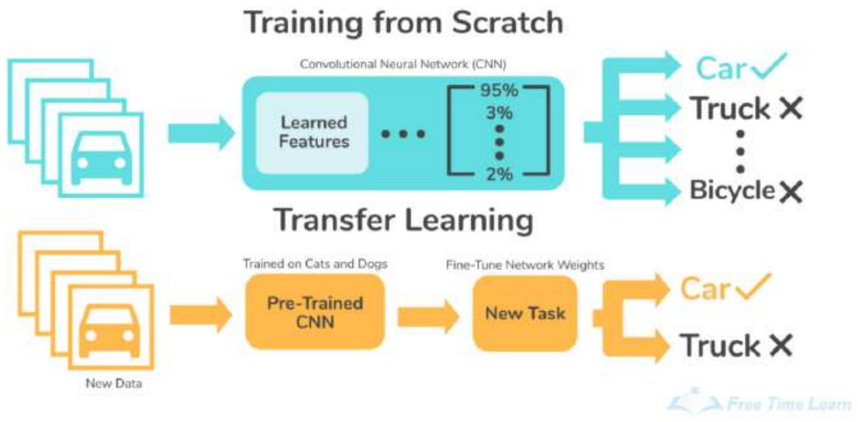


Figure 1.4: This figure represents transfer learning and training from scratch [7]

1.3 Conclusion :

In this chapter, we gave an overview of flood detection for early warning and prevention. We also discussed the challenges involved and introduced how deep learning techniques can help address them. We'll explore these techniques further in detail.

Chapter 2

Artificial Intelligence Technics

2.1 Introduction

In this chapter, we're going to explore a bunch of different AI techniques. We'll look at how each one helps AI systems get better and work more effectively. The idea of AI has been around for a while, dating back to the 1940s and 50s. There were some important research projects during this time, like figuring out how to make networks of connected neurons do computations. Then in 1956, there was a big workshop at Dartmouth College where AI really started to take off. John McCarthy, who was a big deal in AI, organized the workshop.[8]

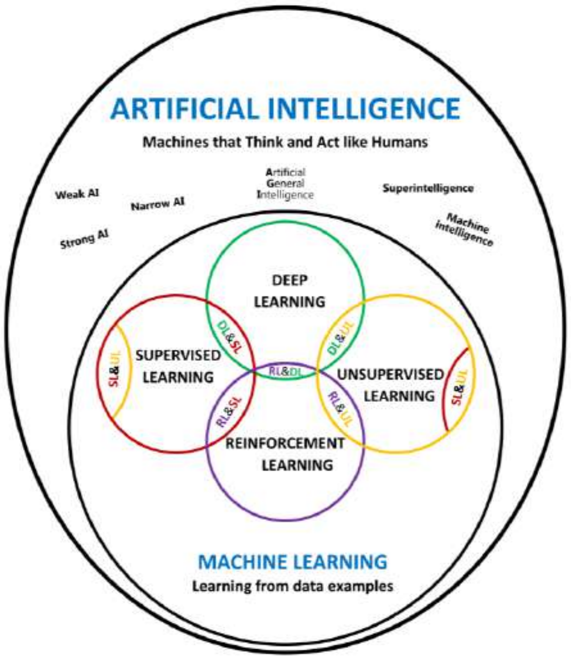


Figure 2.1: Ai Realm and the implied Techniques
[9]

2.2 Definition

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to mimic human behavior and cognitive functions such as learning, problem solving, perception, reasoning, and decision making. Artificial intelligence encompasses a wide range of technologies and approaches, including machine learning, deep learning, natural language processing, computer vision, robotics, and expert systems. The goal of artificial intelligence is to create systems that can perform tasks that normally require human intelligence, such as automating processes, increasing productivity, forecasting, healthcare, finance, transportation, education and training. [10]

2.3 Machine learning

2.3.1 Definition :

Machine Learning (ML) is a subset of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn patterns and make predictions or decisions from data without being explicitly programmed to perform the task. In essence, machine learning algorithms allow computers to learn from experience, adapt to new data, and improve their performance over time.[11]

2.3.2 Types of ML :

There is three general Machine Learning types of which we can mention.

Supervised Learning :

Supervised learning is a type of machine learning where an algorithm is trained on labeled data to learn the relationship between inputs and outputs. It is used for tasks like classification (e.g., detecting spam emails) and regression (e.g., predicting house prices). The algorithm uses this learned relationship to make predictions on new, unseen data.[12]

Unsupervised Learning :

Unsupervised learning is the case where we fit the model without known outputs. Our goal does not involve to predict any labels here. Rather we expect our model to enlighten us by finding unseen patterns within the data set. This might be in the way of grouping the data in various ways to our pleasure. Since we don't have any labels for model output, we need to further analyze the output results so that we can make use of it.[12]

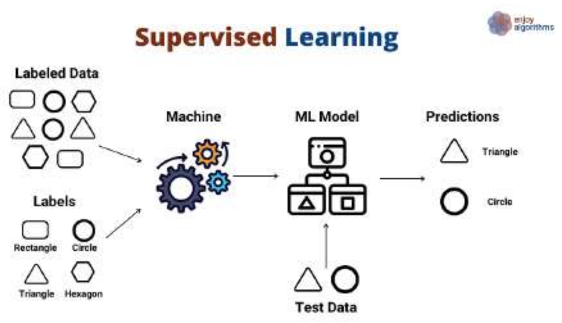


Figure 2.2: Supervised Machine Learning schema
[13]

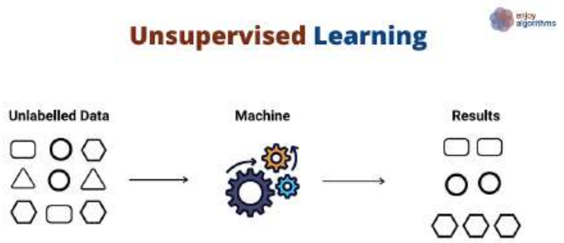


Figure 2.3: Unsupervised Machine Learning schema
[13]

Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning focused on decision-making. Its objective is to learn the best behavior in a particular environment to obtain maximum reward. RL accomplishes this by interacting with the environment and observing its responses, similar to how a child explores their surroundings to learn the actions that lead to a desired outcome. In contrast to supervised learning, RL does not rely on pre-labeled data. Instead, the algorithm must independently discover the sequence of actions that lead to maximum reward. This process involves a trial-and-error approach, where the quality of actions is measured not only by the immediate reward they return but also by the potential delayed reward they may generate. Reinforcement learning is a powerful algorithm that can learn to take the right actions in an unfamiliar environment without the guidance of a supervisor. It has been successfully applied in various fields, such as robotics, game playing, and autonomous vehicle navigation [12]

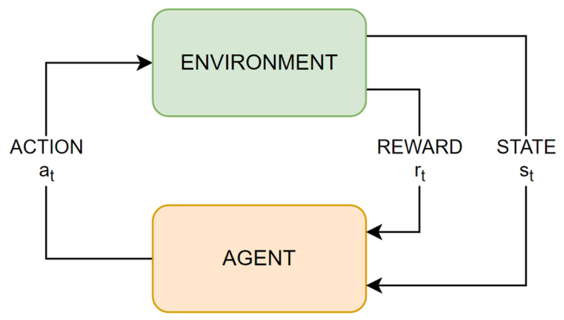


Figure 2.4: Reinforcement Learning schema

2.4 Artificial Neural Networks

Computer architecture known as Artificial Neural Networks (ANNs) is modeled after the neural networks in the human brain. Artificial neurons in ANNs perform brain-like functions as do biological neurons. Each neuron is linked to its neighbors neurons by edges that can transmit messages to other cells. The strength of the connections between neurons increases as the network learns through a process known as training. Each neuron's output is dictated by a non-linear function of the sum of its inputs. Applications for ANNs include audio and picture recognition, natural language processing, and predictive analytics. [14]

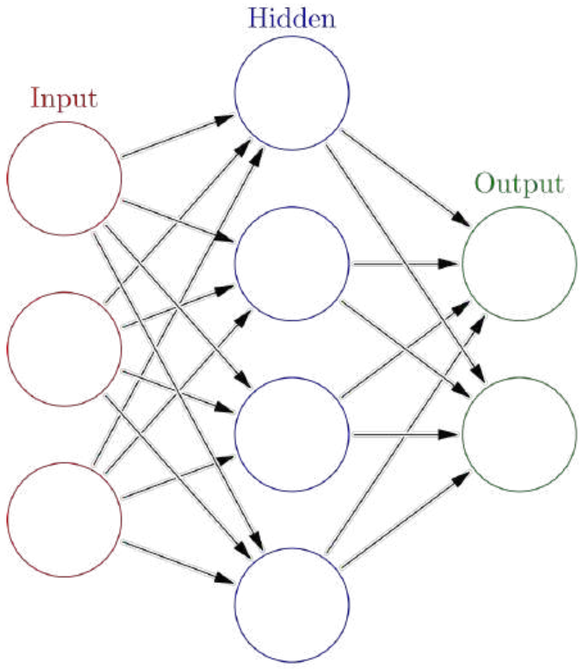


Figure 2.5: Artificial Neural Network Structure
[14]

2.5 Multilayer Perceptron

A multi-layer perceptron (MLP) is a type of neural network comprising input, hidden, and output layers. Input signals are processed by the hidden layers, which perform computational operations, before being propagated to the output layer for tasks like prediction or classification. MLPs utilize backpropagation to train their neurons, adjusting parameters to minimize prediction errors. With the capability to approximate any continuous function, MLPs excel in tasks such as pattern categorization, recognition, prediction, and approximation due to their adaptability and capacity to learn complex data relationships.[15]

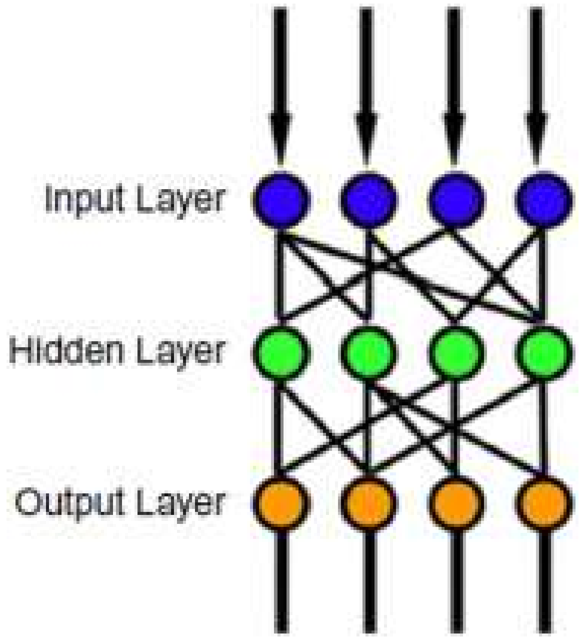


Figure 2.6: MultiLayer Perceptron schema
[16]

2.6 Activation functions

In a neural network, the activation function plays a role similar to how our brains decide whether to fire a neuron based on incoming signals. It's like a filter that helps the neuron decide if the incoming information is important for making predictions. By performing basic math operations, it decides whether to "activate" or "not activate" the neuron. Its main job is to crunch the numbers and produce an output based on the inputs it receives. There are various types of activation functions, and we'll explore a few of them below.[17]

2.6.1 Binary step function:

The Binary Step Function stands as the most basic activation function, easily implemented in Python with straightforward if-else statements.
It's commonly employed in binary classifiers, where decisions are simplified into two categories.
However, it falls short in multiclass classification tasks since it can only distinguish

between two classes, limiting its applicability in scenarios where multiple classes need to be identified.[17]

2.6.2 Linear Activation Function :

The linear activation function is directly proportional to the input. The main drawback of the binary step function was that it had zero gradient because there is no component of x in binary step function. In order to remove that, linear function can be used. It ca be defined as: F(x) = ax The value of variable a can be any constant value chosen by the user[18].

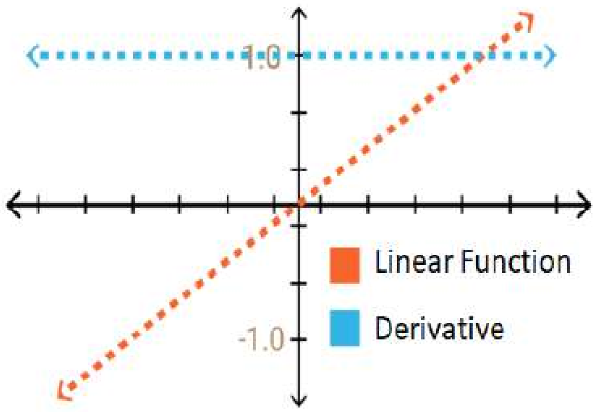


Figure 2.7: Linear Activation Function
[17]

2.6.3 Relu Function :

ReLU stands for rectified liner unit and is a non-linear activation function which is widely used in neural network. The upper hand of using ReLU function is that all the neurons are not activated at the same time. This implies that a neuron will be deactivated only when the output of linear transformation is zero[18]

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

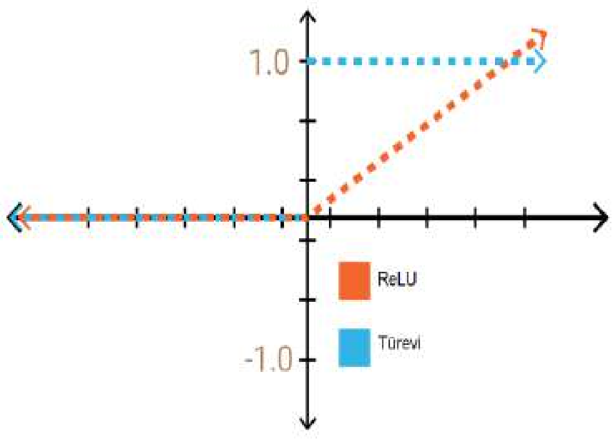


Figure 2.8: Rectified Linear Unit Activation Function
[19]

2.6.4 Sigmoid function

The sigmoid function, also called the logistic function, is an activation function that returns an output ranging from 0 to 1, which is useful for normalizing the output of each neuron. However, the sigmoid function can cause the neural network to stop learning when input values are extremely high or low, resulting in the vanishing gradient problem.[17]

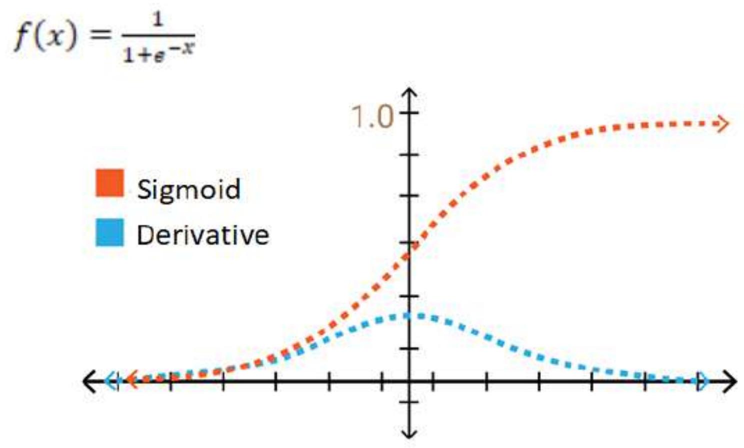


Figure 2.9: Sigmoid Function
[17]

2.7 Deep learning Definition

Deep learning is a subset of machine learning that uses neural networks with multiple layers to analyze complex data. The architecture of deep learning models is based on artificial neurons that aim to mimic the functioning of the human brain. By using multiple neural layers in deep learning models, you can extract and learn high-level features from large data sets. This approach can be applied in various fields such as image recognition, natural language processing and predictive modeling.[20]

2.8 Difference between machine learning and deep learning:

2.8.1 Machine Learning (ML):

Requires manual feature extraction. Uses simpler models with fewer parameters. Less data hungry and computationally intensive compared to deep learning. Suitable for tasks with smaller datasets and when interpretability of the model is important.[21]

2.8.2 Deep Learning (DL):

Learns feature representations directly from raw data. Employs complex neural network architectures with millions or billions of parameters. Requires large amounts of labeled data and significant computational resources. Excels in tasks with large and complex datasets, such as image recognition, natural language processing, and speech recognition.

