CAPSTONE PROJECT REPORT

(Project Term June-July 2024)

CLOUD DETECTION USING MACHINE LEARNING

Submitted by

Kinshuk	**12018893**
Rohan Tyagi	**12008115**
Himmi Raja	**12017317**

Project Group Number :

Course Code : CSE445

Under the Guidance of

Kamalpreet Kaur Bagral

(Assistant Professor)

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab(India)

DECLARATION

We hereby declare that the project work entitled “Cloud Detection Using Machine Learning” is an authentic record of our own work carried out as requirements of Capstone project for the award of B .Tech degree in P-192 : B. Tech(Computer Science & Engineering) from lovely Professional University, Phagwara, under the guidance of Kamalpreet Kaur Bagral during July to August 2024. All the information finished in this capstone project report is genuine and based on our own intensive work.

Project Group Number :

Name of Student 1 : Kinshuk signature of student 1:

Registration Number : 12018893

Name of Student 2 : Rohan Tyagi signature of student 2:

Registration Number : 12008115

Name of Student 3 : Himmi Raja signature of student 3:

Registration Number : 12017317

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. Capstone Project is fit for the submission and partial fulfilment of the conditions for the award of a B. Tech degree in P-192: B.Tech
(Computer Science & Engineering) from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Kamalpreet Kaur Bagral

Assistant Professor

School of Computer Science and Engineering,

Lovely Professional University,

Phagwara, Punjab,

Date:

ACKNOWLEDGEMENT

Working on this Capstone project " **Cloud detection using machine learning**" was a source of huge knowledge and great experience to us. We would like to express our appreciation to Ms. Kamalpreet Kaur Bagral for her guidance, support, and encouragement in carrying out this Capstone project. We would also like to thank to the faculty members and our parents for their support. We would also like to extend our sincere thanks to Lovely Professional University, for providing us with the necessary resources and facilities required for this project. The access to the state-of-the-art equipment and software tools has greatly helped us in achieving the project objectives.

TABLE OF CONTENTS

1. INTRODUCTION

Cloud detection methods play an essential role in remote sensing and satellite image analysis, helping identify and classify clouds found within captured images. Clouds obstruct Earth's surface, distorting data obtained from images used as weather forecasting or climate conditions monitoring tools or environmental impact evaluation evaluation processes; consequently, understanding their dynamics is integral for understanding this remote sensing information accurately.

Cloud detection techniques must be accurate due to the specific challenges presented by cloud cover in satellite images. Clouds may obscure important land features, alter light reflectance from Earth surface reflection, and cause variations that hinder algorithmic image analysis; consequently, accurate recognition and delineation of cloud features is vital in order to guarantee their subsequent analysis and application.

Your choice of neural network architecture is of critical importance when selecting computer vision tasks like cloud detection. Models like ResNet-50, VGG-19, Xception and YOLO bring distinct benefits; ResNet-50 for its extensive residual connections while VGG-19 excels with feature extraction efficiency; while Xception excels due to depth-wise separable convolutions as a depth saver and finally YOLO is used real time object detection capability.

Among these designs, YOLO (You Only Look At Once) stands out in particular among cloud-based detection software applications. While traditional structures focus on feature extraction or image classification, YOLO was specifically built to detect objects in real time; its speedy identification in various environments makes it well-suited to dynamic cloud detection satellite images which change quickly over time. With one forward step processing images through one forward step YOLO offers efficient speedy cloud cover delineation in speedy fashion.

Utilizing YOLO's strength for object and cloud detection techniques will increase efficiency and precision with how clouds are detected in satellite imagery, not only increasing accuracy of remote sensing data but facilitating more informed decision making in fields that rely on precise environmental information. This technology not only increases accuracy in remote sensing imagery but it allows more educated decision making regarding environmental data sources.

Cloud detection techniques utilizing sophisticated neural networks such as YOLO are an invaluable contribution to satellite image analysis, meeting an urgent need to quickly and precisely detect clouds for environmental research, scientific investigations, or operational uses. They meet this demand quickly and precisely with unparalleled reliability - providing applications with crucial reliable remote sensing data necessary for environmental studies, scientific experiments or operational use with confidence.

2. LITRATURE SURVEY

G. E. Bostanc's research paper entitled, "Cloudy/Clear Weather Classification Using Deep Learning Techniques using Cloud Images", 2022 examines the difficulty posed by meteorological mistakes to accurate forecasting weather on mobile phones despite increasing accessibility of weather predictions via these platforms. This study addresses human error inherent to traditional forecasting of weather and suggests using deep learning methods to mitigate them through classification of large volumes of cloud images taken by meteorological instruments from the ground. This study compares four deep learning pretrained algorithms - MobileNet V2, VGG-16 ResNet-152 V2 and DenseNet-201 with VGG-16 having the best accuracy at 91.4 percent accuracy as reported by R. Dhir et al. (2022). Their research paper entitled, "Classification of Land Use and Cover Imageries with Deep Learning Approaches", highlights the value of deep learning for environmental applications using remote sensing imagery, specifically with respect to land use/cover classification (LULC classification). S. Ji et al.'s research reviews different DL designs, outlining their advantages over conventional techniques, and reaching an overall accuracy rate of 92.75% with recall rate at 97.08% and F1 score reaching 94.39%. 2021). "Simultaneous Cloud Detection and Removal from Bitemporal Remote Sensing Images Using Cascade CNN" presents an advanced deep learning framework designed to assist with simultaneous cloud removal/detection using Landsat 8 datasets. Their approach, using VGG16, Deeplab v3+ and U-Net technologies to achieve 96% accuracy by efficiently handling scale-related issues on remote sensing images. Furthermore, N. Zhou et al. (2015) developed their method using VGG16, Deeplab v3+, U-Net technologies in tandem. (2021). They propose in "Geo-Parcel-Based Change Detection Utilizing Optical and SAR Image within Cloudy Areas" an algorithm for detecting changes in regions affected by clouds by using multiple source remote sensing images from multiple source remote sensing platforms, tested at Gui'an Test Site to achieve an accuracy rate of 94% while simultaneously showing its efficiency at detecting altered geoparcels via deep learning models.

There was some debate as to who owned what. After further deliberations it was agreed on the following approach. Alan Li's groundbreaking study from February 2020 entitled, "Cloud Detection Algorithms for Multimodal Satellite Imaging Using Convolutional Neural Network," highlights the crucial role played by cloud detection algorithms for remote sensing applications to enhance data processing by decreasing cloud interference on surfaces of particular interest and improve data processing efficiency. Cloud detection methods present unique difficulties on surfaces with high albedo (such as snow or sand) as well as for transfers between platforms for observation. Li provides an artificial neural network (ANN) algorithm

developed for use by NASA NeMO-Net project to detect clouds and shadows of cloud fields from multichannel satellite images from World-View-2 (WV-2) as well as Sentinel-2 (S-2) using RGB and NIR channels to do just this task. This CNN algorithm, trained with WV-2 images, has achieved an accuracy rate of 89% in cloud detection - outperforming its initial S-2 cloud mask, which had only 81% accuracy for cloud identification. An innovative domain-adaptation strategy enhances an algorithm's capacity to share knowledge across satellites, with potential to increase prediction accuracy for clear as well as cloudy pixels. Similar to Seema Mahajan's review in June 2019, "Cloud Detection Methodologies: Variants and Development," Seema Mahajan examines various cloud identification methodologies between 2004 and 2018, using satellite data and reaching an accuracy rate of 98% using hybrid techniques that combine atmospheric parameters with an artificial neural network. Jingyu Yang's paper from August, "CNN-Based Cloud Detection for Remote Imaging with Cloud-Snow Coexistence," details CDnetV2, an encoder-decoder neural system equipped with adaptive feature fusion and high-level semantic guidance capabilities; CDnetV2 achieves higher accuracy when applied to ZY-3 satellite data. Qingyong Li's May 2012 study "Thin Cloud Detection in All-Sky Images using Markov Random Fields," addresses issues of poor contrast and unclear lines when viewing sky imagery from ground-based platforms; his method yielded an accuracy rate of 83.4 percent. Rachana Gupta and Pradip Panchal's paper from August 2015 entitled, "Advancement of Cloud Detection Algorithm in Satellite Images Utilizing Color Models," explores various color models to detect clouds with 92.4 percent precision using RGB in VIRR data.

Recent advances in cloud detection techniques are significantly improving precision and reliability across a range of remote sensors, according to Li et al. (2015). Researchers from Georgia Tech developed a hybrid classification method which incorporates Threshold Exponential Spectral Angle Map (TESAM), adaptive Markov Random Field (aMRF), and Dynamic Stochastic Resonance (DSR), with the purpose of improving cloud detection in hyperspectral photos. Arun et al. developed an exceptional method that achieved an outstanding 96.28 percent precision rating while outclassing traditional methods, specifically handling issues of misclassification and noise efficiently. Arun et al. Hayatbini et al. developed their Fog Stability Index (FSI) to detect fog and low clouds by employing the Airborne Imaging Aerosol Satellite Imagery Analysis Service Interface and Model of Wind Data across Indo-Gangetic Plain. This technique demonstrated 84.63% accuracy during daytime conditions as well as night time conditions with heavy clouds present - Hayatbini et al. CDnetV2 is a gradient-based cloud image segmentation algorithm featuring adaptive feature fusion and high-level semantic guidance to increase accuracy in cloud detection. Changhui et al. developed an effective cloud detection method using feature extraction employing grayscale, frequency and texture characteristics as features to detect clouds. Their method outclasses

existing methods which offer accuracy up to 98% with significant enhancements over PersiaNN-CCS methods. They achieved an accuracy of 90% in distinguishing cloud formations from ground-based objects, improving on Frantz et al.'s solution. Fmask was enhanced to incorporate the Cloud Displacement Index (CDI). CDI uses parallax effects of multiple near-infrared angles to distinguish clouds from bright surfaces. Their method significantly enhanced general accuracy to 95% while meeting challenges associated with Sentinel-2 images not featuring thermal bands. Finally, Song et al. provided guidance that allowed their algorithm to function more reliably under conditions where thermal data could not be extracted for analysis. Study of cloud detection techniques using MODIS multispectral imagery achieved accuracy up to 95% using both spatial and spectral relationships, pattern recognition methods, statistical recognition tools to utilize all MODIS channels to precisely mask cloud coverage, as well as pattern and statistical recognition methods to effectively utilize MODIS' spectral channels to precisely mask clouds.

Sun et al. have developed an improved geometry-based method of cloud shadow detection within Landsat 8 OLI images by employing dynamic cloud height ranges and object-based/spectral analysis techniques. This method achieved accuracy of at least 80%, significantly surpassing Fmask's accuracy of 60% while simultaneously decreasing leakage of cloud shadows (Ishida et al. 2009). MODIS developed an SVM-based cloud detection technique using adjustable parameters that reduce chances of inaccurate results. Li et al. (2008) have demonstrated the success of their methodology using discriminant analysis combined with feature space adjustments as it achieved over 90% accuracy and reliable performance across all circumstances. MSCN is a deep learning-based cloud detection method using multiscale convolutional characteristics for detection. MSCN scored an outstanding 97.35% accuracy when tested against GaoFen-1 image data and succeeded in distinguishing cloud from bright surfaces over traditional techniques, outdoing them significantly. According to Oishi et al, MSCN provided greater precision when used against this dataset than their traditional techniques did (Oishi et al, 2018). CLAUDIA3, an SVM-based cloud discrimination algorithm specifically for GOSAT-2 was assessed for evaluation alongside its predecessor CLAUDIA1 for effectiveness and comparison purposes. CLAUDIA3's newly developed algorithm demonstrated an accuracy rate of 89.5 percent when testing in tropical rainforest environments and enhanced cloud detection on surfaces where snow exists. Wu and Shi developed a novel method of deep learning for cloud detection that utilizes multi-level feature extraction for improved cloud detection results. Their approach, combining deep convolutional neural networks with composite image filters, produced intersections exceeding 85.38 percent. Additionally, detailed cloud masks were produced that demonstrated significant improvements in handling complex images.