Overall, machine learning is more suitable for simpler tasks or when interpretability is crucial, while deep learning shines in complex tasks where large datasets and computational power are available, and feature engineering is challenging.[21]

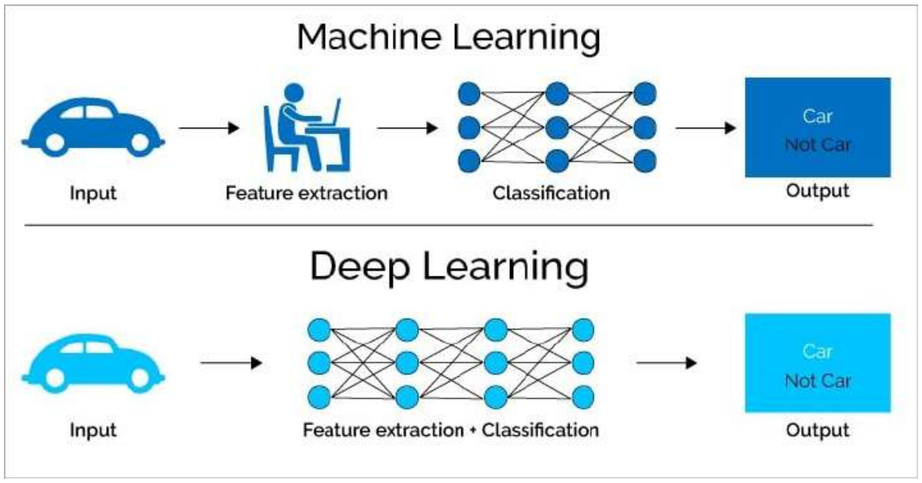


Figure 2.10: Difference between ML and DL
[21]

2.9 Major Deep Learning Types

There are three main types of deep learning, which included as:
- MLP: which stands for Multilayer Perceptron
- RNN: stands for Recurrent Neural Networks
- CNN: stands for Convolutional Neural Networks

2.9.1 Convolutional Neural Networks (CNN):

A deep learning approach used for image identification and classification is a convolutional neural network (CNN). From input photos, it is intended In order to learn spatial hierarchies of features in an automatic and adaptive manner, without requiring manual feature extraction. The key idea behind CNNs is to use convolution operations on the input image, which allows the network to learn filters that capture important patterns and features in the image. By stacking multiple convolutional layers and other types of layers, such as pooling and activation layers, CNNs can learn increasingly complex features and patterns, leading to highly accurate image recognition and classification.[4]

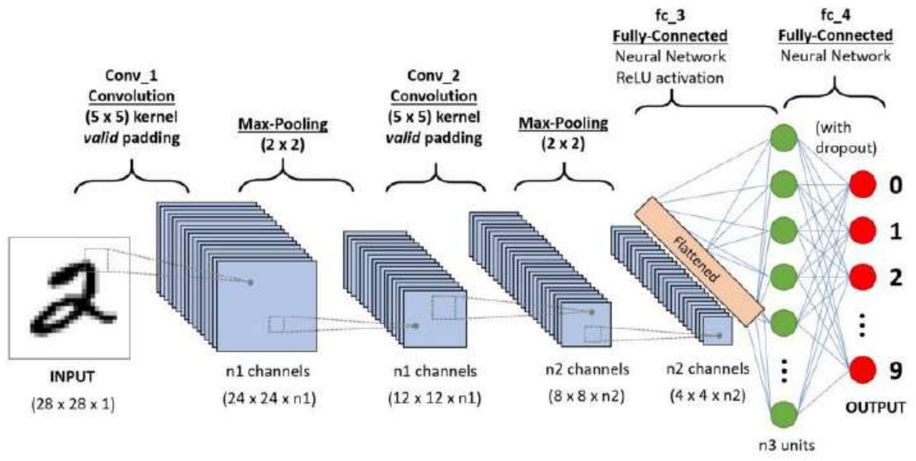


Figure 2.11: Convolutional Neural Networks schema
[22]

2.9.2 CNN components :

the components enable CNNs to learn to extract hierarchical features from input data, making them highly effective for tasks such as image classification, object detection, and semantic segmentation.

1. Convolutional Layer: This layer applies filters (kernels) to the input image to detect specific patterns. Each filter produces an activation map that represents the presence or absence of particular features.

2. Pooling Layer: After each convolutional layer, a pooling layer is typically added to reduce the spatial dimension of the activation map. This helps to decrease the model's complexity while preserving important features.

3. ReLU Activation: The ReLU (Rectified Linear Unit) activation function is often used after the convolutional layer and before the pooling layer. It introduces non-linearity by replacing negative values with zero, allowing the network to better learn non-linear relationships in the data.

4. Fully-Connected Layer: After several convolutional, pooling, and ReLU layers, the final layers of the CNN are typically fully-connected layers. These layers aggregate the information learned by the previous layers and produce an output vector that is then used for classification.

2.10 Tensorflow Definition

TensorFlow is an open-source platform and framework developed by Google researchers , includes libraries and tools based on Python and Java — designed with the objective of training machine learning and deep learning models on data.[23]

2.10.1 Tensorflow application :

There are numerous applications of TensorFlow, which contribute significantly to its widespread popularity across diverse domains. These applications include : image recognition and classification, natural language processing (NLP), speech recognition and synthesis, reinforcement learning, as well as transfer learning and fine-tuning.[23]

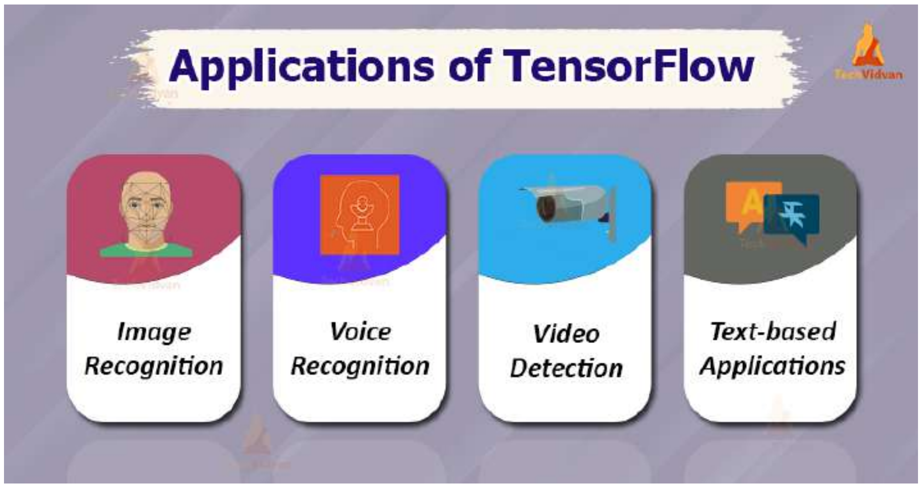
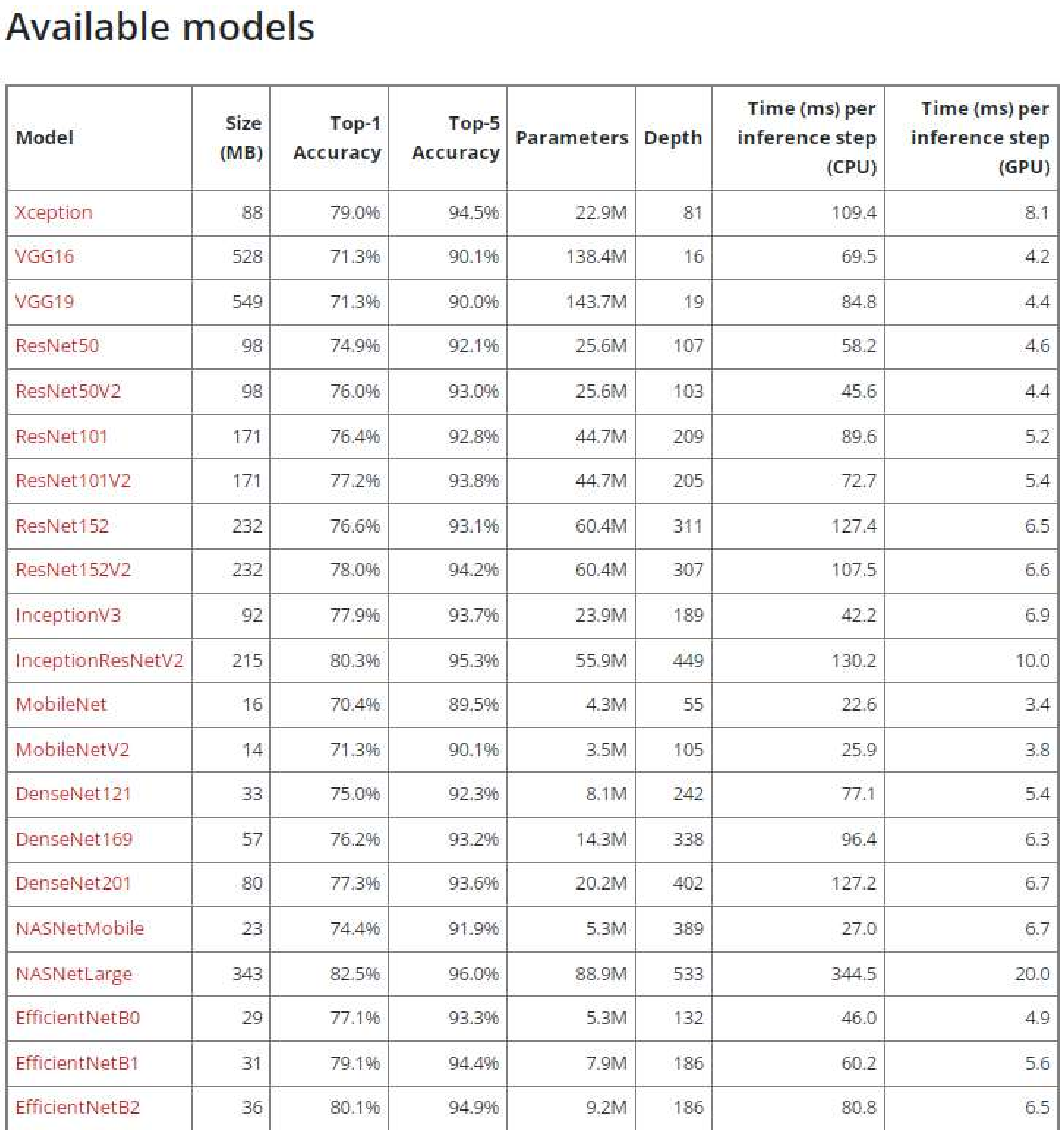


Figure 2.12: Some applications of tensorflow
[24]

2.10.2 Keras Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning. the official Table from keras website below contains a few:



Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5

Figure 2.13: Keras Applications
[25]

VGGnet:

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The "deep" refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers. The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures.[26]

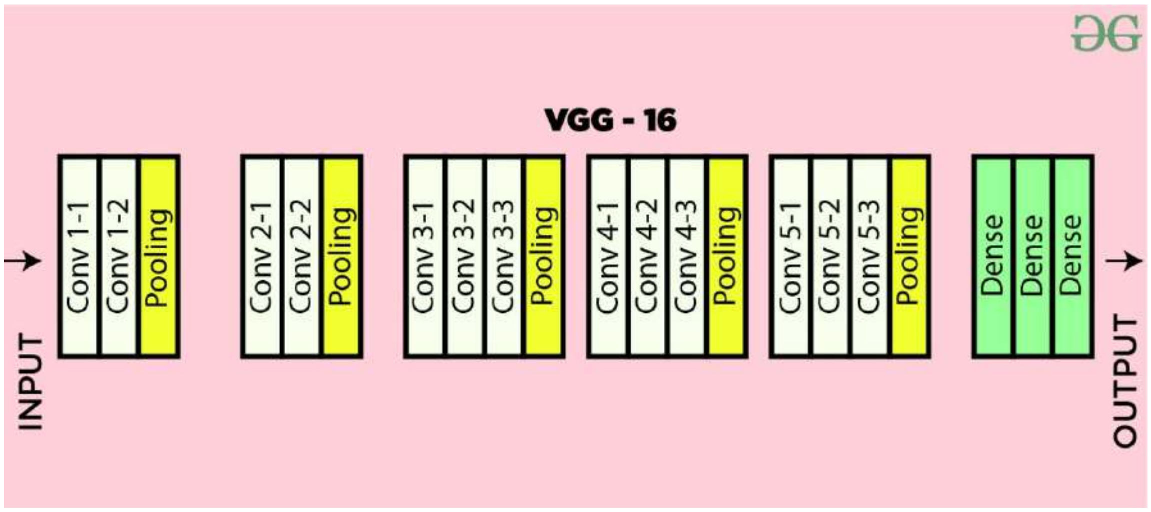


Figure 2.14: VGG-16-architecture
[27]

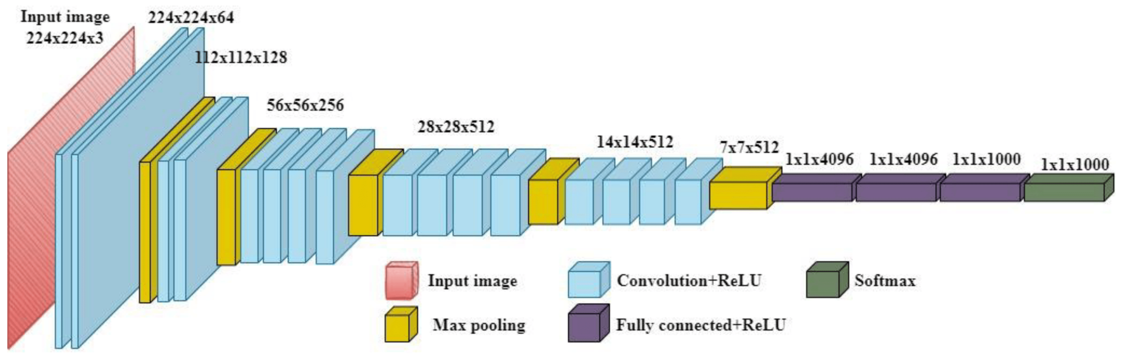


Figure 2.15: VGG-19-architecture
[28]

ResNet

ResNet, short for Residual Networks, is a type of convolutional neural network (CNN) architecture that was introduced by Microsoft Research in 2015. It was designed to address the problem of vanishing gradients and degradation in the training of deep neural networks with a very large number of layers. The key innovation of ResNet is the use of residual connections, also known as skip connections or shortcut connections, which allow information to bypass certain layers in the network. Instead of trying to learn the desired mapping directly, ResNet aims to learn the residual mapping—the difference between the desired mapping and the identity mapping. By incorporating residual connections, deeper networks can be trained more effectively without suffering from vanishing gradients or degradation in performance. - ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. These variants differ primarily in the number of layers and the overall depth of the network.[29]

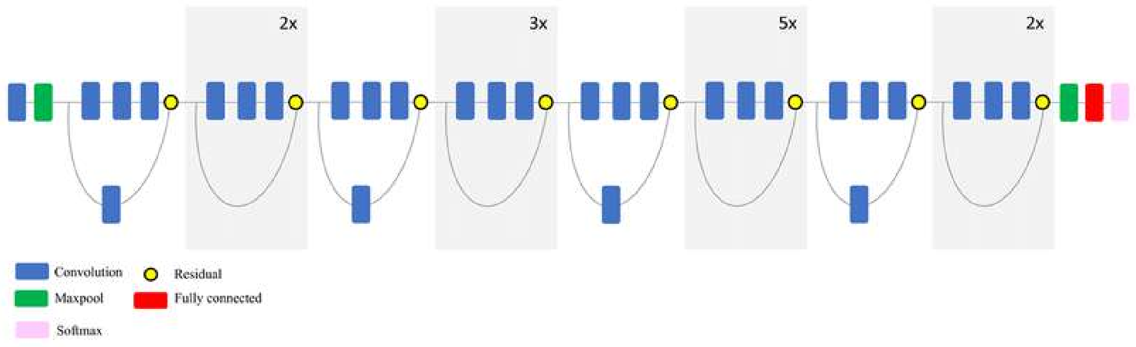


Figure 2.16: ResNet Model Schema

MobileNetV2

MobileNet-v2 is a deep convolutional neural network with 53 layers. It’s well-suited for image classification tasks and it pre-trained on a large dataset over a million images from the ImageNet database. it has been trained to classify images into 1000 object categories, like cats, mice, and pencils to various animals and beyond.[30]

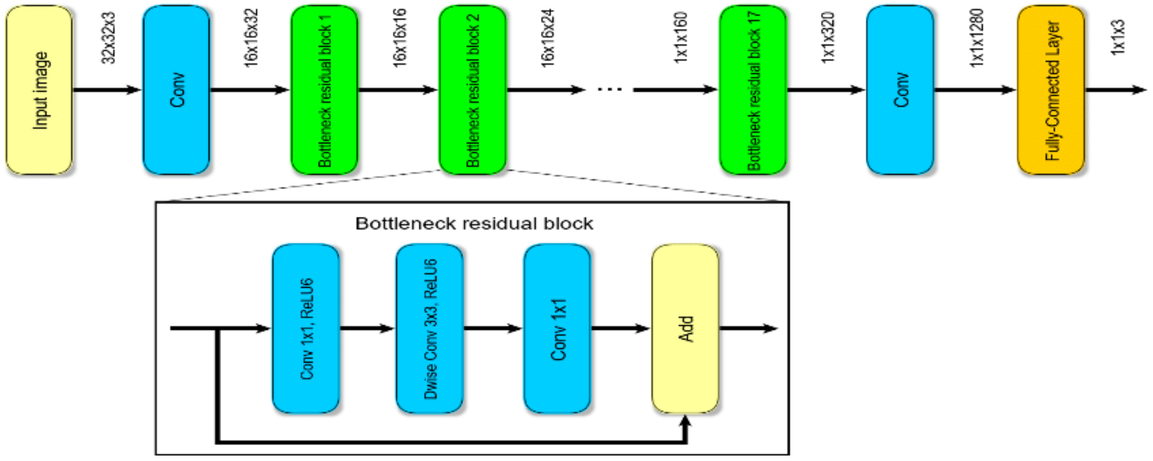


Figure 2.17: MobileNetV2 Schema
[30]

2.11 Conclusion:

In this chapter, we have discussed various artificial intelligence techniques and explored several of its technologies.

Chapter 3

STATE OF THE ART

3.1 Introduction

Deep learning techniques, coupled with satellite imagery, have revolutionized flood detection for early warning and prevention. Consequently, researchers aim to enhance the accuracy and timeliness of flood detection systems. In this chapter, we will explore recent studies related to our field, with a focus on examining transfer learning techniques. Additionally, we will integrate quantum computing techniques ,we will discuss two out of seven articles.

3.2 Related works

3.2.1 Nouioua Tarek (2023) : :

This article is authored by Dr. Nouioua tarek and represents the launch of my project, the goal is flood detection in satellite images using deep learning incorporating transfer learning, this article aims to develop a robust model capable of accurately identifying flooded and non-flooded regions. By leveraging convolutional neural networks (CNNs) and transfer learning techniques, This process involves utilizing pre-trained models specifically, ResNet50, InceptionV3, and VGG16—to generate predictions on a dataset that contains 102 images in total , 82 images for the training process while 20 are reserved for validation. the InceptionV3 model shows better performance for 'non-flood images' but needs improvement for 'flood images.' The VGG16 model has moderate accuracy but also has room for enhancement in precision and recall, while Resnet50 shows good performance and high precision, recall, and F1-score, the model is more accurate.[1]

3.2.2 Danielle Dias and Ulisses Dias(October 2018):

In "Flood detection from social multimedia and satellite images," the authors Danielle Dias and Ulisses Dias from the University of Campinas, Brazil, explore the use of deep convolutional neural networks (CNNs) pre-trained on ImageNet, along with transfer learning mechanisms, the authors selected 10 pre-trained CNN that were trained on imageNet dataset they used transfer learning in feature extraction, model prediction, and ensemble extraction for enhancing the efficiency and effectiveness of their flood detection models using a dataset which contains two categories, the first dataset contains images from social media and the second contains high-resolution satellite images detecting areas affected by flood. using pre-trained CNN architectures as feature extractors, it benefits from the learned features without training the model from scratch, so the authors instead of directly using the output of these pre-trained models for classification replaced the final part of the model with ANN (Artificial Neural Network) to make predictions.

let's take a look at the results, In the social media task, the ensemble of pre-trained models achieved an F1-Score of 64.81%, and the best individual model is (ResNet50) with an F1-Score of 62.93%.
In the satellite imagery task, the ensemble achieved an F1-Score of 71.72, which was slightly below the F1-Score of the best individual model (DenseNet121) at 73.27%. [6]

3.2.3 S. V. Georgakopoulos , K. Kottari1, K. Delibasis1, V. P. Plagianakos , I. Maglogiannis (2018).

this article explores the application of CNN in classifying skin images, particularly in dermoscopy images that are used for diagnosing skin. the new idea is to enhance CNN's performance on classifying skin images by adding techniques called mid-level computer vision filters to the inputs of CNN, the filters help to extract special features in the images while having a small dataset it should integrate transfer learning techniques, which adjust a pre-trained CNN model on the specific task while using the available data. the results of the study demonstrate the improvement of classification accuracy when the dataset is not large, the methodology also simplifies the training process by making the areas analyzed by each neuron in the CNNs. [31]

3.2.4 Zhongling Huang , Zongxu Pan and Bin Lei(2017).

This article focuses on deep learning and its applications in image classification, particularly utilizing transfer learning techniques and training deep CNNs for SAR target recognition. This involves using a large dataset (unlabeled dataset) as the source domain and the labeled dataset (MSTAR dataset) as the target domain. The goal is to transfer features learned from the unlabeled SAR scene data (source domain) to enhance performance on the target SAR classification task. The authors propose a novel architecture for SAR target recognition, consisting of two main pathways: a classification pathway and a reconstruction pathway. Each pathway is interconnected through a feedback bypass connection. In the training stages, the authors utilized three stages (unsupervised pre-training, transfer learning by fine-tuning and Joint fine-tuning) , They observed that the transfer learning approach outperforms the baseline CNN trained only on the MSTAR dataset, achieving an accuracy of 99.09% on the 10-class SAR target recognition task. This indicates better performance, and using transfer learning demonstrated robustness and a reduction in the size of the training dataset. [32]

3.2.5 Tulasi Krishna, Sajja Kalluri, Hemantha kumar. (2019).

The article examines the application of CNNs for image classification tasks. It explores the CNN architecture, discussing its components such as the input layer, convolution layer, fully connected layer, and softmax layer. Additionally, it discusses factors influencing image classification accuracies, such as the number of epochs, dataset size, GPU-based systems, and batch size. It also provides various deep learning architectures, including pre-trained models like LeNet, AlexNet, VGG, and ResNet. The study presents experimental results comparing the performance of deep learning models with traditional machine learning algorithms on datasets such as CIFAR-10, ImageNet, and MNIST, among others.[33]

CNN Architecture	**Year**	**Developed by**	**No. of Parameters**
LeNet[4]	1998	Yann LeCun et al.	60,000
AlexNet[5]	2012	Alex K et al.	62.3 million
VGGNet[6]	2014	Simonyan, Zisserman	138 million
GoogleNet[7, 8]	2014	Google	4 million
ResNet[9]	2015	Kaiming He	25 million

Table 3.1: CNN Architectures with number of parameters
[33]

3.2.6 Tarek, Nouioua Hafid, Belbachir. (2021).

This study explores the power of quantum computers and their development algorithms on information systems security, Quantum computers have the potential to perform calculations faster than classic computers, and the authors highlight the capability of quantum computers to break security protocols such as RSA, commonly used for data transmission, it seems secure but it is not that would be in risk, must be post-quantum cryptography in the future. Classic computers have a bit, quantum has a qubit which means it can have two values 0 and 1 at the same time, it can be in a superposition of state :

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in C$ with $|\alpha|^2 + |\beta|^2 = 1$

Quantum instructions are specified for quantum processors and necessitate special RAM known as QRAM. Additionally, in qubit information, one qubit represents two bits of information which means the number of possible information states increases exponentially (2 raised to the power of n) while quantum computing poses challenges related to hardware and software, it also offers the optimization of complex systems, enhancing data security, and provides researchers with the potential to unlock this transformative technology. [34]

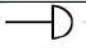
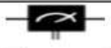
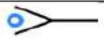
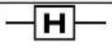
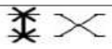
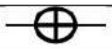
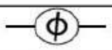
QP-Instruction symbol	Program Text	Quantum Platform
	qc.read(t)	QCEngine
	qc.measure(t, t_had)	Qiskit
	qc.write(t,val) (qc.reset(t): write 0)	QCEngine (Qiskit)
	qc.had(t) (qc.h(t))	QCEngine (Qiskit)
	qc.exchange(t1—t2) (qc.swap(x,y))	QCEngine (Qiskit)
	qc.cnot(t) (qc.x(t))	QCEngine (Qiskit)
	qc.phase(angle,c)	QCEngine

Figure 3.1: QUANTUM INSTRUCTIONS AND THEIR SYMBOLS

3.2.7 Tarek, Nouioua Belbachir, Ahmed. (2022).

This article, authored by Tarek Nouioua and Ahmed Hafid Belbachir, explores the potential of quantum computers in accelerating and strengthening the security of information systems. The authors aims the power of quantum over classical computers in image processing, highlighting its ability to provide exponential speedup for image processing algorithms such as edge detection while reducing memory requirements. For example, a 256x256 image can be represented using just 17 qubits in quantum systems, compared to gigabits of memory in classical systems.

quantum computing can enhance image security through quantum encryption and stenography techniques. The article introduces a quantum edge detection algorithm that operates with quantum image representation, such as the Novel Enhanced Quantum Representation (NEQR). This algorithm offers exponential speedup with a time complexity of O(1), enabling the processing of large high-resolution images, including 3D medical imaging, which is prohibitively expensive for classical computers.

However, quantum computing also poses a significant challenge to classical information security, as it has the potential to break traditional cryptographic methods.

so , post-quantum cryptography offers a solution by developing new cryptographic systems based on quantum physics principles, which are resilient to quantum attacks. It is highlighted in the research the transformative capability of quantum computing across various fields and ensure the need for continued research and development in this rapidly evolving domain. [35]

3.3 Conclusion :

In this chapter, we have examined seven related works. The first five studies focus on utilizing Convolutional Neural Networks (CNNs) in combination with transfer learning techniques. These studies have demonstrated the effectiveness of such approaches in creating accurate and efficient models for classification tasks. The last two studies delve into the realm of quantum computing. Following data augmentation, we aim to incorporate quantum computing elements into our model which becomes hybrid model to observe its potential in accelerating image processing and enhancing the efficiency of building robust flood detection models, in the end we will do a comparison and analysis between the hybrid task and the classical task to evaluate their performances.

Chapter 4

Implementation and Results

4.1 Introduction :

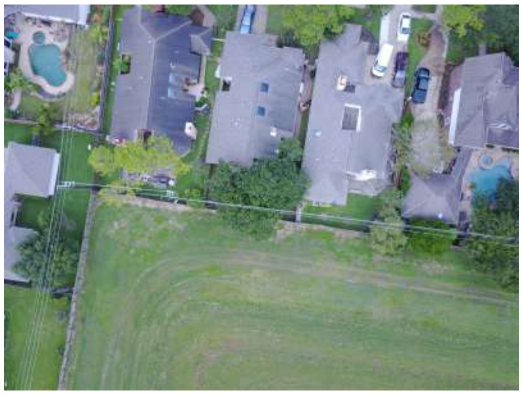
In this chapter, we delve into the practical aspect of our project. We are going to develop a system to predict flood and non-flood images using satellite imagery. There are two tasks: the first involves building a classical model, while the second entails creating a hybrid model. Finally, we will examine the results and conduct a comparison.

4.2 Dataset definition , data preparation :

A dataset is a structured collection of data, typically organized in a way that facilitates analysis and processing , in our project we are using a dataset that contains 102 satellite images belonging to two classes: flooded and non-flooded. The dataset is divided into two subsets: training and validation sets.

Training Set: This subset contains 80 images, which are used to train our deep learning model. During training, the model learns from these images by adjusting its parameters to minimize the difference between its predictions and the actual labels.

Validation Set: This subset contains 20 images, which are used to evaluate the performance of our trained model. The model's performance metrics, such as accuracy, loss, precision, recall, and F1 score, are calculated based on its predictions on these validation images. The validation set helps to assess how well the model generalizes to new, unseen data.