Ricciardelli et al. developed the MACSP algorithm for analysis of MSG-SEVIRI data. This hybrid approach incorporates physical, statistical and temporal techniques. MACSP boasts an accuracy rate of 91.8 percent versus SAFNWC's 89%; thus showing its superior performance when classifying cloudsy pixels against MODIS clouds masks (Chen et al. 2010). At TeckNet we have successfully addressed high-resolution cloud detection with multiconvolutional neural networks (MCNNs). Deng et al.'s method, which combined adaptive segmentation with multiscale feature extraction using MCNNs, achieved 95%-98% accuracy when it came to the detection of cloud levels. Deng et al. also conducted extensive experimentation in order to further their research in this field. As part of their efforts, researchers from Cornell have devised an advanced cloud detection method utilizing natural scene statistics and Gabor attributes that overcomes some of the weaknesses found in existing techniques. Their new technique blends preprocessing, seed extraction, region building steps into an overall accuracy rating of 93%. Ghasemian and Akhoondzadeh developed two random forest techniques called FLFRF and DLFRF that use multiple textual characteristics as input, to achieve higher accuracy than traditional CNN or SVM techniques. Liu et al. developed methods with superior accuracy and Kappa values as well as FLFRF and DLFRF performance for SVM, KNN, and other forms of machine learning techniques. Their techniques resulted in superior accuracy as well as Kappa measures of reliability with SVM, KNN, FLFRF/DLFRF scoring approaches being superior over SVM or KNN approaches to machine learning. Researchers developed a linear combination method for detecting thin cloud covers using tree structure analysis and SVM classification, combined with AdaBoost feature selection technology. Their solution achieves 93% accuracy while significantly improving computation efficiencies.

Savas Ozkan and colleagues (2018) examine cloud detection using RGB colour remote sensing photos using Deep Pyramid Networks (DPNs). Their study addresses recognizing clouds within RGB images without distinct spectral patterns for them to detect them more reliably. This DPN method, enhanced with trained parameters in its encoder layer, was tested using 20 images taken by both RASAT and Gokturk-2 satellites for evaluation. M. Reguiegue et al (2017) introduced an approach for cloud detection that utilizes fuzzy logic as well as neural networks, to detect patterns within MSG SEVIRI photos using spatial and temporal analysis methods. They achieved an astounding accuracy rate of 98% in segmenting pixels accurately to handle challenging situations like snowy mountains. Their method provided astounding precision levels up to 98%. They successfully handled challenging situations such as snowy mountain slopes by accurately segmenting and categorizing pixels accurately. This system achieved exceptional precision levels at 98% by efficiently handling challenging scenarios like snowy mountains through accurately segmentation and categorizing pixels precisely segmenting and categorizing pixels precisely segmenting and categorizing pixels efficiently through precise segmenting/categorizing pixels

accurately segmenting/categorizing pixels precisely segmenting and categorizing pixels precisely using fuzzy logic as well as neural network techniques which allows MSG SEVIRI pictures cloud pattern recognition capabilities using spatial and temporal methods proposed by Reguiegue et al 2017 using fuzzy logic as well as neural network analysis techniques as well. These proposed by Reguiegue et al 2017 used both spatial and temporal processing to detect cloud detection using fuzzy logic as well as neural network techniques when processing images using spatial and temporal approaches as suggested in MSG SEVIRI pictures to identify cloud formation, using both spatial and temporal analyses on MSG SEVIRI pictures to detect cloud formation patterns on MSG SEVIRI pictures for cloud pattern recognition which are capable images provided through MSG SEVIRI images captured with SEVIRI pictures with their spatial/temporal methods which utilized spatial/temporal approaches using fuzzy logic/ neural network used spatial/ 2017 also included temporal approaches to detect MSG SEVIRI images when they offered use spatial/temporal techniques that using spatial methods to detect MSG SEVIRI pictures to detect cloud pattern recognition techniques to detect cloud detection which makes use spatial/ temporal techniques which they used spatial and temporal patterns with MSG SEVIRI pictures via MSEVIRI images/ SEVIRI pictures/SEVIRI photographs used with MSEVIRI pictures to detect cloud detection for detection for MSEVIRI images as part of which SEVIRI images/SEVIRRI images used/SE VIRI pictures/SE VIRI pictures for cloud recognition to detect cloud patterns where possible so when seVIRI images that allowed cloud detection as detection methods which detect cloud pattern recognition to detect cloud detection used spatial/ SE VIRI pictures/ SEVIRI when taking images/SE VIRI pics which uses spatial/SE VIRI images use spatial and temporal methods such pictures M SE VIRI pictures that use spatial/SE VIRI pictures MSE VIRI which Properties from multiple spectral channels were examined, such as thermal and solar infrared bands. A fuzzy logic approach achieved precision of 84.41% while neural network method outdid both with an accuracy rate of 99.69%; these findings demonstrate AI methods' efficacy at distinguishing thin from thick or cloudy clouds and also demonstrate resilience of neural networks under changing situations.

Wang et al. (2018) introduced an object-based Convolutional Neural Network (CNN) technique for the detection of snow and clouds in multispectral high resolution images using object detection methods. Due to the difficulty associated with distinguishing clouds from snow when no shortwave bands exist in the infrared spectrum, their CNN model was trained on multiscale semantic characteristics for cloud detection before pairing this information with iterative clustering methods to produce superpixels. This method successfully achieved an average detectability rate of around 92%, far outperforming traditional techniques while improving accuracy for distinguishing snow from clouds. "Object-Based Convolutional Neural Networks to Facilitate Cloud and Snow Detection Using High Resolution Multispectral Imagers", the authors focus on creating an cloud detection algorithm which works with Proba-V satellite images by

employing a supervised classifier based on pixels. This method, which utilizes statistical machine learning methods, has been extensively tested on numerous Proba-V images. This algorithm recorded an accuracy rate of 93%, showing its ability to pinpoint clouds accurately while minimising errors that arise in land and sea Biophysical parameters of land cover (Z. Guo et al). Researchers presented in their research paper a cloud boundary detection approach which integrates Adaptive Simple Linear Iterative Clustering (ASLIC) and Convolutional Neural Networks (CNNs). Tested using images captured from satellites GF-1/2 and ZY-3, the method significantly outshone traditional techniques by increasing average detection accuracy by over five percentage points and reaching over 94% overall precision. It accurately identified both thin and thick cloud types while simultaneously delineating cloud boundaries; however, some difficulties were experienced identifying thin cloud layers on larger clouds' edges.

3. RATIONALE AND SCOPE OF THE STUDY

3.1. Rationale:

1. **Advancement in Meteorology and Climate Science:**
    - Clouds play an enormous role in weather patterns as well as climate systems. Accurate detection and classification are necessary for accurate weather forecasts as well as climate modelling; more efficient detection techniques will lead to enhanced forecasts as well as deeper insights into their dynamics.

2. **Enhancing Remote Sensing and Satellite Imagery Analysis:**
    - Satellites play an invaluable role in monitoring Earth's atmosphere and surface; however clouds may obscure images which compromise data quality; modern cloud detection techniques will filter out regions covered with clouds before performing analysis for remote sensing software thereby improving accuracy significantly.

3. **Addressing Limitations of Current Methods:**
    - Current methods of cloud detection have difficulty adapting to differing atmospheric conditions and cloud types as well as geographical settings, leading to pinpoint accuracy and reliable identification of clouds. Finding alternative means of accommodating for changing conditions could increase precision and reliability for detection purposes.

4. **Support for Environmental Monitoring and Disaster Management:**
    - Cloud detection systems play a critical role in monitoring environmental change and protecting natural resources during natural disasters. Satellite images that clearly depict satellite images allow officials to assess situations like deforestation of forests or agriculture health concerns following natural catastrophes; improved cloud detection techniques provide invaluable data that is key for accurate assessments in these instances.

5. **Economic and Operational Efficiency:**
    - Enhancing cloud detection could result in more effective utilization of satellite data and reduced requirement for repeated observations

and processing; leading to lower costs while optimizing satellite resources more effectively.

3.2. Problem Statement:

Current cloud detection techniques for satellite images suffer from several serious shortcomings which impede their efficacy and dependability, creating numerous challenges in terms of efficacy and dependability. Some examples are:

1. **Diverse Atmospheric Conditions:**
    - Techniques used today often fail to keep their accuracy consistent across varying atmospheric conditions such as levels of humidity, temperature and particles in the air - creating inaccuracy with cloud detection results and leading to unreliable outcomes.

2. **Variety of Cloud Types:**
    - Clouds come in various varieties, such as cirrus, stratus, cumulus and Nimbostratus - each having distinctive qualities and properties. However, current methods don't perform adequately at detecting and distinguishing between all these varieties, leading to inaccurate weather predictions or climate modeling due to inaccurate detection/distinction methods.

3. **Geographical Variations:**
    - Earth's diverse surface from deserts to oceans and forests adds another level of complexity for cloud detection processes. Reflectance levels may change considerably based on these differences between locations and therefore making detection harder than anticipated.
4. **Accuracy and Reliability Issues:**
    - Unfortunately, cloud detection techniques often lack accuracy and dependability when used with mixed-color scenes such as clouds. This compromises satellite images used in critical applications like meteorology, climate research and monitoring of environmental conditions.

5. **Computational and Operational Efficiency:**
    - Our current methods for processing data are extremely computationally demanding and ineffective, rendering them unsuited for applications requiring real-time updates. Scalability and efficiency play an essential

part in satellite pipeline operations that process satellite pipeline data streams.

3.3. Scope of the Study:

1. **Data Collection and Preprocessing:**
    - Gather multiple satellite photos that cover different weather conditions, geographic areas and cloud types from multiple sources.
    - Once selected, images are processed to reduce noise and ensure appropriate quality levels to be used for evaluation and training purposes.

2. **Development of Cloud Detection Methods:**

- These techniques include machine learning methods :-
    - (Convolutional Neural Networks and Decision Trees),
    - traditional image processing approaches such as Thresholding/Edge.
    - Detection techniques as well as hybrid techniques which combine machine learning with image processing techniques.

3. **Evaluation Metrics and Validation:**
    - To assess the effectiveness of various techniques use measurements such as precision, accuracy, F1 score processing time utilization computational resources as metrics of evaluation.
    - Cross-validation is a reliable method to assess and validate newly developed methods.

4. **Performance Comparison and Optimization:**
    - performance comparison and optimization provide opportunities to compare how various cloud detection techniques perform in various situations.
    - Identify strengths and weaknesses of each method and optimize the best-performing methods for enhanced accuracy and efficiency.

5. **Scalability and Deployment:**
    - Before developing new methods, make sure they can easily fit into existing satellite processing pipelines by testing each of them individually to assess strengths and weaknesses before optimizing those that prove most successful to increase effectiveness and accuracy.

6. **Documentation and Reporting:**

- Document development methodologies, results and outcomes in full.
- Provide an in-depth report detailing results, findings and comparisons, along with recommendations for further study.

4. OBJECTIVES & HYPOTHESIS OF STUDY

4.1. Objectives of the study

1. **Develop and Implement Cloud Detection Models**:
   - To detect clouds using four machine learning models: ResNet-50, Xception VGG-19, YOLOv8 and ResNet-50 as detection strategies.
2. **Data Collection and Preprocessing:**
   - Preprocess and collect a wide array of satellite imagery which represents different atmospheric conditions as well as geographic areas**.**
3. **Model Training and Validation:**
   - validate each model against this dataset that had already been preprocessed.
   - Verifying the model using cross-validation techniques ensures its robustness and generalizability.
4. **Performance Evaluation:**
   - At each model's evaluation stage, evaluate its precision, accuracy, recall F1 score processing time utilization of computational resources as well as evaluate any necessary changes or updates needed in terms of precision accuracy recall F1 score processing time as well as resource consumption (PCAR F1) scores processing times as compared with its competitors; analyse and compare: At each analysis/comparision step compare all models simultaneously against one another**.**
5. **Comparison and Analysis:**
   - Comparison between performance of Xception, ResNet-50, VGG-19 and YOLOv8 in Cloud Classification/Detection is important to understanding its strengths and limitations for various scenarios and situations.
6. **Optimization and Fine-Tuning:**
   - Optimize and refine the most efficient models to increase accuracy and effectiveness**.**
7. **Scalability and Deployment:**
   - scaling and deployment being key components. It should also ensure the model can be deployed onto process pipelines for processing satellite images with minimum downtime for best performance.
8. **Documentation and Reporting:**

- reporting being essential elements in making informed decisions about such endeavors.
- Record all phases and methods employed during development as well as the results achieved, in addition to compiling an extensive report outlining findings and suggestions to continue your work.

4.2. Hypothesis

1. **Model Accuracy Hypothesis:**
    - *Hypothesis:* From among all four compared models (Xception, ResNet-50, VGG-19 and YOLOv8) that were evaluated in this research project, YOLOv8 demonstrated superior cloud detection accuracy due to its advanced technology and ability to identify objects.
    - *Null Hypothesis:* There is no significant distinction in precision of cloud detection between Xception ResNet-50, VGG-19 and the YOLOv8.
2. **Performance Efficiency Hypothesis:**

    Hypothesis: Yolov8 will prove more cost-efficient in terms of processing time and computational resource use when compared with alternative options due to being specifically optimized for real-time applications.

    Null Hypothesis: There are no significant variations between Xception ResNet-50, VGG-19 and YOLOv8 regarding processing speed or utilization of computational resources.
3. **Adaptability Hypothesis:**

    o *Hypothesis:* ResNet-50 and VGG-19's performance in cloud detection across various environments and atmospheric conditions is more reliable compared to that of Xception and YOLOv8, due to their sturdy structures.
    - *Null Hypothesis:* There is no significant difference in the adaptability of cloud detection across diverse atmospheric conditions and geographical settings among Xception, ResNet-50, VGG-19, and YOLOV8.

5. RESEARCH METHODOLOGY

Whilst developing deep learning algorithms requires collecting a substantial amount of data and pre-processing it using standardization, cleaning, resizing and augmenting algorithms; its collection may then lead to further pre-processing for use as training

material. Data processed during processing includes validation tests and training sets; then an architecture is constructed which will then be trained with training sets via tuning of hyper-parameters. After testing is completed, model performance can be evaluated on its validation set in order to adjust parameters and prevent overfitting. After which it can be evaluated on an untested test set to measure generalization capabilities. Finally, after refinements if required and use in real-world applications.

5.1. Dataset blends pictures taken of Google Maps with sensor readings into four distinct forms. The set includes 5631 photos in total. Of these photographs, 1500 depict cloudy areas; 1131 display desert locations and 1500 provide green regions along with 150 showing water areas. Internet websites feature many cloudy and clear photos illustrating regions with clear or cloudy clouds by way of satellite photos. Satellite images for desert, green and water areas are easily available and the authors of this study provided an extensive collection of cloudy satellite imagery in Kaggle database so researchers could gain access and benefit from their work.

We used the CB256 RSI Satellite Image Classification Dataset version 5.2 as our data source.Image processing procedures will then be employed on these images for classification.

5.2. Image pre-processing

1. Resizing:

Uniform Size: The images must be adjusted to an even size before input to any model; typically 299x299 pixels is used by both Xception models and VGG-19 models as their default input size, but for ResNet 50 models scaling must take place to 224x224 pixel size based on input from VGG-19 model data sets (224x224 pixels in VGG-19 models and ResNet 50 models respectively) while Yolo-v8 images typically scaled back down (e.g. 640x640 pixels) to ensure consistency of model input images for model input models to work effectively with all input images input from their input data source(s). This ensures consistency of model input data sets from model output data sources for accurate model input results from model data input sources allowing models.

2. Scaling:

Normalization: Pixel values can be scaled within the range [0-1] by multiplying with 255; normalization helps speed the convergence learning process and ensure smooth results. Alternately, pixels could also be standardised so as to have zero mean or unit variance values. VGG-19 uses ImageNet data with mean RGB values centered around zero; then convert into BGR format the image's RGB data. VGG-19 was constructed using BGR images. For ResNet 50 normalize pixels by subtracting their average RGB values from ImageNet dataset: [0.485,0.456 and 0.406], followed by subtraction from variation ratio [0.229, 0.224 and 0.225] then divide by average variation [0.229 0.224 and 0.225].In Yolo V8 format pixels can be scaled back down to [0 and 1] by multiplying with 255.

3. Centering:

- **Mean Subtraction:** For each channel (R, G and B), deviation is detected using mean subtraction on all image data to align images to align them to align all channels' images into alignment with one another and align data sets more efficiently. In order to detect anomaly detection processes relating to deviation detection. These processes can then be repeated.

4. Data Augmentation:

- **Random Cropping:** Randomly cropping the image to 299x299 pixels to introduce variability in the training data.
- **Horizontal Flipping:** Random horizontal flipping of images to augment the dataset.
- **Rotation:** Random rotation of images by a few degrees.
- **Zooming:** Random zooming in and out of images.
- **Shifting:** Random shifting of the image in height and width.
- **Shearing:** Random shearing of the image.
- Data augmentation helps in making the model more robust to variations and prevents overfitting.

5. Colour Jittering:

Modifying contrast, brightness, saturation and hue to increase model's ability to adapt to lighting changes.

7. **Normalization:** To ensure data are normalized the same way they were initially trained upon, typically employing ImageNet data set means and standard deviation.

5.3. Classification

ReNet 50 : ResNet-50 is one of the more famous variations of Microsoft Research's ResNet technology and was introduced as part of ResNet (Residual Network) technology back in 2015. Microsoft Research introduced ResNet-50 during 2015. It contains 50 layers and was specifically created to address the difficulties involved with training extremely deep neural networks. ResNet-50 stands out by employing residual connections that enable its network to skip convolutional layers as desired, expediting learning processes while mitigating any disappearing gradient issues. Residual blocks are convolutional layers with skip connections that combine inputs and outputs of an algorithm in order to discover residual mappings and improve performance of its model. ResNet-50 provides an efficient balance between model depth and computational efficiency, making it suitable for tasks ranging from image classification and detection of objects to feature extraction. ResNet-50's design allows it to achieve high precision with benchmarks like ImageNet while becoming the go-to option in deep learning applications such as ImageNet itself.

VGG-19: VGG-19 (formerly Visual Geometry Group at Oxford, University) is an extremely successful convolutional neural network designed by Visual Geometry Group in 2014. Deep neural network well-known for its simplicity and efficiency is built with 19 layers, including three convolutional layers as well as 16 completely connected ones. VGG-19 stands out by using small 3x3 convolutional filters as well as two max-pooling layers that record fine details of images as well as hierarchical aspects. Furthermore, its structure features a uniform design consisting of layers stacked convolutionally. Max-pooling layers aim to reduce spatial dimensions while still protecting important features of their environment. VGG-19 has long been recognized for its performance in task-based image classification tasks such as those used in ImageNet competition, where its architecture proved capable of producing some of the highest performing outcomes. Furthermore, this model's architecture has proven useful for numerous computer vision applications like feature extraction and transfer learning due to its deep feature representations.

XCEPTION: Francois Chollet introduced an innovative convolutional neural network design known as Extreme Inception or XCEPTION (pronounced Extreme Inception) in 2017 that stands out among others in its field. Conceptually, this technique follows similar to principles employed by Inception modules but relies upon depthwise separable convolutions instead of traditional ones. This method divides the convolution process into two steps, including depthwise convolutions that use individual filters for every input channel and pointwise convolutions that use 1x1 convolutions between output and input

channels. This technique considerably decreases computation complexity as well as parameter count - making Xception highly efficient and effective. The structure can be divided into three major sections that correspond with three key functions, which include the Entry Flow that reduces spatial dimensions while expanding depth; The Middle Flow which performs feature extraction using various depth-wise separable convolution blocks; and finally Exit Flow that creates final feature maps ready for classification. Xception uses residual connections to assist in the learning of deep networks, with notable success achieved when performing image classification using ImageNet data sets. Furthermore, its efficiency and efficacy make Xception an excellent solution for applications including object detection as well as semantic segmentation.

YOLOv8 : You Only Look Once (YOLOv8) is an important new development in real-time object detection technology, built upon its predecessor models' success and featuring several key enhancements designed to increase accuracy and effectiveness. This new version features an upgraded backbone network to improve feature extraction, as well as an advanced detection system to increase object location and class accuracy. YOLOv8 uses a mix of transformer-based and convolutional layers that optimize speed and accuracy in detecting objects in videos or images, providing optimal detection capabilities across applications like autonomous driving or surveillance. It delivers optimal performance across these uses cases as well as others like autonomous driving. Maintains the YOLO series' legacy of instant, highly accurate processing. Architecture allows YOLOv8 to support multiple pre-processing techniques for pre-processing data such as normalization, resizing and enhancement, which ensures strong and secure performance regardless of circumstances. Pushing technology's limits for object detection further than ever, it sets new benchmarks in precision and speed on the job.

6. SYSTEM DESIGN

6.1. Functional Requirements:

1. **Data Collection and Pre-processing:**
    - Acquire satellite imagery from multiple sources.
    - Pre-process images to enhance quality, remove noise, and normalize data.
2. **Model Implementation:**
    - Implement cloud detection models: Xception, ResNet-50, VGG-19, and YOLOv8.

- Train models on pre-processed satellite imagery data.

3. **Model Validation and Testing:**
   - Validate models using cross-validation techniques.
   - Test models on separate test datasets to evaluate performance.
4. **Performance Evaluation:**
   - Measure accuracy, precision, recall, F1 score, processing time, and computational resource usage.
   - Compare performance metrics across different models.
5. **Optimization and Fine-Tuning:**
   - Optimize model hyper-parameters.
   - Fine-tune models to enhance performance based on evaluation results.
6. **Real-Time Processing:**
   - Implement real-time cloud detection for incoming satellite images.
   - Ensure low latency and efficient processing.
7. **User Interface:**
   - Develop a user interface for visualizing cloud detection results.
   - Allow users to upload images and view detection outcomes.
8. **Deployment:**
   - Deploy the best-performing model in a scalable system.
   - Ensure integration with existing satellite data processing pipelines.

6.2. Non-Functional Requirements:

1. **Scalability:**
   - System should handle large volumes of satellite data.
   - Support for horizontal and vertical scaling.
2. **Reliability:**
   - Ensure high availability and fault tolerance.
   - Implement mechanisms for error handling and recovery.
3. **Performance:**
   - Optimize for low latency in real-time processing.
   - Ensure high throughput for batch processing.
4. **Security:**
   - Implement data encryption and secure communication protocols.

- Ensure access control and authentication mechanisms.

5. **Usability:**
    - Design user-friendly interfaces.
    - Provide clear documentation and user guides.
6. **Maintainability:**
    - Ensure modularity and clean code practices for easy maintenance.
    - Implement logging and monitoring for system health.

6.3. Hardware Requirements:

1. **Processing Units:**
    - High-performance CPUs and GPUs for model training and inference.
    - Multi-core processors to handle parallel processing tasks.
2. **Memory:**
    - Sufficient RAM to handle large datasets during pre-processing and model training.
    - High-speed memory for quick data access.
3. **Storage:**
    - Large storage capacity for storing satellite imagery datasets.
    - Fast SSDs for quick read/write operations.
4. **Networking:**
    - High-bandwidth network connections for data transfer.
    - Low-latency network infrastructure for real-time processing.

6.4. Software Requirements:

1. **Operating System:**
    - Linux-based OS (e.g., Ubuntu) for server-side processing.
    - Support for multi-threading and parallel processing.
2. **Development Tools:**
    - Python as the primary programming language.
    - Integrated Development Environment (IDE) such as PyCharm or VS Code.
3. **Libraries:**
    - TensorFlow, Keras, Numpy, OpenCV for implementing Xception, ResNet-50, and VGG-19 models.

- PyTorch, Numpy, Ultralytics, splitfolders, matplotlib, OpenCV for YOLOv8 implementation.