(a) this figure represents non-flood image

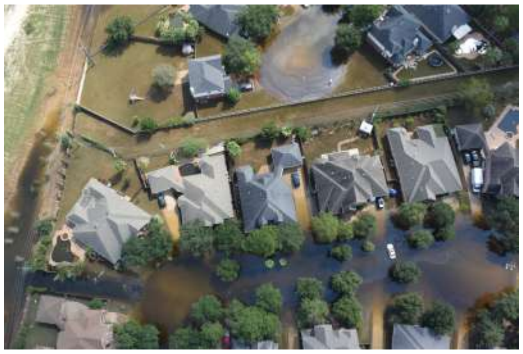
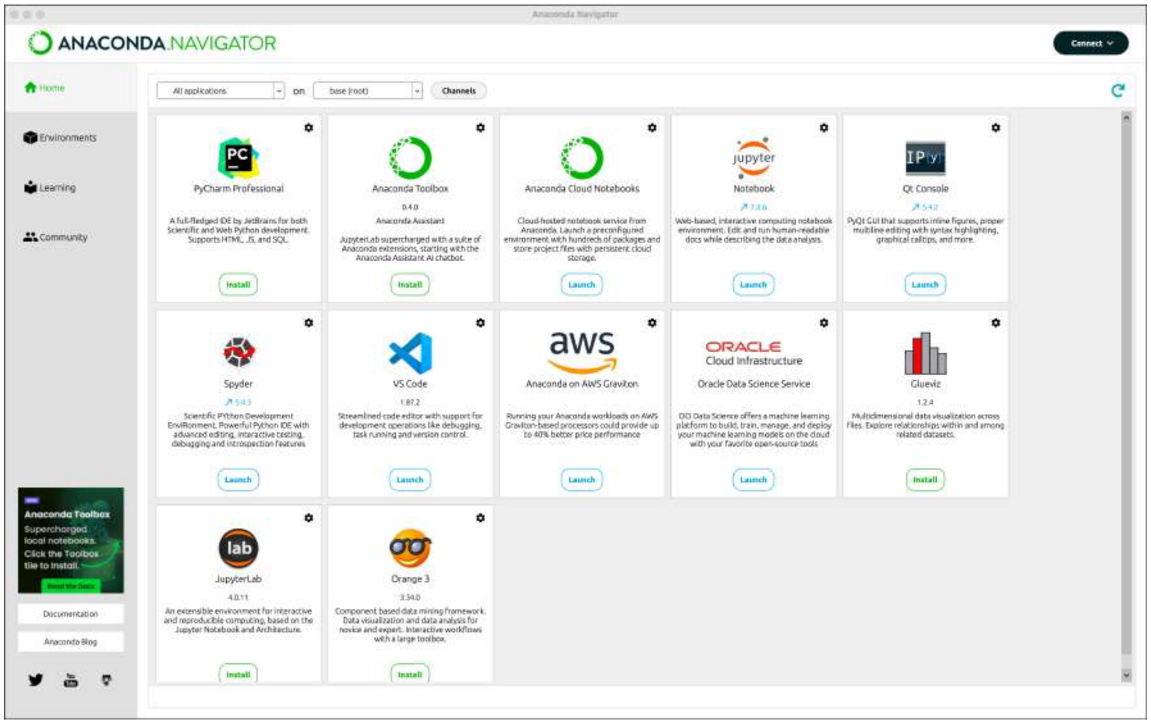

(b) this figure represents flood image

Figure 4.1: Flood and non-flood images from My dataset

4.3 Work environment and Tools :

4.3.1 Anaconda

What is Anaconda? Anaconda is a free software package designed to help you with research and scientific projects. By installing Anaconda, you get access to various environments where you can code in Python or R. These environments, known as integrated development environments (IDEs), are platforms or apps that make writing and developing code much easier, similar to how Microsoft Word or Google Docs make writing documents simpler.[36]



PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

Anaconda Toolbox
0.4.0
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.

Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

Notebook
↗ 7.0.6
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
↗ 5.4.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
↗ 5.4.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.87.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Anaconda on AWS Graviton
Running your Anaconda workloads on AWS Graviton-based processors could provide up to 40% better price performance

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

JupyterLab
4.0.11
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

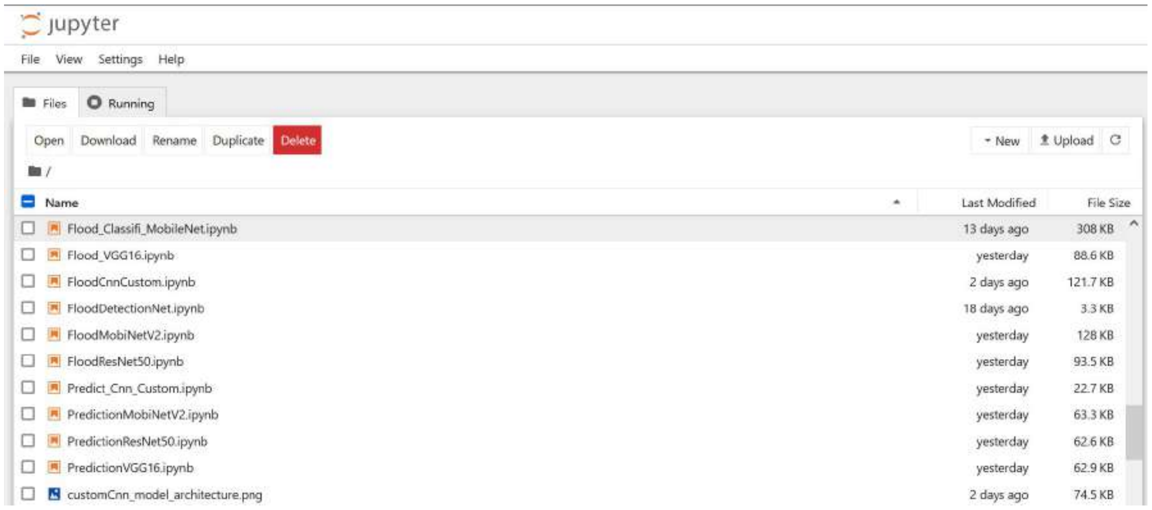
Orange 3
3.34.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Anaconda Toolbox
Supercharged local notebooks. Click the Toolbox tile to install.
Read the Docs

Figure 4.2: This figure represents the interface of anaconda

4.3.2 Jupyter notebook

Jupyter Notebook is a web-based tool that you can access through your browser. It allows you to run blocks of code separately, making it very flexible and perfect for experimenting. You can combine different types of content, such as code, visualizations, equations, and text, all in one place. This makes it easy to create, share, and present your work in an organized and visually appealing way. Since it's web-based, sharing your notebooks with others is simple, making it great for collaboration.[36]



Name	Last Modified	File Size
Flood_Classifi_MobileNet.ipynb	13 days ago	308 KB
Flood_VGG16.ipynb	yesterday	88.6 KB
FloodCnnCustom.ipynb	2 days ago	121.7 KB
FloodDetectionNet.ipynb	18 days ago	3.3 KB
FloodMobiNetV2.ipynb	yesterday	128 KB
FloodResNet50.ipynb	yesterday	93.5 KB
Predict_Cnn_Custom.ipynb	yesterday	22.7 KB
PredictionMobiNetV2.ipynb	yesterday	63.3 KB
PredictionResNet50.ipynb	yesterday	62.6 KB
PredictionVGG16.ipynb	yesterday	62.9 KB
customCnn_model_architecture.png	2 days ago	74.5 KB

Figure 4.3: Jupyter Notebook interface

4.4 Python (tools and libraries)

4.4.1 Python Programming language

Python is a versatile, high-level programming language that's easy to learn and use. It supports object-oriented programming and has powerful data structures. Python is used in many areas, including web development, game creation, computer vision, machine learning, robotics, web scraping, data analysis, automation, scripting, scientific computing, and artificial intelligence. This paper explores Python's popularity, features, application areas, and popular libraries used in Python projects.[37]

Figure 4.4: Tensorflow logo

4.4.2 Tensorflow

TensorFlow is a versatile open-source toolkit, compatible with Python, that greatly streamlines and speeds up the process of building neural networks and machine learning algorithms. It's designed to handle complex numerical computations efficiently, making it an invaluable tool for researchers, developers, and data scientists alike.[38]

TensorFlow

Figure 4.5: tensorflow logo

4.4.3 Keras

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, [39]

K Keras

Figure 4.6: Keras library

4.4.4 Numpy((Numerical Python)

NumPy is the fundamental package for scientific computing in Python. It provides a powerful multidimensional array object and various related tools, like masked arrays and matrices. With NumPy, you can perform fast operations on arrays, including math, logic, shape manipulation, sorting, selecting, I/O, Fourier transforms, basic linear algebra, statistical operations, random simulations, and much more.[40]

Figure 4.7: Numpy Icon

4.4.5 Matplotlib

Matplotlib is a plotting library designed for the Python programming language, along with its numerical mathematics extension, NumPy. It offers an object-oriented API, allowing users to embed plots into applications using various general-purpose GUI toolkits, such as Tkinter, wxPython, Qt, and GTK. Additionally, Matplotlib includes a procedural ”pylab” interface that operates based on a state machine, similar to OpenGL, and is intended to closely mimic MATLAB’s interface, although its use is generally discouraged. The SciPy library also utilizes Matplotlib for its plotting capabilities.[40]

Figure 4.8: Matplotlib library

4.4.6 Pennylane

A Python library for quantum machine learning, automatic differentiation, and optimization of hybrid quantum-classical computations.[41]

Figure 4.9: Pennylane library

4.4.7 Scikit-learn

a popular machine learning library in Python. It provides a wide range of tools and algorithms for tasks. scikit-learn's role is to import specific functions related to metrics evaluation. These functions are used to compute precision, recall, and F1 scores, which are common metrics used to evaluate the performance of classification models.[40]

Figure 4.10: Scikit-learn library

4.4.8 Imutils

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3. [42]

4.5 Plan overview

In this section, we provide a figure that represents the process of our system . This includes dataset division into training and testing, pre-processing, model creation and finally model evaluation.

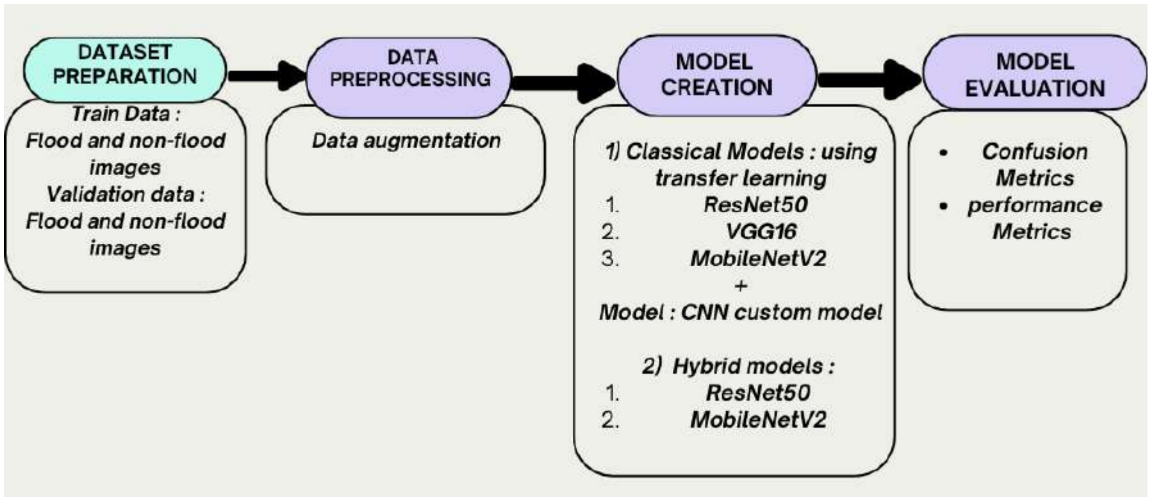


Figure 4.11: process overview

4.6 Hybrid model creation :

Leveraging the principles of quantum mechanics in our approach, PennyLane enables the creation of hybrid models that combine quantum circuits with classical neural networks, leading to enhanced computational capabilities.

4.6.1 the principles of quantum mechanics

a) Quantum Bits (Qubits):

Unlike classical bits, which can be either 0 or 1, qubits can exist in a superposition of states, meaning they can be both 0 and 1 simultaneously. This property allows quantum computers to process a vast amount of information in parallel.

b) Superposition:

Superposition allows quantum systems to explore multiple states simultaneously. For instance, if you have a system with two qubits, it can represent four states (00, 01, 10, 11) at once. Mathematically, a qubit's state can be represented as:

$$|\alpha|^2 + |\beta|^2 = 1$$

Where $|\psi\rangle|\psi\rangle$ is the state of the qubit, and α and β are complex numbers representing the probability amplitudes of the qubit being in state $|0\rangle|0\rangle$ and $|1\rangle|1\rangle$, respectively. The probabilities of these states must add up to 1:

$$|\alpha|^2 + |\beta|^2 = 1$$

4.6.2 Model creation with Pre-trained Models and Hybrid Techniques:

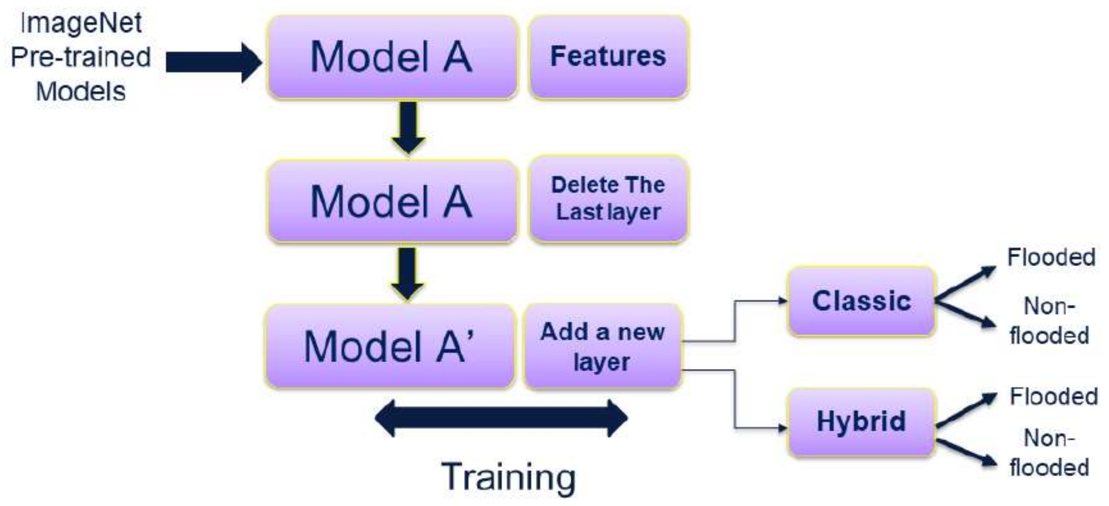


Figure 4.12: This figure represents the structure of the implementation

The figure shows two paths for evaluating the models:

- **Classic Model:** The modified and trained Model A' is evaluated directly as a classic model. The output consists of two categories: Flooded and Non-Flooded.
- **Hybrid Model:** After deleting the last layer and adding a new layer, the model incorporates quantum function simulation. The output categories remain the same: Flooded and Non-Flooded.

Detailed explanation :

1) ImageNet Pre-trained Model:

We've adapted pretrained models from ImageNet. This serves as the initial model (Model A) for further modifications.

2) Model A - Features Extraction:

The pretrained model (Model A) is used to extract features from the input data. This step utilizes the knowledge gained from training on the ImageNet dataset.

3) Model A - Layer Modification:

This step involves deleting the last layer of pretrained Model A to replace it with a new layer.

4) Model A' - New Layer Addition:

After removing the last layer, a new layer is added to create Model A', which can perform a new classification task (e.g., flooded vs. non-flooded).

5) Training:

Training each model on the dataset for its specific task.

4.7 Models Evaluation

we used the classification performance metrics to evaluate our models and We evaluated their accuracy on the training and testing data across 30 epochs.

4.7.1 Classification Performance Metrics

Classification performance metrics plays a crucial role and its an essential tools to evaluate the Efficiency of a classification model.

Accuracy :

Accuracy is the ratio of correctly predicted instances to the total instances.[43]

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

Precision :

Precision (Precision or positive predictive value) calculate the how many positive instances where correct out of total predicted positives.[43]

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

Recall:

Recall (also known as sensitivity or true positive rate) is the ratio of correctly predicted positive instances to all actual positives.[43]

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

F1 score:

The F1 Score: F1 Score is the harmonic mean of precision and recall.[43]

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

4.8 Models Architecture

4.8.1 Model 1 (ResNet50) :

this model based on transfer learning uses a pre-trained ResNet50 model. The use of ResNet50 allows the model to leverage pre-trained weights and sophisticated feature extraction , this architecture consists of an input layer for image data, following a Global Average Pooling layer for dimensionality reduction, a Dropout layer for regularization, and a Dense layer for final classification Flood or non flood.