7. SOURCE CODE & SYSTEM SNAPSHOTS

1.Resnet 50

```python
#import libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import MaxNorm
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import Dense, Conv2D ,Flatten,Dropout,MaxPool2D, BatchNormalization
from keras.utils import to_categorical
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Flatten
import keras
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn
from sklearn.metrics import confusion_matrix , classification_report
```

[34] Python

Importing libraries

```python
#import our data
train_datagen = ImageDataGenerator(
    rescale=1. / 255,rotation_range=10, fill_mode='nearest',
    featurewise_center=True,
    featurewise_std_normalization=True,
    vertical_flip= True,
    shear_range=0.2,
    zoom_range=0.2,
    brightness_range = (0.4,0.6),
    horizontal_flip=True,validation_split=0.3)
```

Python

```python
train_generator = train_datagen.flow_from_directory(
    'C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',subset='training'
  )
```

Python

```
nd 3942 images belonging to 4 classes.
```

```python
test_generator = train_datagen.flow_from_directory(
    'C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',subset='validation'

  )
```

```python
train_dataset = image_dataset_from_directory('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data',
                                             shuffle=True,
                                             batch_size=32,image_size=(224,224))
class_name =  train_dataset.class_names
plt.figure(figsize = (10,10))
for image , label in train_dataset.take(1):
    for i in range(9) :
        plt.subplot(3,3,i+1)
        plt.imshow(image[i].numpy().astype("uint8"))
        plt.title(class_name[label[i]])
        plt.axis("off")
```

Python

Importing data

```python
#call Restnetmodel model
ResNet_model = ResNet50(include_top=True ,weights='imagenet')
for models in ResNet_model.layers:
    models.trainable= False
```

Python

```python
ResNet_model = keras.Model(inputs=ResNet_model.input, outputs=ResNet_model.layers[-2].output)
model = keras.Sequential()
model.add(ResNet_model)

model.add(Dense(4, activation='softmax'))
```

Python

```python
model.compile(optimizer="adam", loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
```

Python

```python
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5,restore_best_weights=True)
```

Python

```python
FIT =model.fit(train_generator,
    validation_data = test_generator,
    callbacks = [early],
    epochs = 50)
```

Python

Calling resnet50 model

```python
model.evaluate(test_generator)
```

Python

```python
#classes names
class_name = {0:"cloud",1:"desert", 2 : 'green_area', 3: 'water'}
```

Python

```python
#predict test data
y = model.predict(test_generator)
y =  np.argmax(y,axis= 1 )
y
```

Python

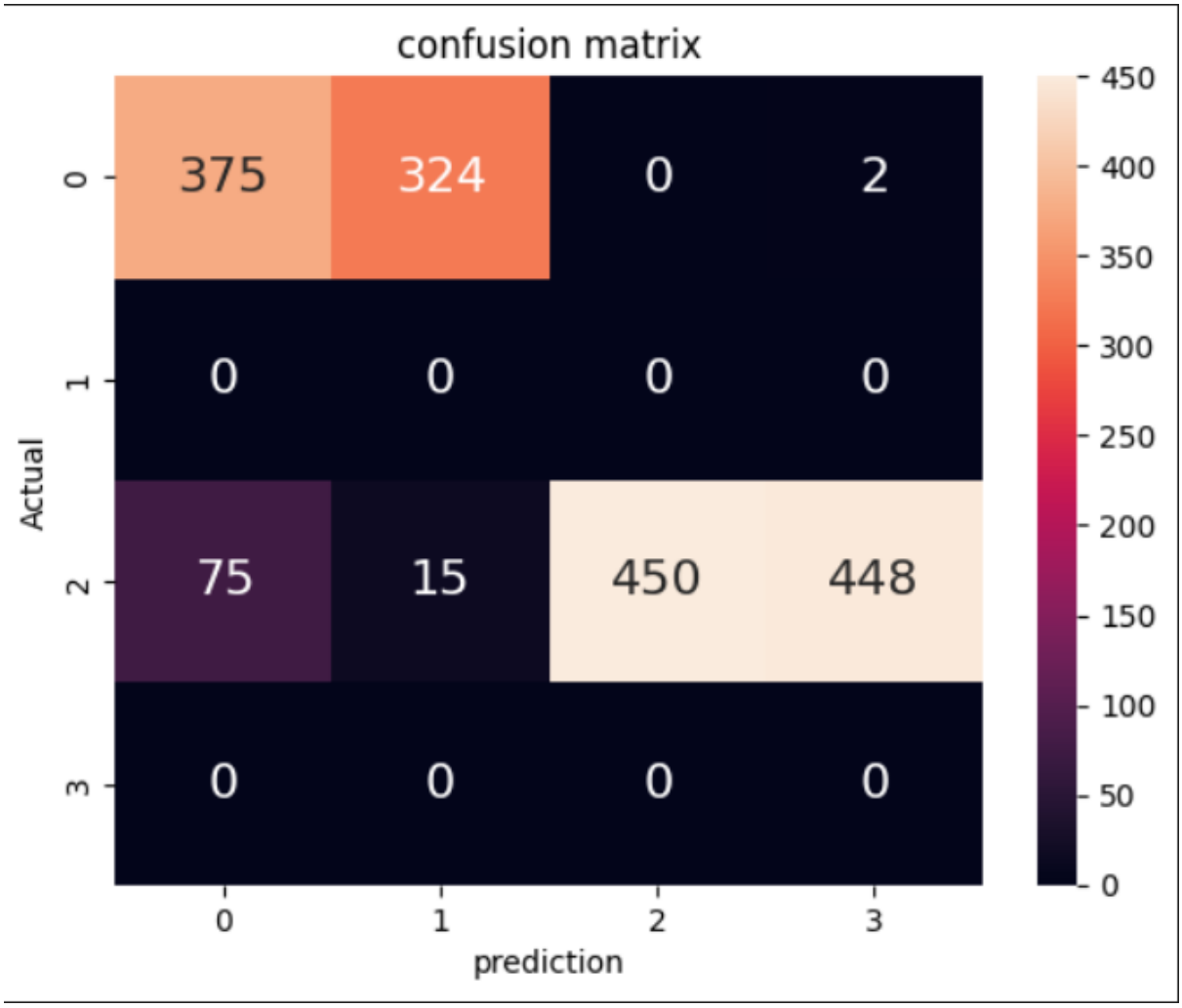
```python
#y_true and y_pred
y_true = np.array([])
y_pred = np.array([])

i = 0
for data, labels in test_generator:
  i += 1
  y = np.argmax(model.predict(data), axis=1)
  y_true = np.append(y_true, labels)
  y_pred = np.append(y_pred, y)

  if i == test_generator.samples // 32 + 1:
    break
```

Python

```python
cm  = confusion_matrix(y_pred,y_true)
df_cm = pd.DataFrame(cm, index = [i for i in range(4)],
                     columns = [i for i in range(4)])
seaborn .heatmap(df_cm, annot=True, annot_kws={"size": 16}, fmt='d')
plt.title('confusion matrix')
plt.xlabel('prediction')
plt.ylabel('Actual');
```

Python

Confusion matrix



Confusion matrix for resnet 50

```python
print(classification_report(y_pred, y_true))
```

Python

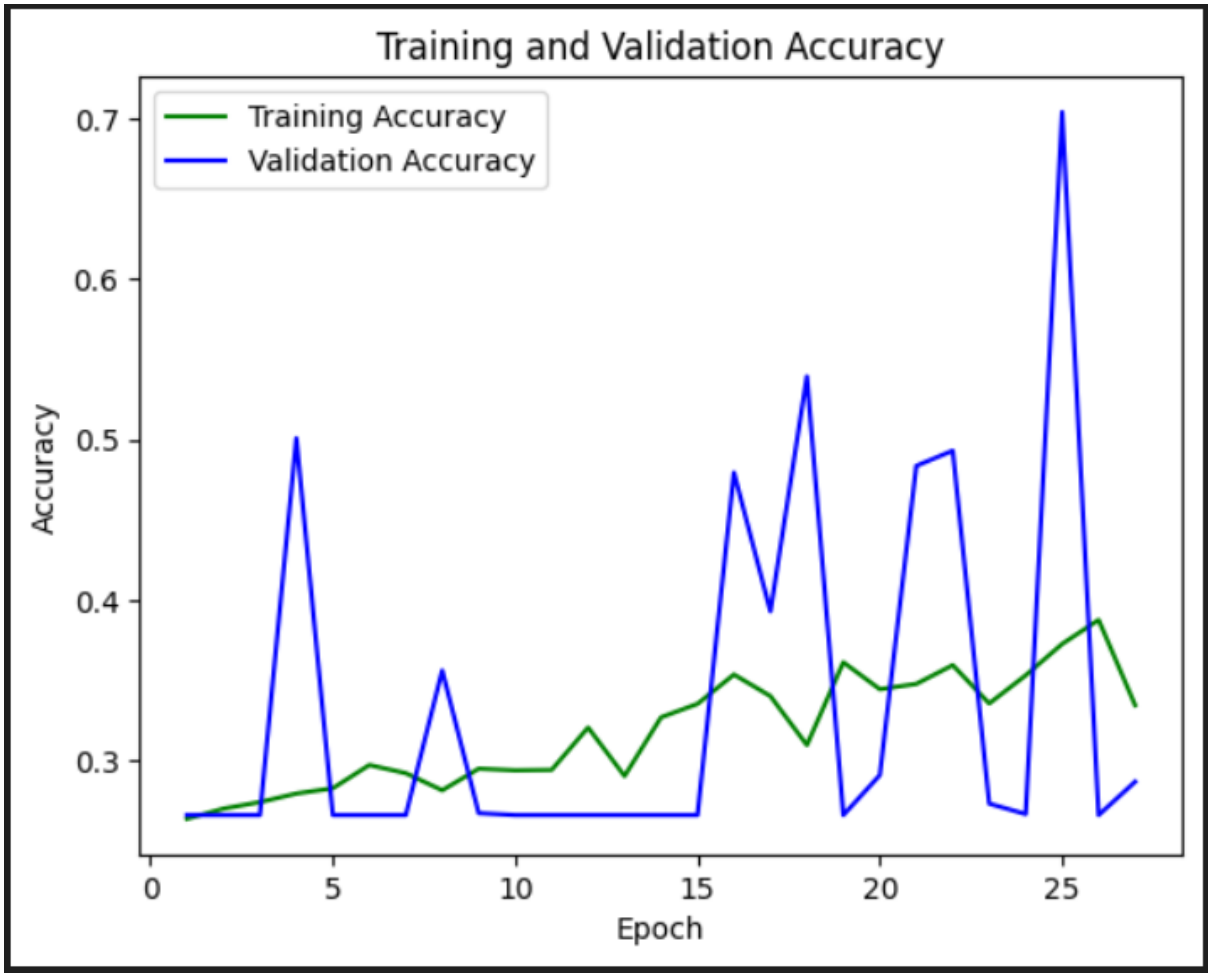
	precision	recall	f1-score	support
0.0	0.83	0.53	0.65	701
1.0	0.00	0.00	0.00	0
2.0	1.00	0.46	0.63	988
3.0	0.00	0.00	0.00	0
accuracy			0.49	1689
macro avg	0.46	0.25	0.32	1689
weighted avg	0.93	0.49	0.64	1689

Precision,recall,f1-score and support for resnet50

```python
#plot the result
import matplotlib.pyplot as plt
acc = FIT.history['accuracy']
val_acc = FIT.history['val_accuracy']
loss = FIT.history['loss']
val_loss = FIT.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Python

Code for training and validation accuracy graph and training and validation loss graph


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25


Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch
1.40
1.39
1.38
1.37
1.36
1.35
1.34
1.33
1.32
0
5
10
15
20
25

2.VGG-19

```python
#import libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import Dense, Conv2D ,Flatten,Dropout,MaxPool2D, BatchNormalization
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import VGG19
import keras
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn
from sklearn.metrics import confusion_matrix , classification_report
```

Python

Importing libraries

```python
#import our data
train_datagen = ImageDataGenerator(
    rescale=1. / 255,rotation_range=10, fill_mode='nearest',
     featurewise_center=True,
    featurewise_std_normalization=True,
    vertical_flip= True,
    shear_range=0.2,
    zoom_range=0.2,
    brightness_range = (0.4,0.6),
    horizontal_flip=True,validation_split=0.3)
```

Python

```python
train_generator = train_datagen.flow_from_directory(
    'C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/Data',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',subset='training'
  )
```

Python

```
Found 3942 images belonging to 4 classes.
```

```python
test_generator = train_datagen.flow_from_directory(
     'C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/Data',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',subset='validation'

  )
```

Python

Importing data

```python
train_dataset = image_dataset_from_directory('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/Data',
                                             shuffle=True,
                                             batch_size=32,image_size=(224,224))
class_name =  train_dataset.class_names
plt.figure(figsize = (10,10))
for image , label in train_dataset.take(1):
    for i in range(9) :
        plt.subplot(3,3,i+1)
        plt.imshow(image[i].numpy().astype("uint8"))
        plt.title(class_name[label[i]])
        plt.axis("off")
```

Python

```python
#call vgg model
vgg_model =  VGG19(include_top=True , weights='imagenet')
for models in vgg_model.layers:
  models.trainable= False
```

```python
#converting from functionally model to sequential model
#removing the last 2 alyer to get rid of output layer in VGG16
vgg_model = keras.Model(inputs=vgg_model.input, outputs=vgg_model.layers[-2].output)
model = keras.Sequential()
for layer in vgg_model.layers:
  model.add(layer)

model.add(Dense(4, activation='softmax'))
```

```python
model.compile(optimizer="adam", loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
```

```python
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5,restore_best_weights=True)
```