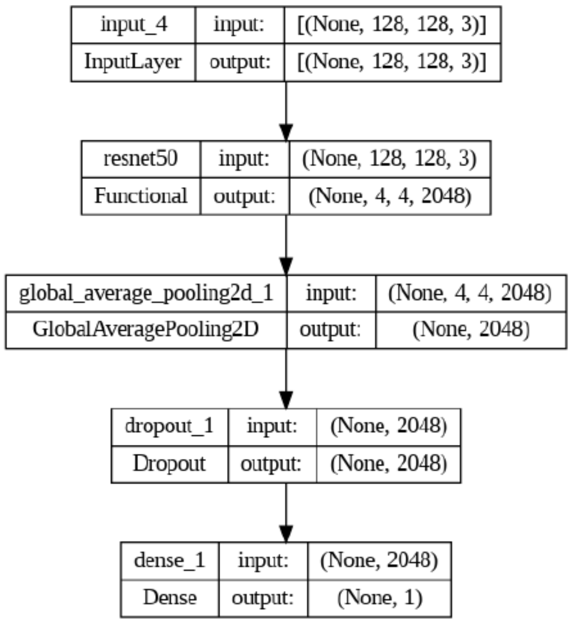


Figure 4.13: Model 1 ResNet50 architecture

4.8.2 Model 1 (VGG16) :

this model based on transfer learning used a pre-trained VGG16 model.

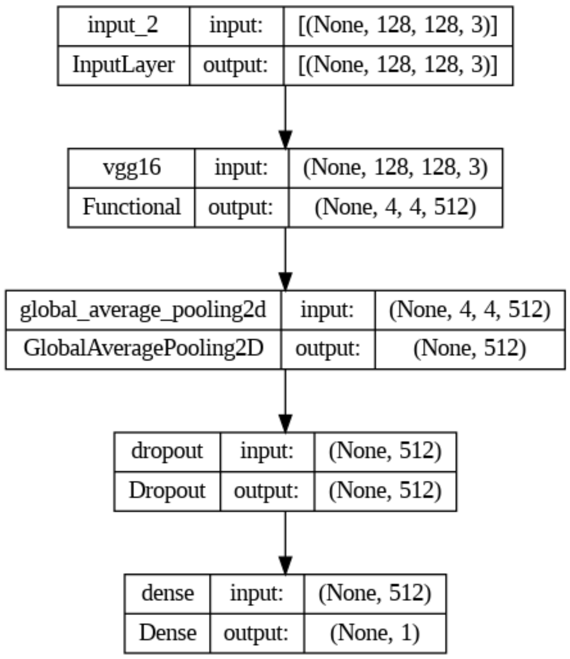


Figure 4.14: Model 2 VGG16 architecture

4.8.3 Model 3 (MobileNetV2) :

this model based on transfer learning used a pre-trained MobileNetV2 model.

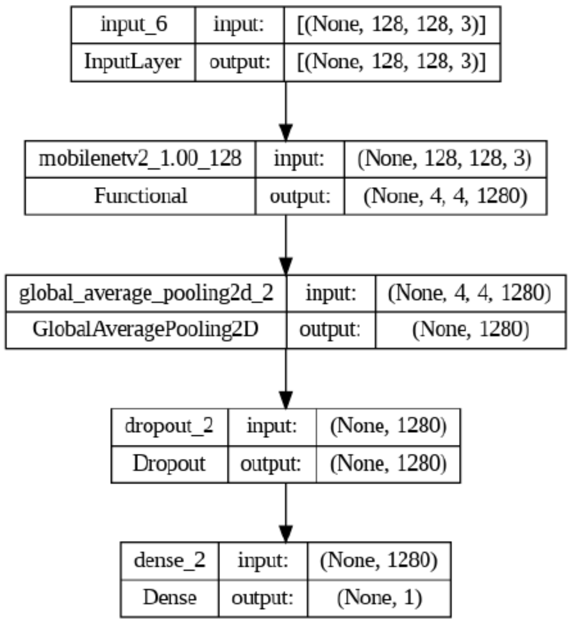


Figure 4.15: Model3 MobileNetV2 architecture

4.8.4 Model 4 (Cnn custom model) :

I designed a custom Convolutional Neural Network (CNN) model from scratch, and here is its architecture. The model is designed for image classification tasks, specifically to classify images as either ”flood” or ”non-flood. The model starts with input layer with image size 128x128 pixels with three color channels (RGB) , the model consists of 3 convolutional layers followed by max-pooling layer , first layer uses 32 filters , the second 64 filters and the third used 128 filters each with kernel size of 3x3 , max pooling with size 2x2. after convolutional layer the model includes a flatten layer that converts from 3D feature map to 1D vector, this vector is passed through a dense layer which it made a binary classification by sigmoid activation funtion and finally a dropout layer to prevent overfitting.

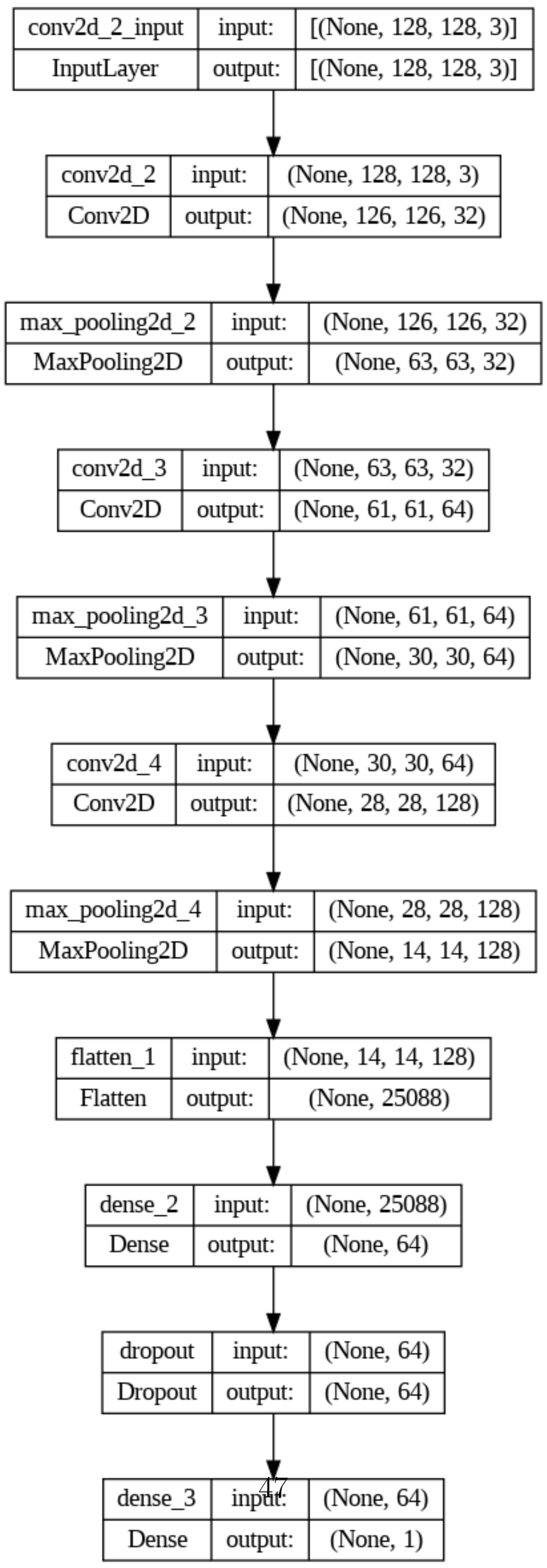

conv2d_2_input | input: | [(None, 128, 128, 3)]
InputLayer | output: | [(None, 128, 128, 3)]
conv2d_2 | input: | (None, 128, 128, 3)
Conv2D | output: | (None, 126, 126, 32)
max_pooling2d_2 | input: | (None, 126, 126, 32)
MaxPooling2D | output: | (None, 63, 63, 32)
conv2d_3 | input: | (None, 63, 63, 32)
Conv2D | output: | (None, 61, 61, 64)
max_pooling2d_3 | input: | (None, 61, 61, 64)
MaxPooling2D | output: | (None, 30, 30, 64)
conv2d_4 | input: | (None, 30, 30, 64)
Conv2D | output: | (None, 28, 28, 128)
max_pooling2d_4 | input: | (None, 28, 28, 128)
MaxPooling2D | output: | (None, 14, 14, 128)
flatten_1 | input: | (None, 14, 14, 128)
Flatten | output: | (None, 25088)
dense_2 | input: | (None, 25088)
Dense | output: | (None, 64)
dropout | input: | (None, 64)
Dropout | output: | (None, 64)
dense_3 | input: | (None, 64)
Dense | output: | (None, 1)

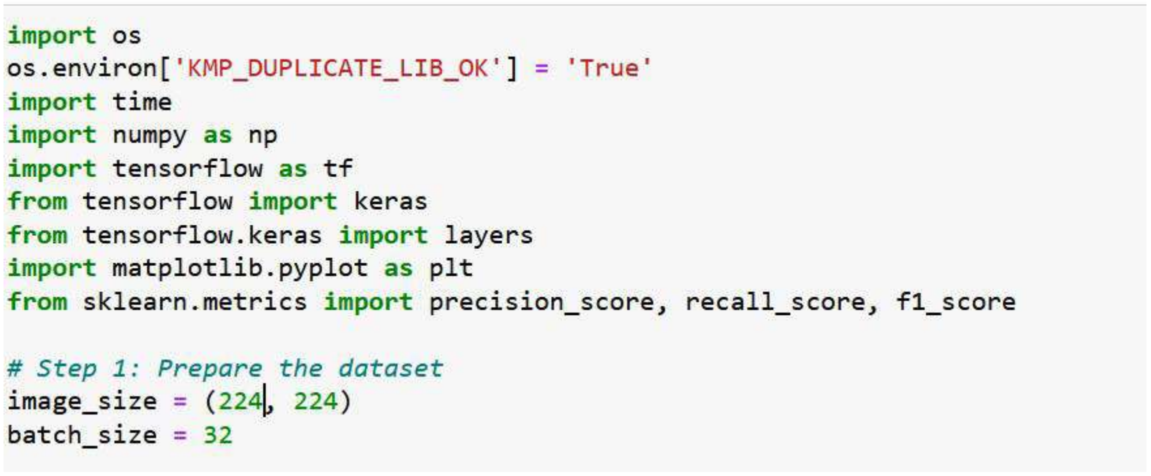
Figure 4.16: Model 4 Custom Cnn architecture

4.9 Programming

4.9.1 Model training

1) Importing libraries:

to use this libraries it should be called and imported

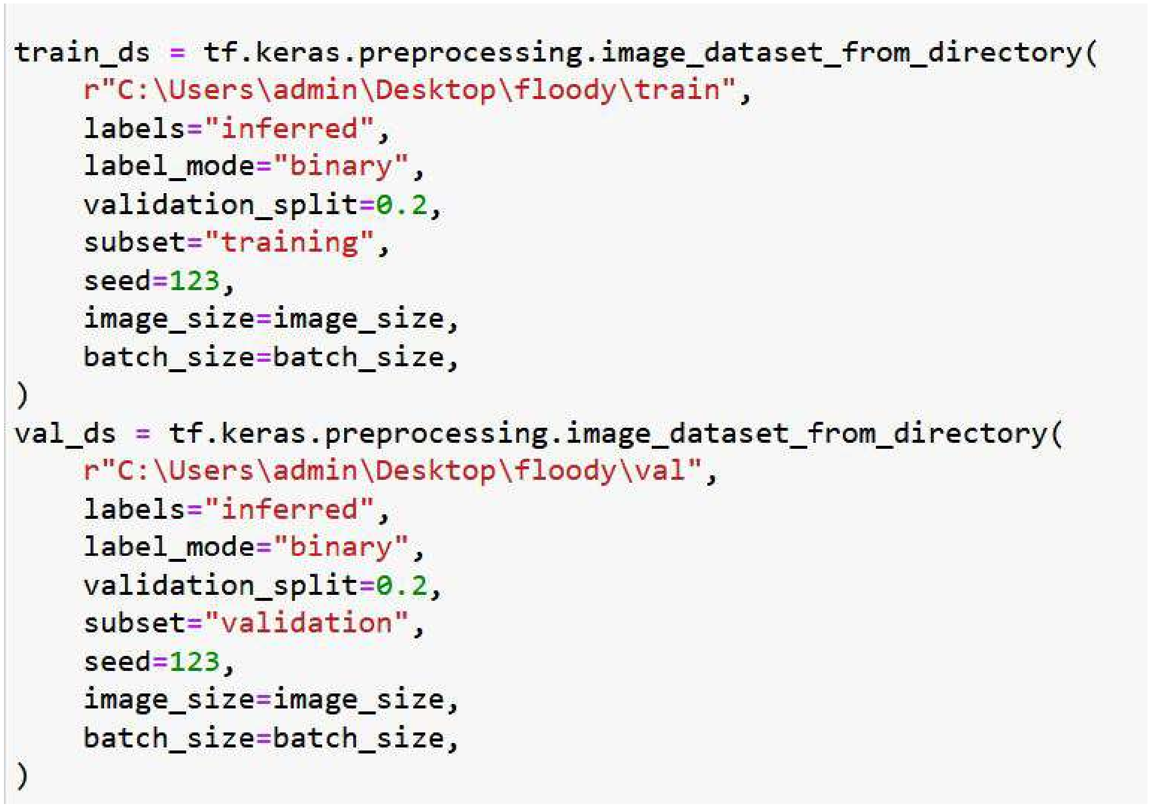


```
import os
os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'
import time
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score

# Step 1: Prepare the dataset
image_size = (224, 224)
batch_size = 32
```

Figure 4.17: Importing Libraries

2) Data path access:

there is two different folder one for train other for validaion.

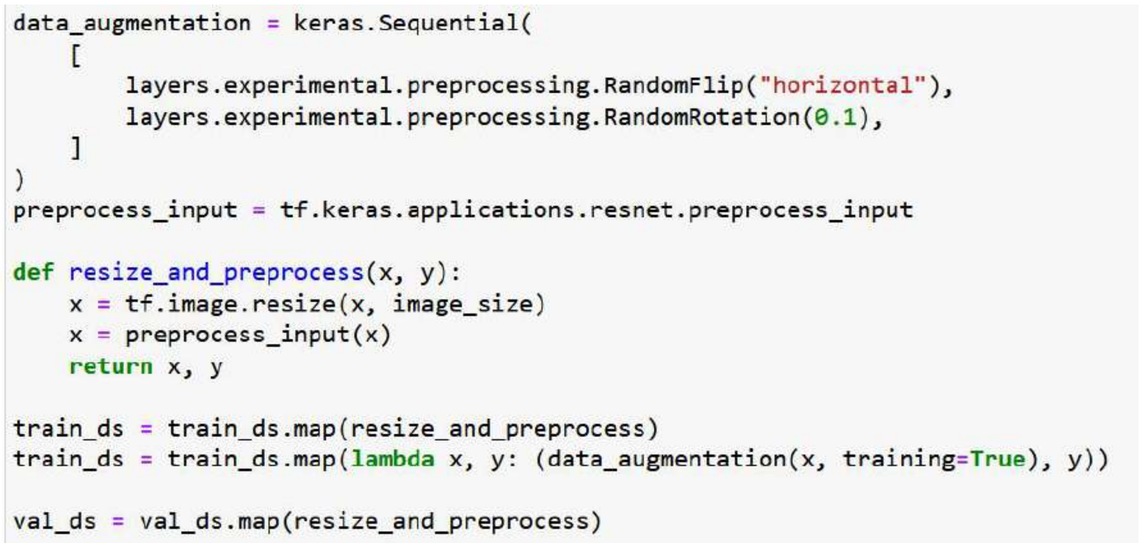


```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    r"C:\Users\admin\Desktop\floody\train",
    labels="inferred",
    label_mode="binary",
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=image_size,
    batch_size=batch_size,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    r"C:\Users\admin\Desktop\floody\val",
    labels="inferred",
    label_mode="binary",
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=image_size,
    batch_size=batch_size,
)
```

Figure 4.18: Dataset access

3) Data augmentation and preprocess

this figure is responsible for augmenting and preprocessing our training images before feeding them into your CNN model. Data augmentation helps to improve the generalization of the model by introducing variability in the training data.

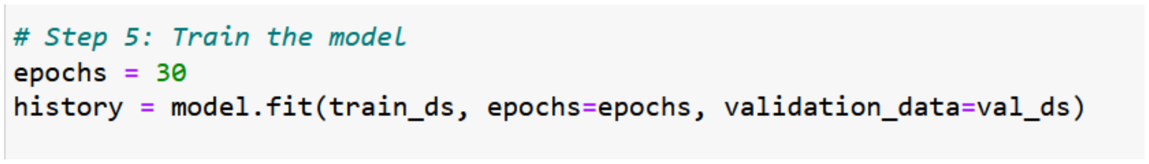


```
data_augmentation = keras.Sequential(
    [
        layers.experimental.preprocessing.RandomFlip("horizontal"),
        layers.experimental.preprocessing.RandomRotation(0.1),
    ]
)
preprocess_input = tf.keras.applications.resnet.preprocess_input

def resize_and_preprocess(x, y):
    x = tf.image.resize(x, image_size)
    x = preprocess_input(x)
    return x, y

train_ds = train_ds.map(resize_and_preprocess)
train_ds = train_ds.map(lambda x, y: (data_augmentation(x, training=True), y))

val_ds = val_ds.map(resize_and_preprocess)
```

Figure 4.19: Data augmentation and preprocess

4) training the model

training the model for 30 epochs and validation metrics for evaluating the model's progress

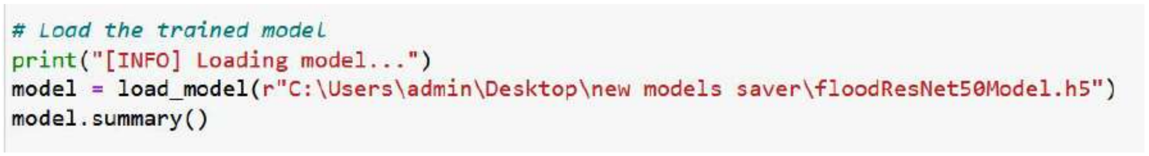


```
# Step 5: Train the model
epochs = 30
history = model.fit(train_ds, epochs=epochs, validation_data=val_ds)
```

Figure 4.20: training and val

4.9.2 Model prediction

Here we are loading the trained model that makes the predictions.



```
# Load the trained model
print("[INFO] Loading model...")
model = load_model(r"C:\Users\admin\Desktop\new models saver\floodResNet50Model.h5")
model.summary()
```

Figure 4.21: loading trained model

1) the predicted images

In this subsection, there are flood and non-flood images predicted by the pre-trained ResNet50 model.

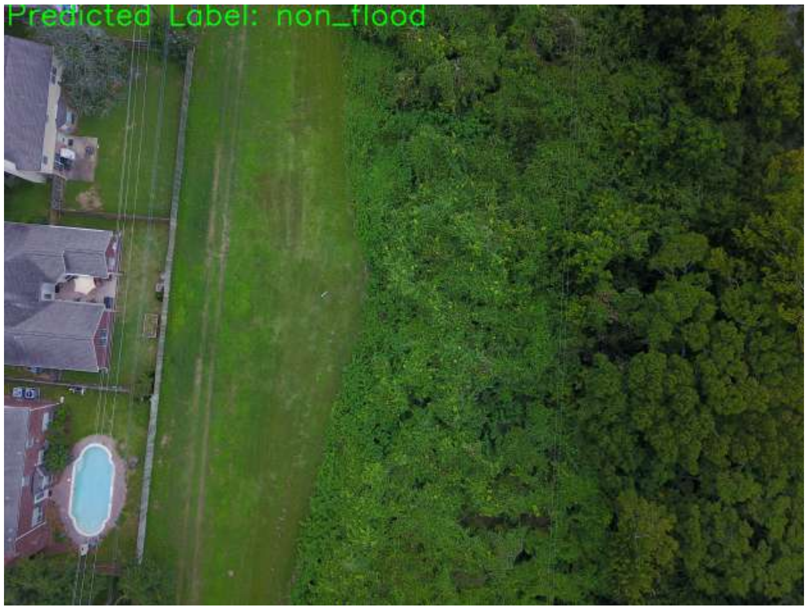


Figure 4.22: predited non flood image

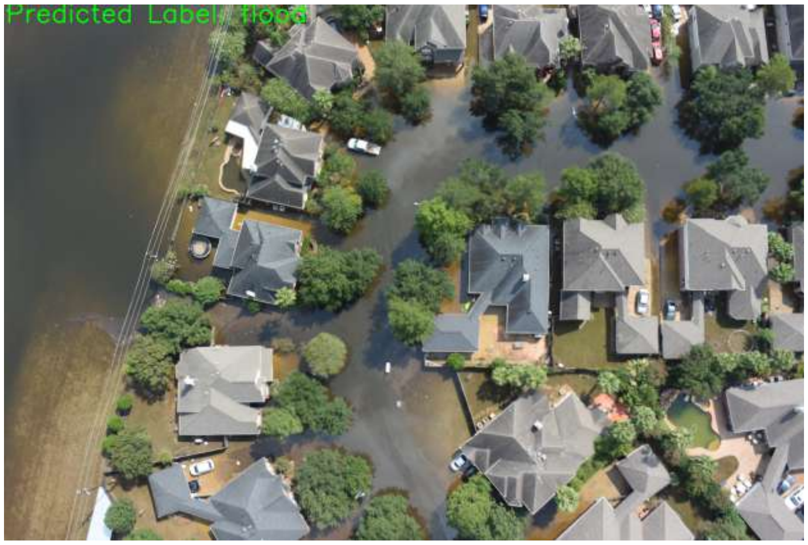
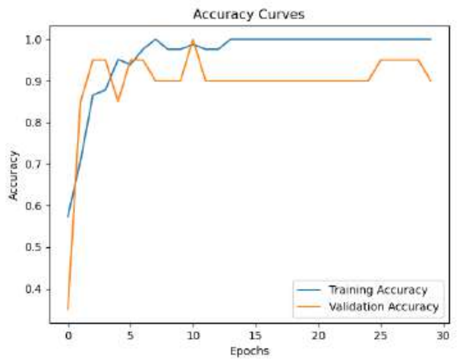


Figure 4.23: predited flood image

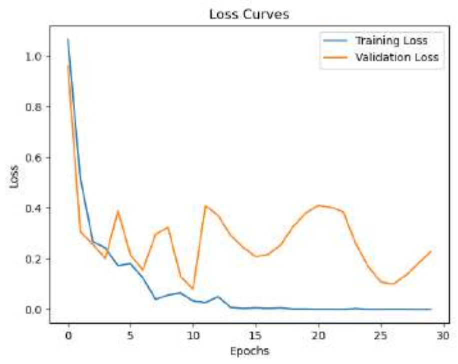
4.10 Models' evaluation

In this section, we will present the model evaluations, graphical presentations, and classification reports for each model used. We employed transfer learning with three models: ResNet50, VGG16, and MobileNetV2, in addition to a custom CNN model.

4.10.1 Model 1 (ResNet50) :



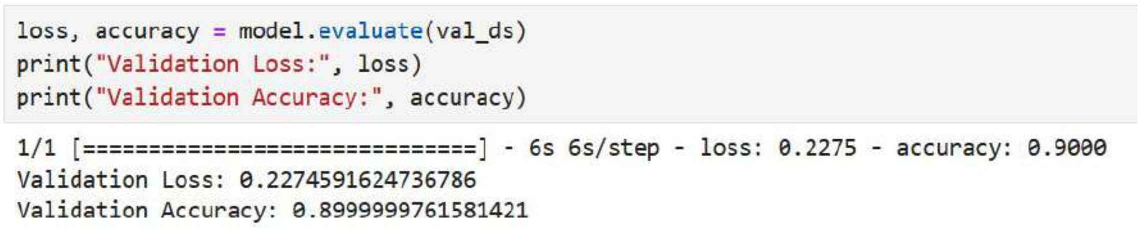
(a) ResNet50 training and validation Accuracy



(b) ResNet50 training and validation Loss

Figure 4.24: Model 1 Accuracy and Loss graph

- **Accuracy Graphs:** these graphs demonstrate the training accuracy and validation accuracy for pre-trained model over the epochs.
- **Loss Graphs:** These graphs demonstrate the training loss and validation loss for pre-trained model over the epochs.



```
loss, accuracy = model.evaluate(val_ds)
print("Validation Loss:", loss)
print("Validation Accuracy:", accuracy)
```

```
1/1 [==============================] - 6s 6s/step - loss: 0.2275 - accuracy: 0.9000
Validation Loss: 0.2274591624736786
Validation Accuracy: 0.8999999761581421
```

Figure 4.25: Resnet50 test accuracy and loss

- **Validation Accuracy:** The model achieved a high validation accuracy of 90%. This indicates that the model correctly classified 90% of the validation images , the model has learned well from the training data to unseen data
- **Validation Loss:** The validation loss of 0.2275 is actually low , which means that the model's prediction are close to the actual labels.

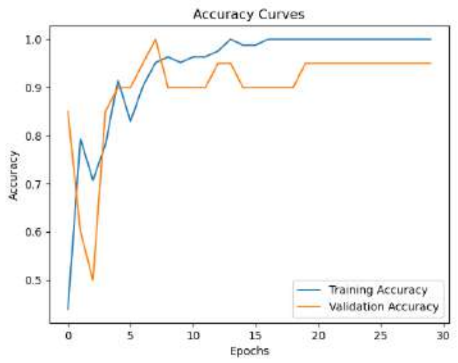
- **Classification report of ResNet50:**

Class	Precision (%)	Recall (%)	F1-Score (%)
Flood	86.67	100.00	92.86
Non-Flood	100.00	71.43	83.33

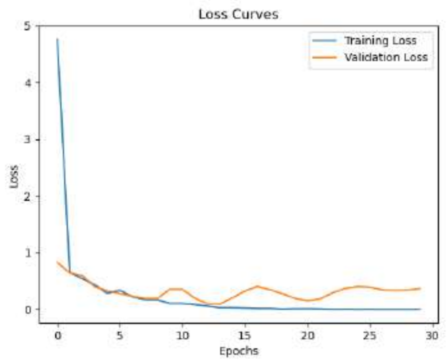
Table 4.1: Classification metrics for ResNet50

the table displays the performance metrics for binary classification , the pretained model show high performance in detecting floods with precision of 86.67% , perfect recall of 100% and F1 score 92.86% ensuring that all instances are correctly. In the non-flood class , precision is perfect 100% no false positive but recall is lower 71.4% , the F1 score is 82.33% there is a need for improved recall.

4.10.2 Model 2 (VGG16):

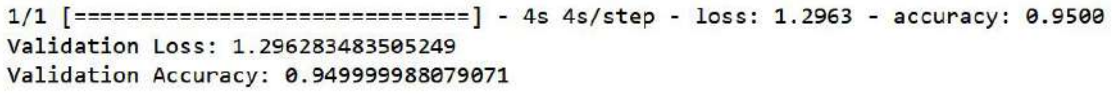


(a) training and validation Accuracy



(b) training and validation Loss

Figure 4.26: Model 2 Accuracy and Loss graph



```
1/1 [==============================] - 4s 4s/step - loss: 1.2963 - accuracy: 0.9500
Validation Loss: 1.296283483505249
Validation Accuracy: 0.949999988079071
```

Figure 4.27: VGG16 test accuracy and loss

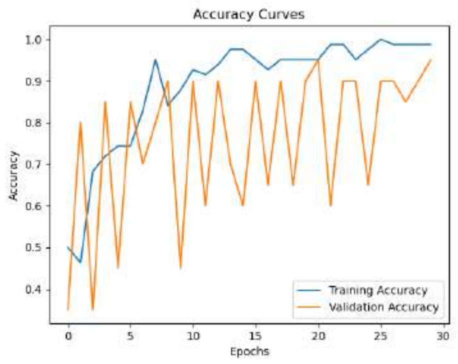
- **Classification report of VGG16:**

Class	Precision (%)	Recall (%)	F1-Score (%)
Flood	92.86	100.00	96.30
Non-Flood	100.00	85.71	92.31

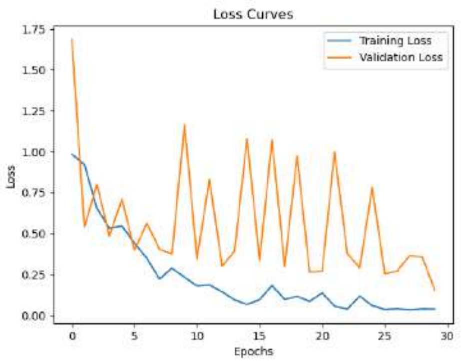
Table 4.2: Classification metrics for VGG16

The VGG16 model demonstrates robust performance, achieving a high validation accuracy close to 95%, with a validation loss of 1.2963. According to the classification report, for the Flood class, the precision is 92.86%, the recall is a perfect 100%, and the F1-score is excellent at 96.30%. For the Non-Flood class, the precision is 100%, but the model shows some misclassification of Non-Flood instances as Flood, with a recall of 85.71% and an F1-score of 92.31%. Overall, the model is highly effective in identifying flood instances, there are some errors in classifying non-flood images.