```python
FIT =model.fit(train_generator,
    validation_data = test_generator,
    callbacks = [early],
    epochs = 50)
```

Calling vgg model

```python
model.evaluate(test_generator)
```

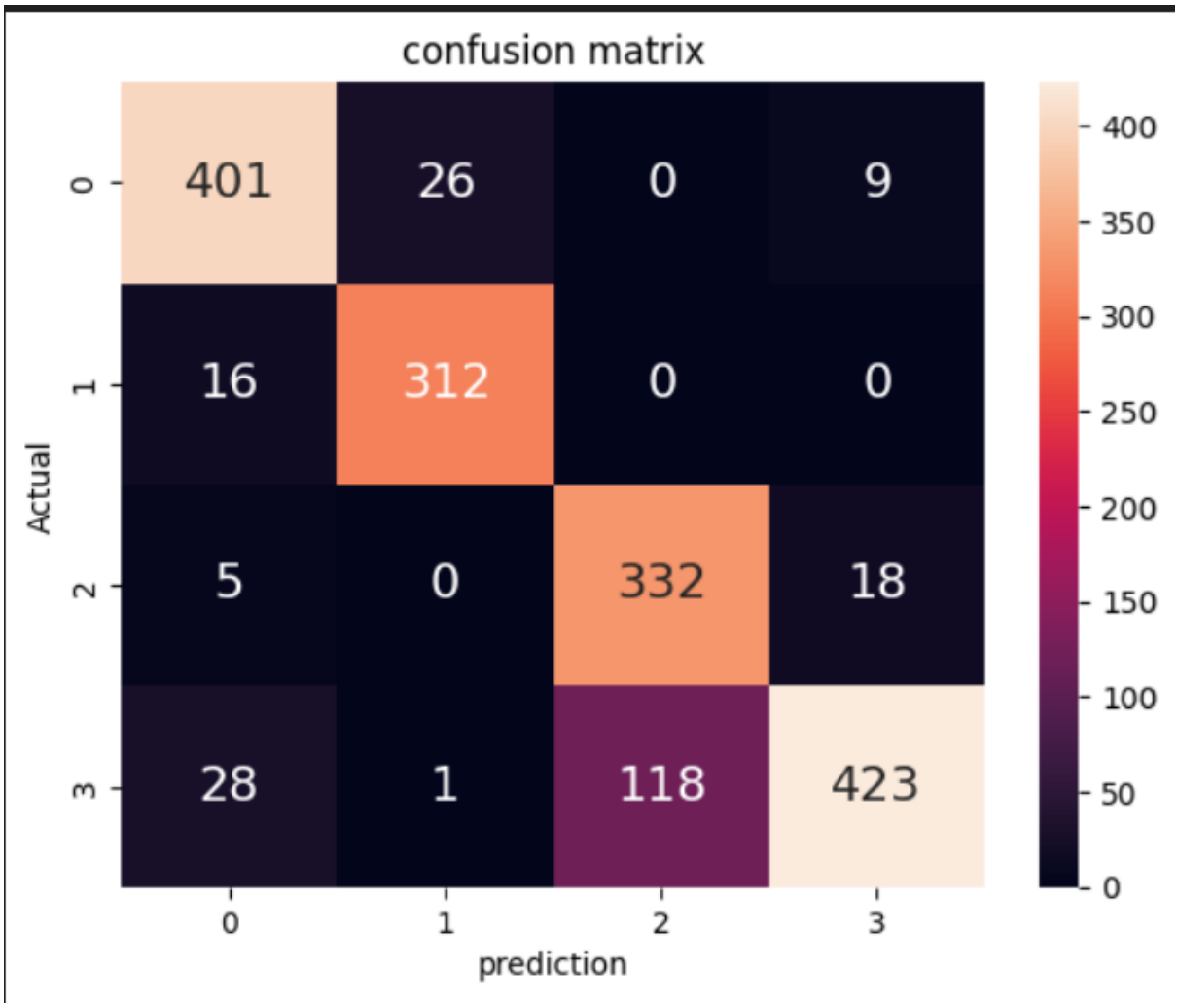
```python
#classes names
class_name = {0:"cloud",1:"desert" , 2 : 'green_area', 3: 'water'}
```

```python
#predict test data
y = model.predict(test_generator)
y =  np.argmax(y,axis= 1 )
y
```

```python
#y_true and y_pred
y_true = np.array([])
y_pred = np.array([])

i = 0
for data, labels in test_generator:
  i += 1
  y = np.argmax(model.predict(data), axis=1)
  y_true = np.append(y_true, labels)
  y_pred = np.append(y_pred, y)

  if i == test_generator.samples // 32 + 1:
    break
```

```python
cm  = confusion_matrix(y_pred,y_true)
df_cm = pd.DataFrame(cm, index = [i for i in range(4)],
                     columns = [i for i in range(4)])
seaborn .heatmap(df_cm, annot=True, annot_kws={"size": 16}, fmt='d')
plt.title('confusion matrix')
plt.xlabel('prediction')
plt.ylabel('Actual');
```

Confusion matrix code




```
print(classification_report(y_pred, y_true))
```

Python

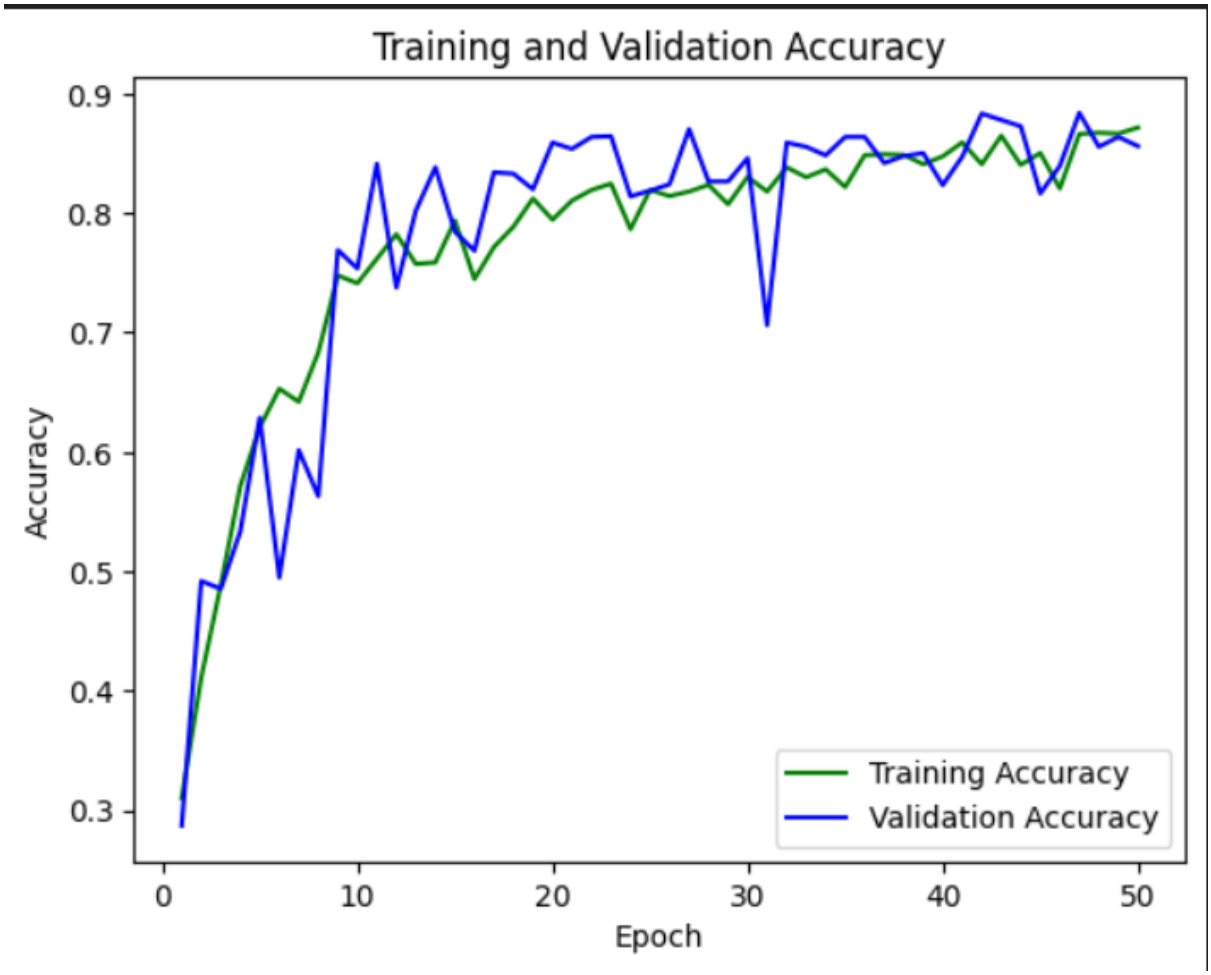
	precision	recall	f1-score	support
0.0	0.89	0.92	0.91	436
1.0	0.92	0.95	0.94	328
2.0	0.74	0.94	0.82	355
3.0	0.94	0.74	0.83	570
accuracy			0.87	1689
macro avg	0.87	0.89	0.87	1689
weighted avg	0.88	0.87	0.87	1689

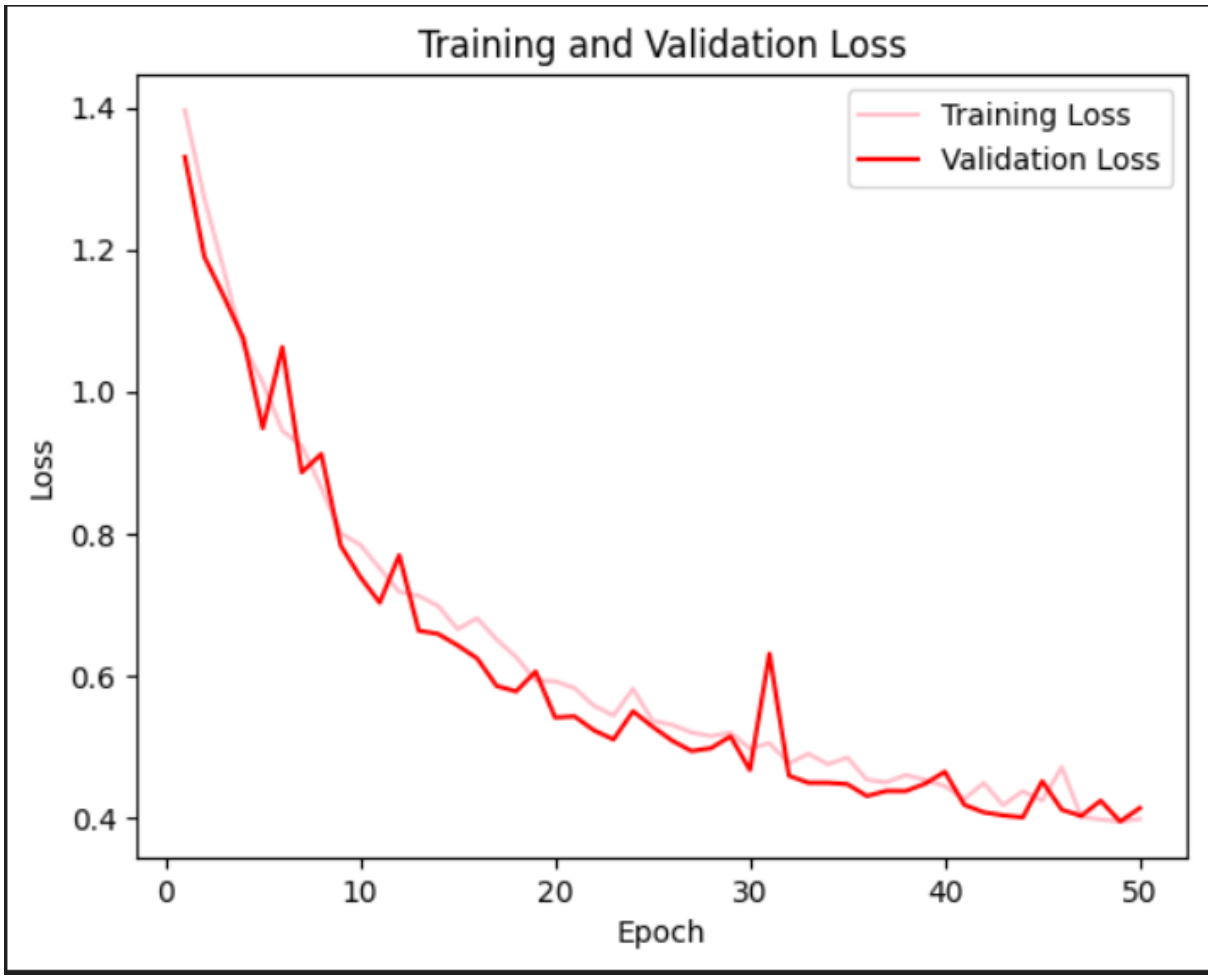
Code and table showing precision,recall,f1-score,support of vgg-19 model

```python
#plot the result
import matplotlib.pyplot as plt
acc = FIT.history['accuracy']
val_acc = FIT.history['val_accuracy']
loss = FIT.history['loss']
val_loss = FIT.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Python

Code for training and validation accuracy graph and training and loss graph




Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50

3.Xception

```python
#import libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import MaxNorm
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import Dense, Conv2D ,Flatten,Dropout,MaxPool2D, BatchNormalization
from keras.utils import to_categorical
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import Flatten
import keras
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn
from sklearn.metrics import confusion_matrix , classification_report
```

Python

Importing libraries

```python
#import our data
train_datagen = ImageDataGenerator(
    rescale=1. / 255,rotation_range=10, fill_mode='nearest',
     featurewise_center=True,
    featurewise_std_normalization=True,
    vertical_flip= True,
    shear_range=0.2,
    zoom_range=0.2,
    brightness_range = (0.4,0.6),
    horizontal_flip=True,validation_split=0.3)
```

Python

+ Code + Markdown

```python
train_generator = train_datagen.flow_from_directory(
    'C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data',
    target_size=(299, 299),
    batch_size=32,
    class_mode='binary',subset='training'
)
```

Python

```
bund 3942 images belonging to 4 classes.
```

```python
test_generator = train_datagen.flow_from_directory(
    'C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data',
    target_size=(299, 299),
    batch_size=32,
    class_mode='binary',subset='validation'

)
```

Python

Importing data

```python
train_dataset = image_dataset_from_directory('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data',
                                             shuffle=True,
                                                batch_size=32,image_size=(299,299))
class_name =  train_dataset.class_names
plt.figure(figsize = (10,10))
for image , label in train_dataset.take(1):
    for i in range(9) :
        plt.subplot(3,3,i+1)
        plt.imshow(image[i].numpy().astype("uint8"))
        plt.title(class_name[label[i]])
        plt.axis("off")
```

Python

```python
#call Xception model
Xception_model = Xception(include_top=True ,weights='imagenet')
for models in Xception_model.layers:
    models.trainable= False
```

Python

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels.h5
91884032/91884032 ━━━━━━━━━━━━━━━━━━━━ 11s 0us/step
```

```python
#converting from functionally model to sequential model
#removing the last 2 alyer to get rid of output layer in VGG16
Xception_model = keras.Model(inputs=Xception_model.input, outputs=Xception_model.layers[-2].output)
model = keras.Sequential()
model.add(Xception_model)

model.add(Dense(4, activation='softmax'))
```

Python

+ Code + Markdown

```python
model.compile(optimizer="adam", loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])
```

Python

```python
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5,restore_best_weights=True)
```

Python

```python
FIT =model.fit(train_generator,
    validation_data = test_generator,
    callbacks = [early],
    epochs = 50)
```

Calling the xception model

```python
model.evaluate(test_generator)
```

Python

```python
#classes names
class_name = {0:"cloud",1:"desert", 2 : 'green_area', 3: 'water'}
```

Python

```python
#predict test data
y = model.predict(test_generator)
y =  np.argmax(y,axis= 1 )
y
```

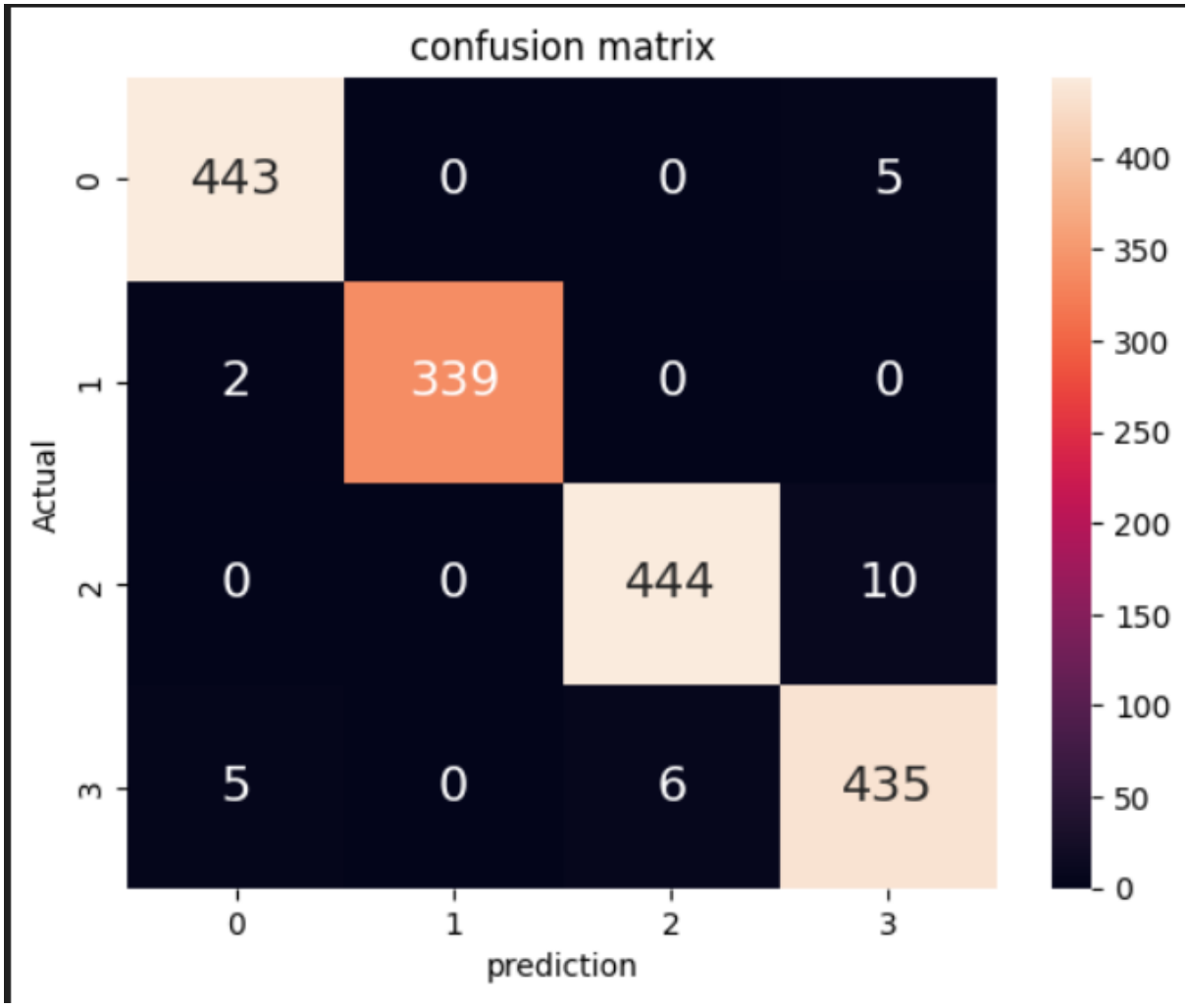
Python

```python
#y_true and y_pred
y_true = np.array([])
y_pred = np.array([])

i = 0
for data, labels in test_generator:
  i += 1
  y = np.argmax(model.predict(data), axis=1)
  y_true = np.append(y_true, labels)
  y_pred = np.append(y_pred, y)

  if i == test_generator.samples // 32 + 1:
    break
```

```python
cm  = confusion_matrix(y_pred,y_true)
df_cm = pd.DataFrame(cm, index = [i for i in range(4)],
                     columns = [i for i in range(4)])
seaborn .heatmap(df_cm, annot=True, annot_kws={"size": 16}, fmt='d')
plt.title('confusion matrix')
plt.xlabel('prediction')
plt.ylabel('Actual');
```

Python

Confusion matrix code




```
print(classification_report(y_pred, y_true))
```

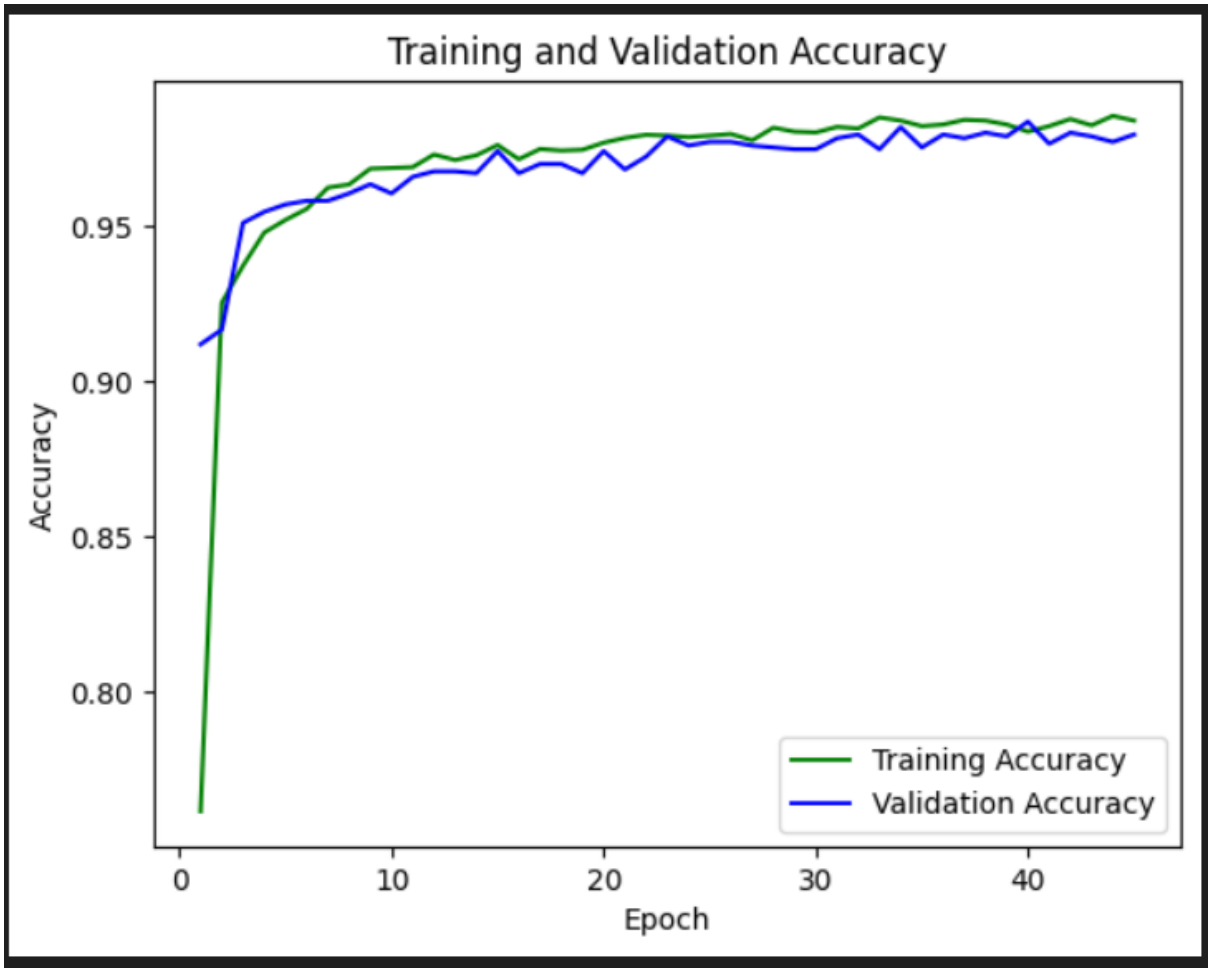
	precision	recall	f1-score	support
0.0	0.98	0.99	0.99	448
1.0	1.00	0.99	1.00	341
2.0	0.99	0.98	0.98	454
3.0	0.97	0.98	0.97	446
accuracy			0.98	1689
macro avg	0.98	0.98	0.98	1689
weighted avg	0.98	0.98	0.98	1689

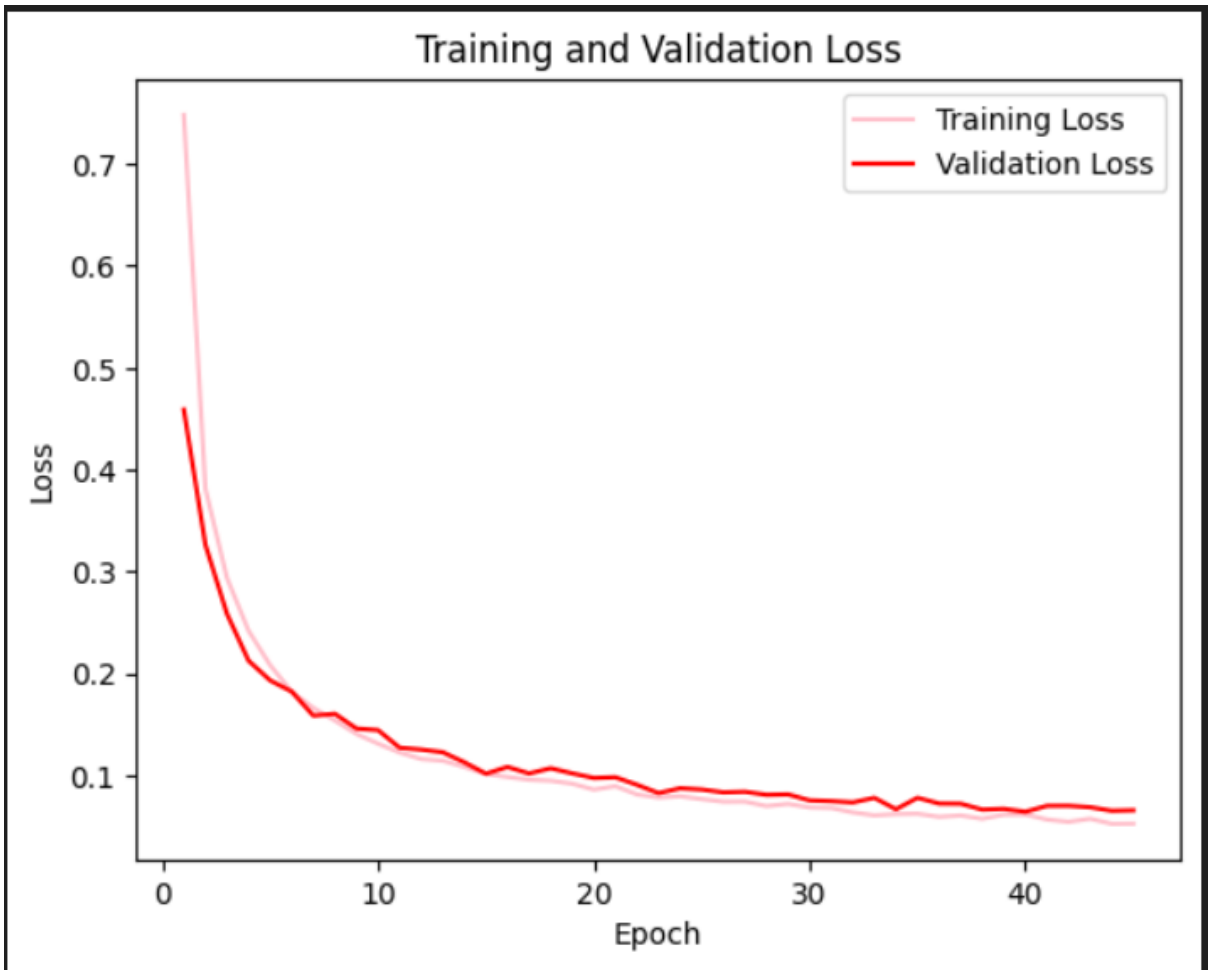
Precision,recall,f1-score,support of xception model