4.10.3 Model 3 (MobileNetV2):

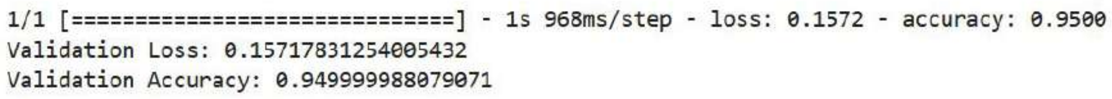


(a) MobileNetV2 training and validation Accuracy



(b) MobileNetV2 training and validation Loss

Figure 4.28: Model 3 Accuracy and Loss graph



```
1/1 [==============================] - 1s 968ms/step - loss: 0.1572 - accuracy: 0.9500
Validation Loss: 0.15717831254005432
Validation Accuracy: 0.949999988079071
```

Figure 4.29: MobileNetV2 test accuracy and loss

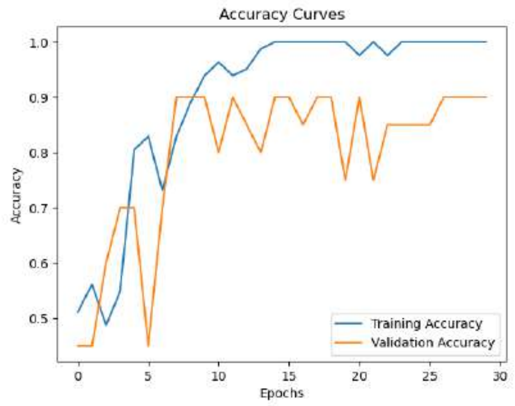
- **Classification report of MobileNetV2:**

Class	Precision (%)	Recall (%)	F1-Score (%)
Flood	92.88	100.00	96.40
Non-Flood	100.00	85.77	92.38

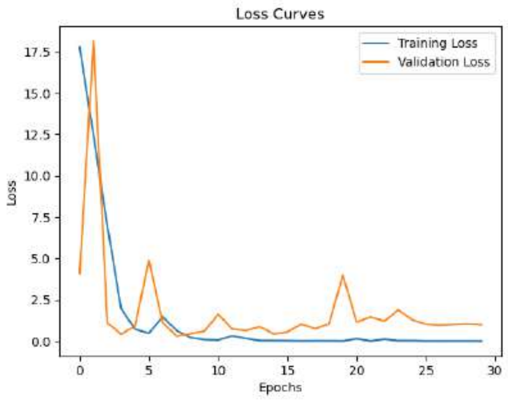
Table 4.3: Classification metrics for MobileNetV2

The MobileNetV2 model performs very well, with a high accuracy of 95% and a low loss of 0.15. For the Flood class, it achieved a precision of 92.88%, a perfect recall of 100%, and an F1-score of 96.40%. For the Non-Flood class, the precision is 100%, recall is 85.77%, and the F1-score is 92.38%. This means the model is excellent at detecting floods but sometimes misclassifies non-floods as floods.

4.10.4 Model 4 (DenseNet201):

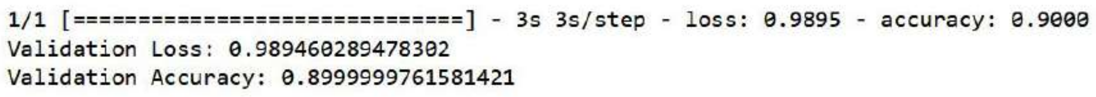


(a) DenseNet201 training and validation Accuracy



(b) DenseNet201 training and validation Loss

Figure 4.30: Model 4 Accuracy and Loss graph



```
1/1 [==============================] - 3s 3s/step - loss: 0.9895 - accuracy: 0.9000
Validation Loss: 0.989460289478302
Validation Accuracy: 0.8999999761581421
```

Figure 4.31: DenseNet201 test accuracy and loss

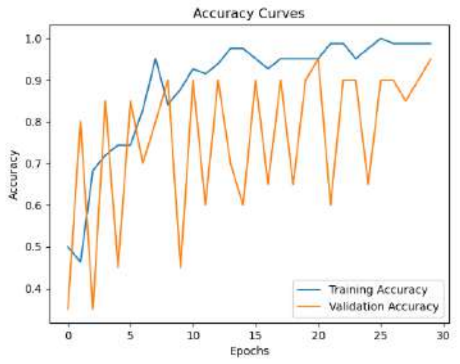
- **Classification report of DenseNet201:**

	Precision (%)	**Recall (%)**	**F1-Score (%)**
Flood Class	88.89	88.89	88.89
Non-Flood Class	90.91	90.91	90.91

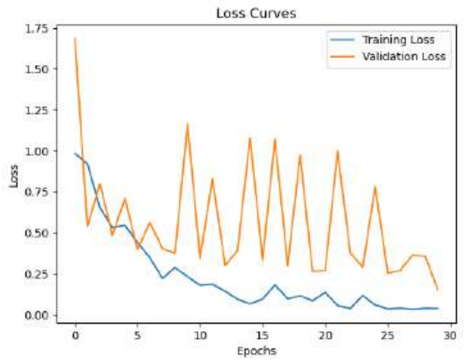
Table 4.4: Classification Performance Metrics

These results suggest that the DenseNet201 model is effective in classifying images into flood and non-flood categories with high accuracy and balanced performance across both classes , These results suggest that the DenseNet201 model is effective in classifying images into flood and non-flood categories with high accuracy and balanced performance across both classes , However, there is still some improvement in reducing the validation loss further to enhance the model's overall reliability and robustness.

4.10.5 Model 5 (Custom CNN):

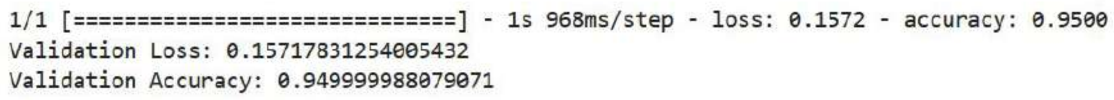


(a) Custom CNN training and validation Accuracy



(b) Custom CNN training and validation Loss

Figure 4.32: Model 4 Accuracy and Loss graph



```
1/1 [==============================] - 1s 968ms/step - loss: 0.1572 - accuracy: 0.9500
Validation Loss: 0.15717831254005432
Validation Accuracy: 0.949999988079071
```

Figure 4.33: Custom CNN test accuracy and loss

- **Classification report of Custom CNN:**

Class	Precision (%)	Recall (%)	F1-Score (%)
Flood	91.33	94.80	93.03
Non-Flood	94.80	60.46	73.83

Table 4.5: CNN custom classification metrics

4.10.6 Hybrid Model (DenseNet201)

The concept of hybrid classical-quantum models involves integrating quantum computing approaches to leverage their strengths and enhance the performance of deep learning and AI applications. This integration leads to the development of more efficient and powerful models.

Result of Prediction :

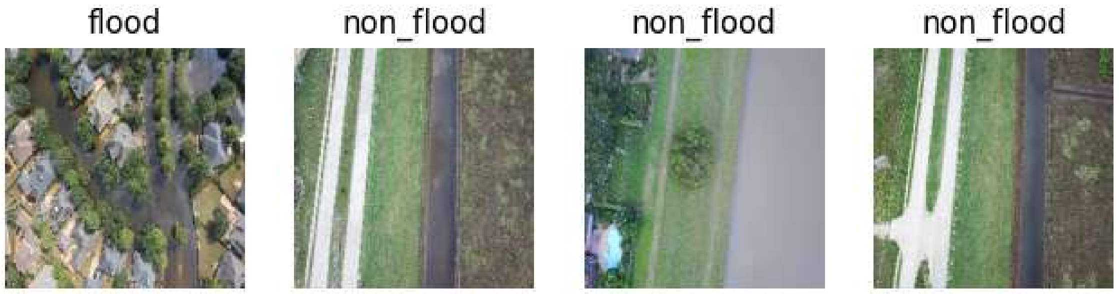
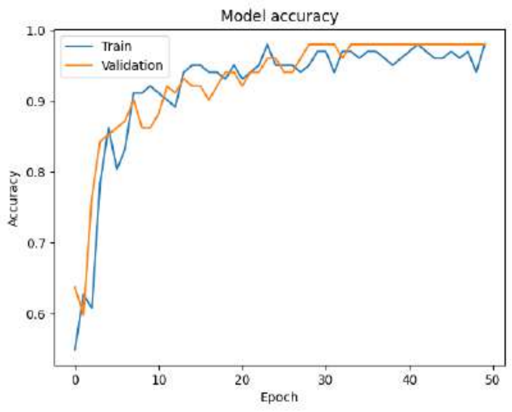
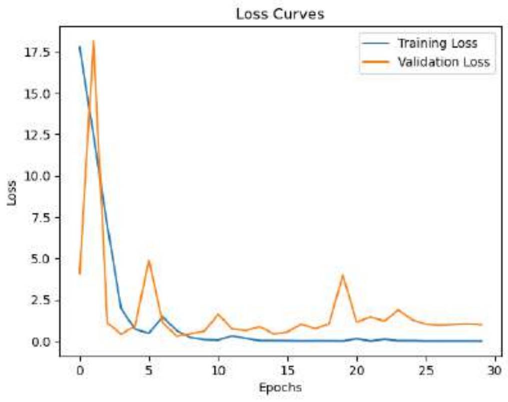


Figure 4.34: Images prediction

Graphic presentation :



(a) training and validation Accuracy



(b) training and validation Loss

Figure 4.35: Model hybrid (DenseNet201) Accuracy and Loss graph

4.10.7 Model evaluation

We conducted our experiment using a pre-trained DenseNet201 model. We built the model, froze its layers, and added a quantum layer for binary classification, achieving an impressive accuracy of 98%. The process is highly efficient, demonstrating the quantum computer's capability to handle complex computations and optimize performance.

Class	Precision (%)	Recall (%)	F1-Score (%)
Flood	99	98	98
Non-Flood	99	98	98

Table 4.6: DenseNet201 hybrid classification report

4.10.8 Hybrid Model (ResNet50)

The hybrid model's performance, with an accuracy of 81%, demonstrates the potential of integrating quantum computing into classical deep learning

Result of prediction :

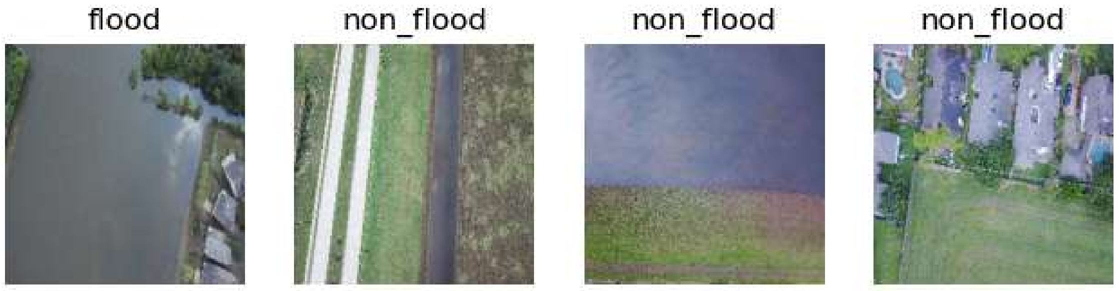
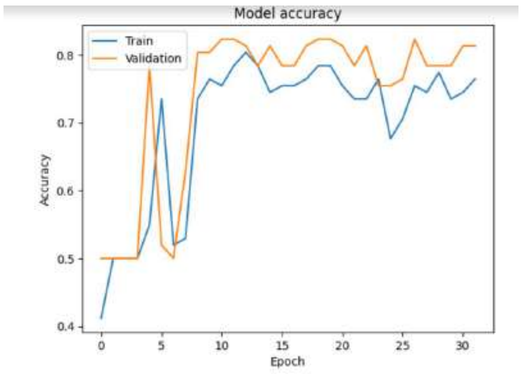
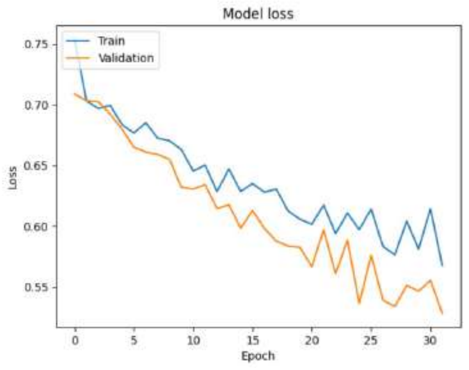


Figure 4.36: Images prediction

Graphic presentation :



(a) training and validation Accuracy

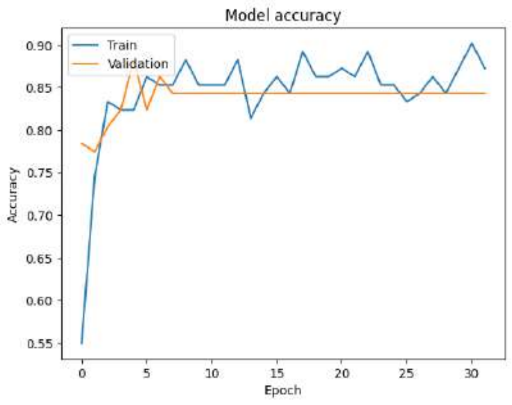


(b) training and validation Loss

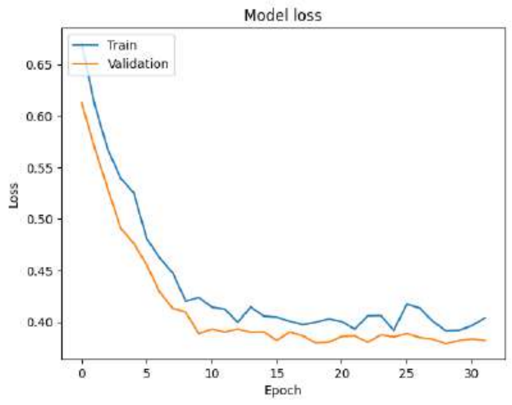
Figure 4.37: Model hybrid (ResNet50) Accuracy and Loss graph