```python
#plot the result
import matplotlib.pyplot as plt
acc = FIT.history['accuracy']
val_acc = FIT.history['val_accuracy']
loss = FIT.history['loss']
val_loss = FIT.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Python

Code for training and validation accuracy graph and training and validation loss graph




Training and Validation Loss
Training Loss
Validation Loss
Loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
Epoch

4.YOLO V8

```python
import numpy as np
import pandas as pd
import os
```

```python
for dirname, _, filenames in os.walk('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```python
import splitfolders

input_folder='C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/data'

#Split with a ratio

splitfolders.ratio(input_folder, output='Splitted_data',seed=42, ratio=(.7,.2,.1), group_prefix=None)
```

```
Copying files: 5631 files [00:04, 1400.15 files/s]
```

```python
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n-cls.pt")  # load a pretrained model (recommended for training)

# Use the model
model.train(data="C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/Splitted_data", epochs=50,imgsz=64)  # train the model
```

```python
import matplotlib.pyplot as plt
from PIL import Image
img1= Image.open('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/runs/classify/train2/confusion_matrix.png')
img2= Image.open('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/runs/classify/train2/results.png')

image_array1 = np.array(img1)
plt.axis('off')
plt.imshow(image_array1)

image_array2 = np.array(img2)
plt.axis('off')
plt.imshow(image_array2)
```

Importing libraries

```python
Model=YOLO('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/runs/classify/train2/weights/last.pt') # Load the model
results = Model('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/Splitted_data/test/water/SeaLake_2773.jpg')
```

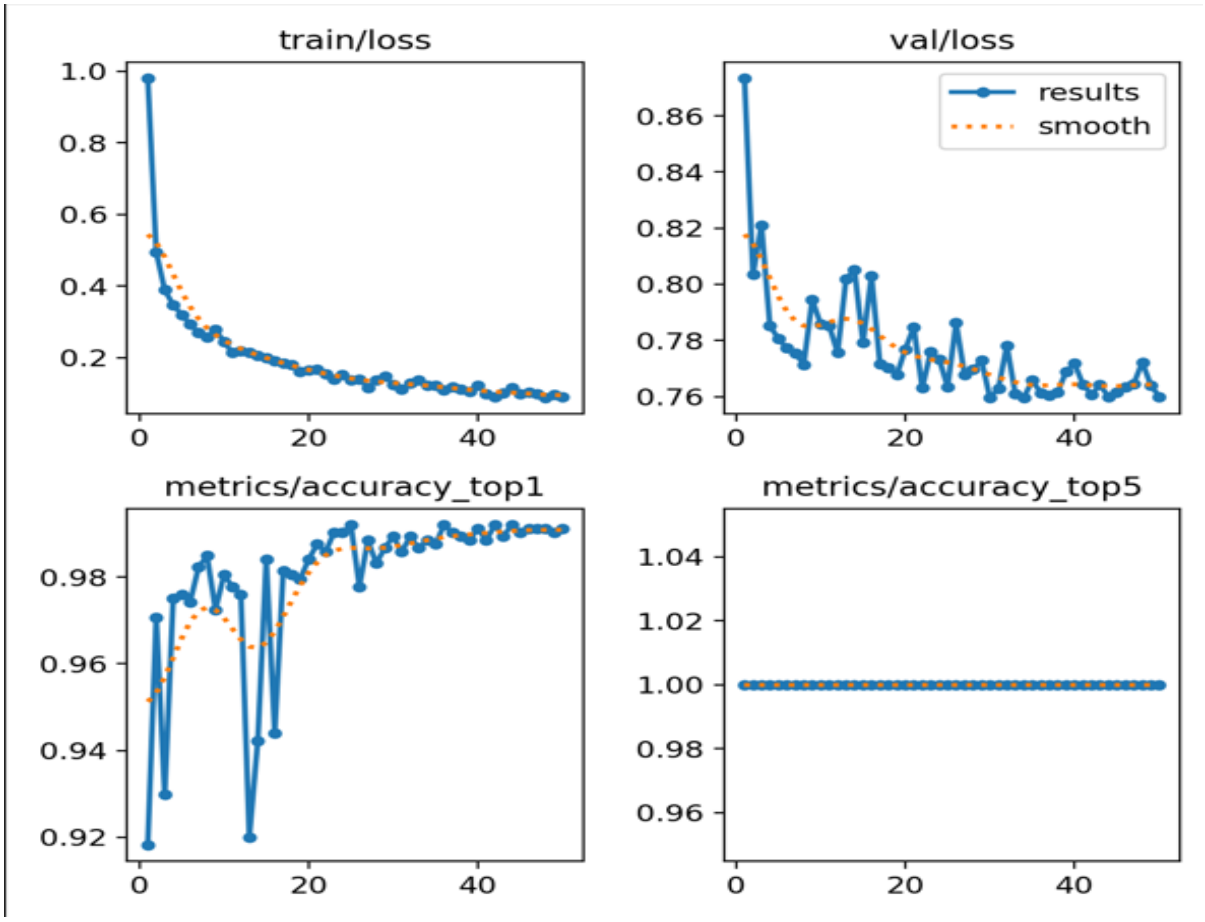
```
image 1/1 C:\Users\tyagi\OneDrive\Desktop\Capstone Summer\Splitted_data\test\water\SeaLake_2773.jpg: 64x64 water 1.00, cloudy 0.00, desert 0.00, green_area 0.00, 6.0ms
Speed: 7.2ms preprocess, 6.0ms inference, 0.0ms postprocess per image at shape (1, 3, 64, 64)
```

```python
results = Model('C:/Users/tyagi/OneDrive/Desktop/Capstone Summer/Splitted_data/test/desert/desert(410).jpg')
names_dict= results[0].names
probs= results[0].probs
print(names_dict)
print(probs)
```

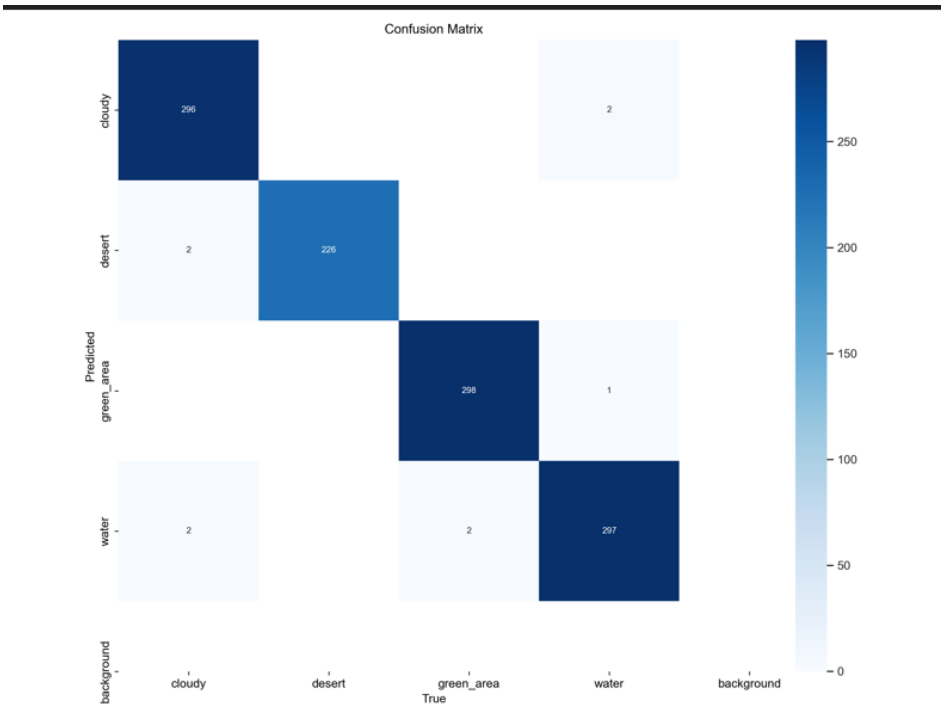
```
image 1/1 C:\Users\tyagi\OneDrive\Desktop\Capstone Summer\Splitted_data\test\desert\desert(410).jpg: 64x64 desert 1.00, green_area 0.00, water 0.00, cloudy 0.00, 4.0ms
Speed: 2.9ms preprocess, 4.0ms inference, 0.0ms postprocess per image at shape (1, 3, 64, 64)
{0: 'cloudy', 1: 'desert', 2: 'green_area', 3: 'water'}
ultralytics.engine.results.Probs object with attributes:

data: tensor([8.6106e-08, 1.0000e+00, 3.9871e-06, 1.4233e-07])
orig_shape: None
shape: torch.Size([4])
top1: 1
top1conf: tensor(1.0000)
top5: [1, 2, 3, 0]
top5conf: tensor([1.0000e+00, 3.9871e-06, 1.4233e-07, 8.6106e-08])
```

```python
# Evaluate the model's performance on the validation set
results = model.val()
```

Calling the yolo v8 model



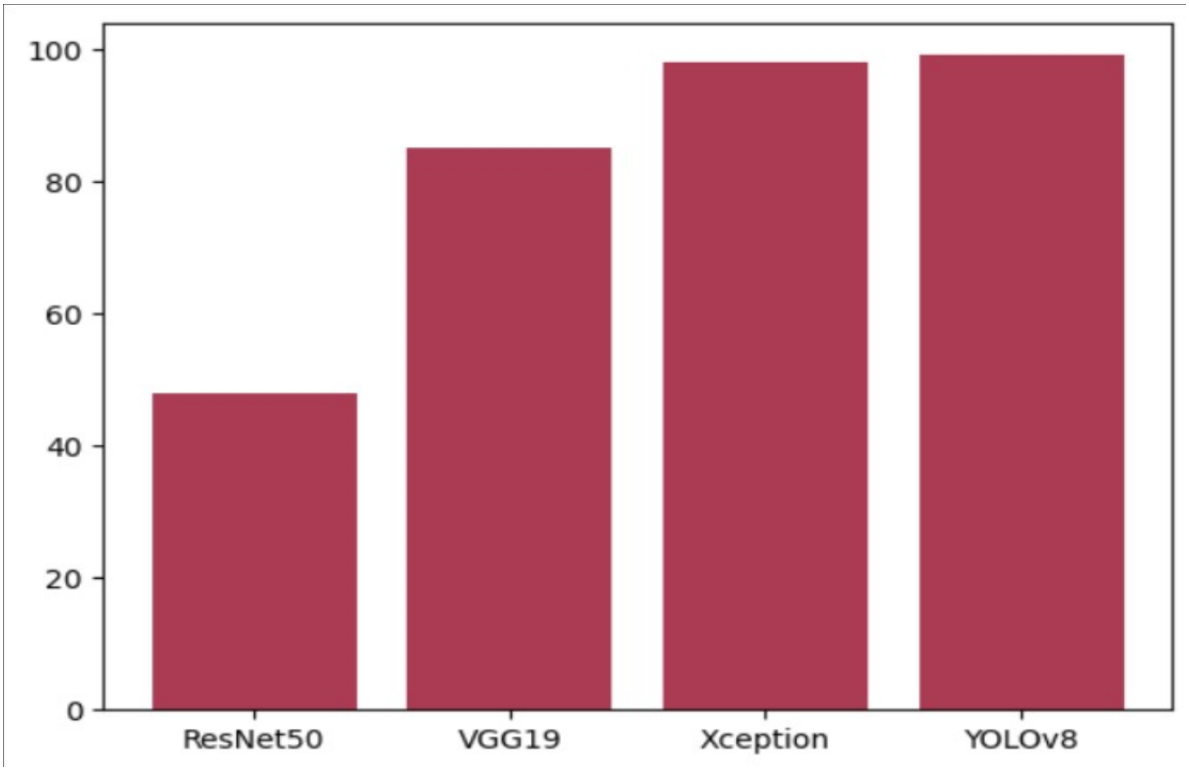
Train/loss of yolo v8 model



Confusion matix of yolo v8 model

8. RESULTS

Our Satellite Image Classification project employed various algorithms with differing accuracy levels: Resnet50 provides 48% accuracy; VGG19 can reach 81% while Xception reaches 98% precision, but for us at our project the best result came from using our own model with its accuracy of 99% achieved within 20-50 epochs of processing time (yOLOV8 model).



Graph showing accuracy of different model where dataset is trained to 50 epoch for each model.

9. CONCLUSION

In our research on cloud detection using satellite images, we employed ResNet-50, VGG-19, Xception, and YOLO v8. Among these, YOLO v8 demonstrated the highest accuracy. This trend is supported by literature, highlighting YOLO's growing popularity for detecting clouds in satellite images. Studies indicate that YOLO and other deep learning techniques excel in locating and removing clouds from satellite photos. Enhancing accuracy by incorporating atmospheric models, varied locations, and time frames has proven effective.

Future work in cloud identification can focus on incorporating sophisticated atmospheric models, expanding datasets to cover various geographic locations and time frames, and optimizing algorithms for real-time applications. Utilizing multispectral and hyperspectral imagery can provide additional data for more accurate cloud identification. Combining data from multiple sensors, such as radar and lidar, with optical imagery can enhance detection capabilities. Additionally, applying transfer learning techniques can help adapt models to new environments, improving their generalizability and applicability across different regions. These advancements can significantly improve atmospheric monitoring and understanding.

10. REFERENCES

[1] Kalkan, M., Bostanci, G. E., Guzel, M., Kalkan, B., Özsarı, Ş., Soysal, Ö., & Köse, G. (2022). Cloudy/clear weather classification using deep learning techniques with cloud images. *Computers and Electrical Engineering* vol. 102, p. 108271, doi: 10.1016/j.compeleceng.2022.108271.

[2] Digra, M., Dhir, R., & Sharma, N. (2022). Land use land cover classification of remote sensing images based on the deep learning approaches: a statistical analysis and review. *Arabian Journal of Geosciences 2022 15:10*, *15*(10), 1–24. https://doi.org/10.1007/S12517-022-10246-8

[3] Ji, S., Dai, P., Lu, M., & Zhang, Y. (2020). Simultaneous Cloud Detection and Removal From Bitemporal Remote Sensing Images Using Cascade Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, pp. 1-17, doi: 10.1109/TGRS.2020.2994349.

[4] Zhou, N., Li, X., Shen, Z., Wu, T., & Luo, J. (2020). Geo-Parcel-Based Change Detection Using Optical and SAR Images in Cloudy and Rainy Areas. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. PP, pp. 1-1, doi: 10.1109/JSTARS.2020.3038169.

[5] M. Segal-Rozenhaimer, A. Li, K. Das, and V. Chirayath, "Cloud detection algorithm for multi-modal satellite imagery using convolutional neural-networks (CNN)," Remote Sensing of Environment, vol. 237, p. 111446, 2020, doi: 10.1016/j.rse.2019.111446.

[6] S. Mahajan and B. Fataniya, "Cloud detection methodologies: variants and development—a review," *Complex & Intelligent Systems*, vol. 6, pp. 1-11, 2019, doi: 10.1007/s40747-019-00128-0.

[7] J. Yang, J. Guo, H. Yue, Z. Liu, H. Hu, and K. Li, "CDnet: CNN-Based Cloud Detection for Remote Sensing Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, pp. 1-17, 2019, doi: 10.1109/TGRS.2019.2904868.

[8] Q. Li, W. Lyu, J. Yang, and J. Wang, "Thin Cloud Detection of All-Sky Images Using Markov Random Fields," *IEEE Geoscience and Remote Sensing Letters*, vol. 9, pp. 417-421, 2012, doi: 10.1109/LGRS.2011.2170953.

[9] P. Panchal and R. Gupta, "Advancement of cloud detection algorithm in satellite images with application to color models," 2015, doi: 10.13140/RG.2.1.2449.3285.

[10] H. Li, H. Zheng, C. Han, H. Wang, and M. Miao, "Onboard Spectral and Spatial Cloud Detection for Hyperspectral Remote Sensing Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 1, pp. 38-50, Jan. 2018, doi: 10.1109/TGRS.2017.2756031.

[11] S. H. Arun, S. Chaurasia, A. Misra, and R. Kumar, "Fog Stability Index: A novel technique for fog/low clouds detection using multi-satellites data over the Indo-Gangetic plains during winter season," *Journal of Atmospheric and Oceanic Technology*, vol. 35, no. 1, pp. 1-15, Jan. 2018, doi: 10.1175/JTECH-D-17-0074.1.

[12] N. Hayatbini, K.-l. Hsu, S. Sorooshian, Y. Zhang, and F. Zhang, "Effective Cloud Detection and Segmentation using a Gradient-Based Algorithm for Satellite Imagery; Application to improve PERSIANN-CCS," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 5, pp. 2938-2952, May 2019, doi: 10.1109/TGRS.2018.2875120.

[13] Y. Changhui, Y. Yuan, M. Minjing, and Z. Menglu, "Cloud Detection Method Based on Feature Extraction in Remote Sensing Images," *International Journal of Remote Sensing*, vol. 34, no. 7, pp. 2387-2404, May 2013, doi: 10.1080/01431161.2012.741430.

[14] D. Frantz, E. Ha, A. U., J. Stoffels, and J. Hill, "Improvement of the Fmask algorithm for Sentinel-2 images: Separating clouds from bright surfaces based on parallax effects," *Remote Sensing*, vol. 10, no. 1, p. 38, Sep. 2018, doi: 10.3390/rs10010038.

[15] X. Song, Z. Liu, and Y. Zhao, "Cloud Detection and Analysis of MODIS Multispectral Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 9, pp. 1965-1975, Sep. 2004, doi: 10.1109/TGRS.2004.835424.

[16] L. Sun, X. Liu, Y. Yang, T. Chen, Q. Wang, and X. Zhou, "A cloud shadow detection method combined with cloud height iteration and spectral analysis for Landsat 8 OLI data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 141, pp. 135-148, Apr. 2018, doi: 10.1016/j.isprsjprs.2018.04.004.

[17] H. Ishida, Y. Oishi, K. Morita, K. Moriwaki, and T. Y. Nakajima, "Development of a support vector machine-based cloud detection method for MODIS with the adjustability to various conditions," *Remote Sensing of Environment*, vol. 195, pp. 162-174, Jan. 2017, doi: 10.1016/j.rse.2017.04.015.

[18] Z. Li, H. Shen, Y. Wei, Q. Cheng, and Q. Yuan, "Cloud Detection by Fusing Multi-Scale Convolutional Features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4523-4537, Aug. 2018, doi: 10.1109/TGRS.2018.2794531.

[19] Y. Oishi, H. Ishida, T. Y. Nakajima, R. Nakamura, and T. Matsunaga, "Preliminary verification for application of a support vector machine-based cloud detection method to GOSAT-2 CAI-2," *Journal of Atmospheric and Oceanic Technology*, vol. 35, no. 5, pp. 1063-1077, May 2018, doi: 10.1175/JTECH-D-17-0138.1.

[20] X. Wu and Z. Shi, "Utilizing Multilevel Features for Cloud Detection on Satellite Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 11, pp. 8497-8512, Nov. 2018, doi: 10.1109/TGRS.2018.2839937.

[21] [21] E. Ricciardelli, F. Romano, and V. Cuomo, "Physical and statistical approaches for cloud identification using Meteosat Second Generation-Spinning Enhanced Visible and Infrared Imager Data," *Remote Sensing of Environment*, vol. 112, no. 4, pp. 1590-1604, Apr. 2008, doi: 10.1016/j.rse.2007.08.020.

[22] Y. Chen, R. Fan, M. Bilal, X. Yang, J. Wang, and W. Li, "Multilevel Cloud Detection for High-Resolution Remote Sensing Imagery Using Multiple Convolutional Neural Networks," *Remote Sensing*, vol. 10, no. 5, pp. 764-779, May 2018, doi: 10.3390/rs10050764.

[23] C. Deng, Z. Li, W. Wang, S. Wang, L. Tang, and A. C. Bovik, "Cloud Detection in Satellite Images Based on Natural Scene Statistics and Gabor Features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 803-815, Apr. 2019, doi: 10.1109/TGRS.2018.2863155.

[24] N. Ghasemian and M. Akhoondzadeh, "Introducing two Random Forest based methods for cloud detection in remote sensing images," *Remote Sensing*, vol. 10, no. 6, pp. 929-944, Jun. 2018, doi: 10.3390/rs10060929.

2% Overall Similarity

Top sources found in the following databases:

- 1% Internet database
- 1% Publications database
- Crossref database
- Crossref Posted Content database
- 1% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	**link.springer.com** Internet	<1%
2	**Kaining Li, Nan Ma, Lin Sun. "Improving Cloud Detection over Bright Un...** Crossref	<1%
3	**mgv.sggw.edu.pl** Internet	<1%
4	**cris.tau.ac.il** Internet	<1%
5	**ijraset.com** Internet	<1%
6	**Cranfield University on 2012-01-18** Submitted works	<1%
7	**researchgate.net** Internet	<1%
8	**Gitam University on 2023-11-01** Submitted works	<1%

Sources overview

9	**King's College on 2019-04-22** Submitted works	<1%