4.10.9 Graphical results of other models :

Classic-classic

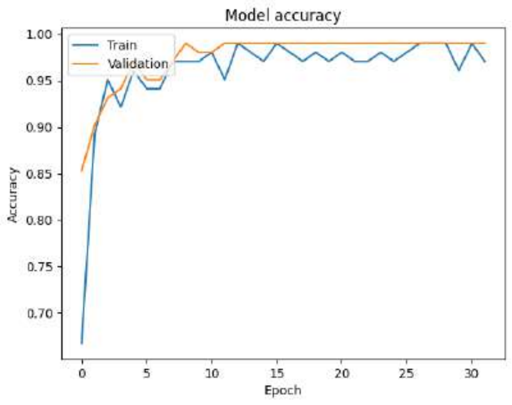


(a) training and validation Accuracy

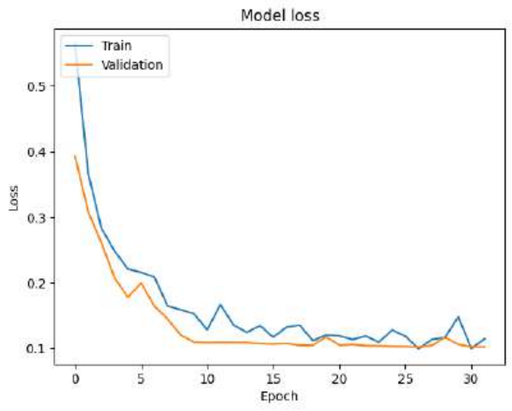


(b) training and validation Loss

Figure 4.38: Model classic (DenseNet121) Accuracy and Loss graph

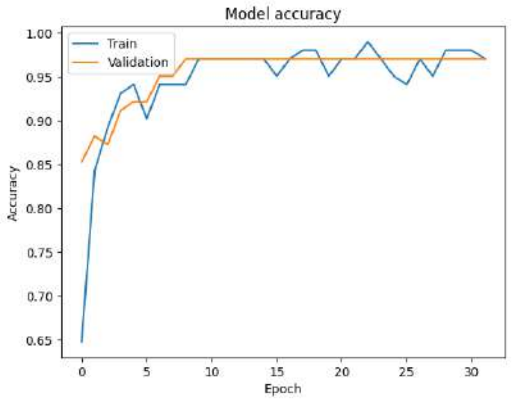


(a) training and validation Accuracy

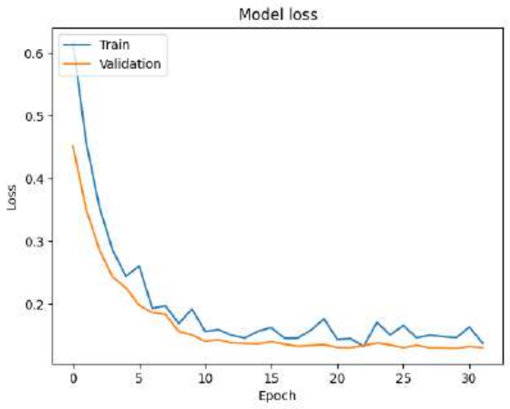


(b) training and validation Loss

Figure 4.39: Model classic (DenseNet169) Accuracy and Loss graph



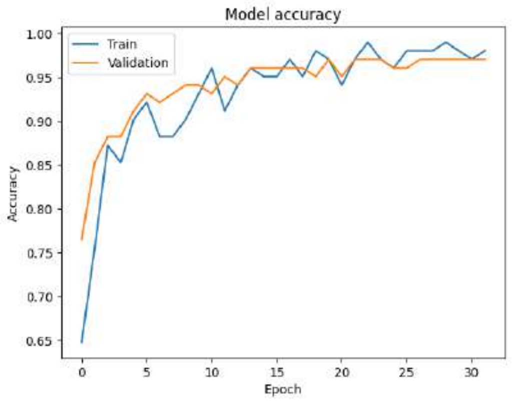
(a) training and validation Accuracy



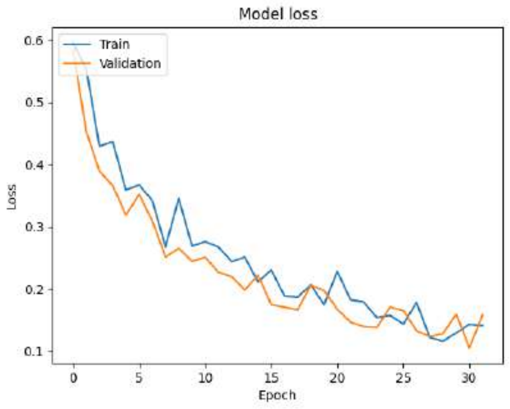
(b) training and validation Loss

Figure 4.40: Model classic (InceptionV3) Accuracy and Loss graph

Classic-Quantum

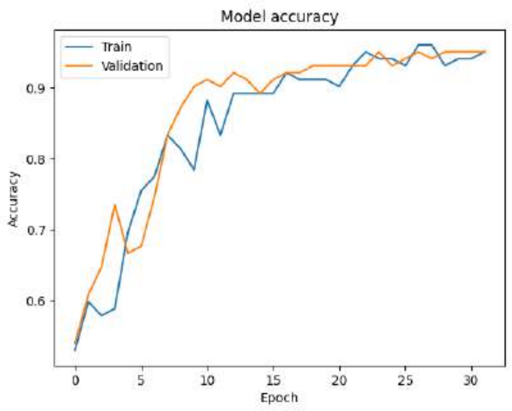


(a) training and validation Accuracy

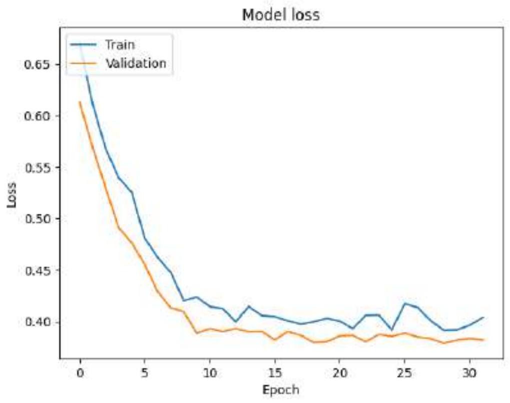


(b) training and validation Loss

Figure 4.41: Model hybrid (DenseNet169) Accuracy and Loss graph

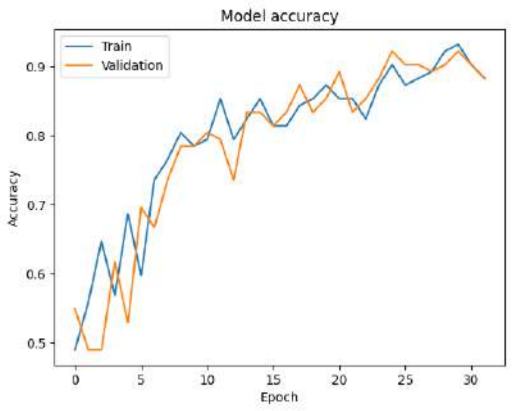


(a) training and validation Accuracy

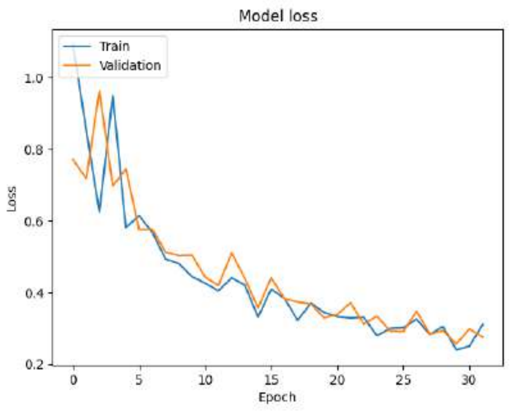


(b) training and validation Loss

Figure 4.42: Model hybrid (DenseNet121) Accuracy and Loss graph

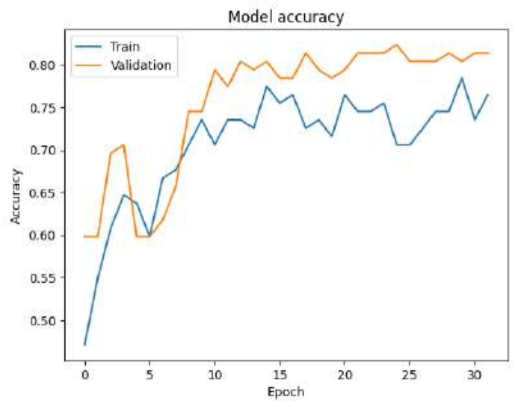


(a) training and validation Accuracy

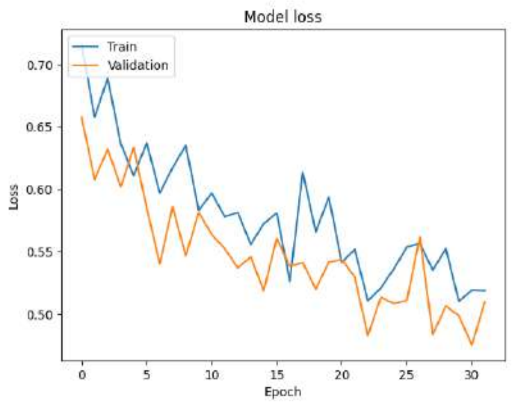


(b) training and validation Loss

Figure 4.43: Model hybrid (InceptionV3) Accuracy and Loss graph



(a) training and validation Accuracy



(b) training and validation Loss

Figure 4.44: Model hybrid (VGG16) Accuracy and Loss graph

4.11 Comparison of Results

4.11.1 comparison of classical models with the state of the art

The table below is a summary for comparing results (Averaged F1-score of our models with other related work.

Related works	**Binary classifier**	**Averaged F1-Score (%)**
Danielle Dias and Ulisses Dias (2018)	DenseNet-121	73.27
	MobileNet	56.82
	InceptionV3	62.69
Nouioua Tarek (2023)	ResNet50	68.66
	VGG16	60
	InceptionV3	58.33
My Models	ResNet50	88.10
	VGG16	94.31
	MobileNetV2	94.39
	DenseNet201	88.89
	custom CNN	83.43
	InceptionV3	93.67

Table 4.7: Comparison of various binary classifiers and their F1-score.

4.11.2 Comparison of Classical and Hybrid Models

MODELS	**Classical-Classical accuracy%**	**Classical-Quantic Accuracy %**
ResNet-50	90	81
VGG-16	84	81
MobileNet-V2	95	95
DenseNet201	97	98
DenseNet169	99	97
DenseNet121	97	95
InceptionV3	92	90

Table 4.8: Accuracy Comparison of Classical and Hybrid Models.

Note : We have tested also DenseNet169, DenseNet121 and InceptionV3 for both classical-classical and classical-quantum Transfer learning models, where we have reported their accuracy results with the other models discussed in the previous section in Table .

4.12 conclusion:

In this chapter, we demonstrated our contribution, which involves a deep learning model and transfer learning technique for the classification of flood and non-flood satellite images. We discussed the results and compared them with other models.

General Conclusion

this project focused on developing and evaluating deep learning models for classifying flood and non-flood satellite images. Through the utilization of various CNN architectures and transfer learning techniques, we aimed to improve the accuracy of flood detection from satellite imagery. the custom CNN model achieved the highest accuracy of 96.7%. Additionally, transfer learning with pre-trained models like VGG-16 and MobileNet-V2 showed promising results, each achieving an accuracy of 95However, the hybrid model, integrating quantum computing approaches, The DenseNet201 model demonstrated an impressive accuracy of 98%, highlighting its effectiveness in flood detection tasks and ResNet50 demonstrated accuracy of 81% This suggests that while hybrid classical-quantum models hold potential, there are challenges and limitations to be addressed, including hardware constraints. In the future, more research is necessary to develop advanced techniques that make AI-based flood detection systems more accurate and reliable.

Bibliography

[1] Nouioua Tarek. Flood detection in satellite images using deep learning. 12 2023.

[2] Tensorflow . https://ceur-ws.org/Vol-2283/MediaEval$_1 8_p aper_6 1.pdf$/.

[3] Tensorflow . https://gemini.google.com/app/.

[4] Rahul Chauhan, Kamal Kumar Ghanshala, and R.C Joshi. Convolutional neural network (cnn) for image detection and recognition. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 278–282, 2018.

[5] Muhammad Zakaria and Waheb Al-Areeqi. Flood monitoring and warning systems: A brief review. *Journal of Southwest Jiaotong University*, 56:140–156, 06 2021.

[6] Danielle Dias and Ulisses Dias. Flood detection from social multimedia and satellite images using ensemble and transfer learning with cnn architectures. In Martha Larson, Piyush Arora, Claire-Hélène Demarty, Michael Riegler, Benjamin Bischke, Emmanuel Dellandréa, Mathias Lux, Alastair Porter, and Gareth J. F. Jones, editors, *Working Notes Proceedings of the MediaEval 2018 Workshop, Sophia Antipolis, France, 29-31 October 2018*, volume 2283 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[7] transfer-learning . https://skyengine.ai/se/skyengine-blog/128-what-is-transfer-learning/.

[8] AN Ramesh, Chandra Kambhampati, John RT Monson, and PJ Drew. Artificial intelligence in medicine. *Annals of the Royal College of Surgeons of England*, 86(5):334, 2004.

[9] Kouassi Konan Jean-Claude. A comprehensive overview of artificial intelligence (now published in proceedings of the cs it conference - cscp, vol. 12, no. 23, pp. 173-194, doi:10.5121/csit.2022.122314), 10 2022.

[10] artificial intelligence . https://insights.btoes.com/what-is-artificial-intelligence/.

[11] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[12] Kouassi Konan Jean-Claude. A comprehensive overview of artificial intelligence (now published in proceedings of the cs it conference - cscp, vol. 12, no. 23, pp. 173-194, doi:10.5121/csit.2022.122314), 10 2022.

[13] transfer-learning . https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644/.

[14] Jayvadan Patel and Anita Patel. Chapter 10 - artificial neural networking in controlled drug delivery. In Munish Puri, Yashwant Pathak, Vijay Kumar Sutariya, Srinivas Tipparaju, and Wilfrido Moreno, editors, *Artificial Neural Network for Drug Design, Delivery and Disposition*, pages 195–218. Academic Press, Boston, 2016.

[15] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197, 1991.

[16] S. Abinaya and M.K. Kavitha Devi. Chapter 12 - enhancing crop productivity through autoencoder-based disease detection and context-aware remedy recommendation system. In Mohammad Ayoub Khan, Rijwan Khan, and Mohammad Aslam Ansari, editors, *Application of Machine Learning in Agriculture*, pages 239–262. Academic Press, 2022.

[17] İsmail Akgül. *ACTIVATION FUNCTIONS USED IN ARTIFICIAL NEURAL NETWORKS*, pages 41–58. 10 2023.

[18] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[19] Ahmed Afif Monrat, Raihan Islam, Mohammad Hossain, and Karl Andersson. *Challenges and Opportunities of Using Big Data for Assessing Flood Risks: Trends, Issues, and Challenges*, pages 31–42. 07 2018.

[20] Chandrahas Mishra and D. Gupta. Deep machine learning and neural networks: An overview. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 6:66, 06 2017.

[21] transfer-learning . https://levity.ai/blog/difference-machine-learning-deep-learning/.

[22] cnn . https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/.

[23] Tensorflow . https://www.techtarget.com/searchdatamanagement/definition/TensorFlow/.

[24] Tensorflow . https://techvidvan.com/tutorials/tensorflow-applications/.

[25] Keras applications. https://keras.io/api/applications/.

[26] vgg-very-deep-convolutional-networks. ://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/.

[27] Residual Networks (ResNet) in Deep Learning. https://www.geeksforgeeks.org/vgg-16-cnn-model/.

[28] Thanh-Hai Nguyen, Thanh-Nghia Nguyen, and Ba-Viet Ngo. A vgg-19 model with transfer learning and image segmentation for classification of tomato leaf disease. *AgriEngineering*, 4(4):871–887, 2022.

[29] Jiazhi Liang. Image classification based on resnet. *Journal of Physics: Conference Series*, 1634:012110, 09 2020.

[30] Ulzhalgas Seidaliyeva, Daryn Akhmetov, Lyazzat Ilipbayeva, and Eric Matson. Real-time and accurate drone detection in a video with a static background. *Sensors*, 20:3856, 07 2020.

[31] Spiros Georgakopoulos, K. Kottari, Konstantinos Delibasis, Vassilis Plagianakos, and Ilias Maglogiannis. Improving the performance of convolutional neural network for skin image classification using the response of image analysis filters. *Neural Computing and Applications*, 31, 06 2019.

[32] Zhongling Huang, Zongxu Pan, and Bin Lei. Transfer learning with deep convolutional neural network for sar target classification with limited labeled data. *Remote Sensing*, 9:907, 08 2017.

[33] Sajja Tulasi Krishna and Hemantha kumar Kalluri. Deep learning and transfer learning approaches for image classification. 06 2019.

[34] Nouioua Tarek and Belbachir Hafid. The quantum computer and the security of information systems. pages 1–9, 09 2021.

[35] Nouioua Tarek and Ahmed Belbachir. The quantum computer for accelerating image processing and strengthening the security of information systems. *Chinese Journal of Physics*, 81, 11 2022.

[36] Damien Rolon-Mérette, Matt Ross, Thaddé Rolon-Mérette, and Kinsey Church. Introduction to anaconda and python: Installation and setup. *Quant. Methods Psychol*, 16(5):S3–S11, 2016.

[37] AL Sayeth Saabith, T Vinothraj, and M Fareez. Popular python libraries and their application domains. *International Journal of Advance Engineering and Research Development*, 7(11), 2020.

[38] Pramod Singh, Avinash Manure, Pramod Singh, and Avinash Manure. Introduction to tensorflow 2.0. *Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python*, pages 1–24, 2020.

[39] Navin Kumar Manaswi. *Understanding and Working with Keras*, pages 31–43. Apress, Berkeley, CA, 2018.

[40] MMM Fareez, Vinothraj Thangarajah, and Sayeth Saabith. Popular python libraries and their application domains. 11 2020.

[41] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[42] imutils . https://chatgpt.com/.

[43] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.