DEMOCRATIC AND PEOPLE'S REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY Dr. MOULAY TAHAR SAIDA
FACULTY: TECHNOLOGY
DEPARTMENT: COMPUTER SCIENCE

Master Thesis

Major : Computer Modeling of Knowledge and Reasoning

Theme

Deep Learning For Lunar Image Segmentation And Object Detection.

Presented By :

Khouloud Saadia KAID

Manar Dounia TAZI

Supervised By :

Dr. RAHMANI MOHAMED ELHADI

Academic Year : 2022-2023

ABSTRACT

Abstract

The exploration of the lunar surface and its geological features plays a crucial role in advancing our understanding of the moon's composition and history.However, traditional methods of lunar image analysis are time- consuming and rely heavily on human expertise. In recent years, the availability of high-resolution lunar image datasets has opened up new opportunities for the application of deep learning more specifically in the feild of computer vision techniques in lunar research.

This thesis employs various deep learning techniques in segmentation and object detection. For segmentation, different models such as U-Net, VGG16, ResNet, and autoencoder are utilized.In addition Transfer learning techniques were also applied, where pre-trained models on large-scale datasets are fine-tuned on the lunar image dataset to improve segmentation performance.In terms of object detection, the YOLOv8 model was the best choice for us.

By combining these segmentation and object detection techniques, the study aims to achieve accurate and detailed analysis of the lunar image dataset, enabling researchers to extract valuable insights about the lunar surface and supporting future missions.

Keywords: computer vision,U-Net,Transfer learning,YOLOv8,object detection,image segmentation. .

Résumé

L'exploration de la surface lunaire et de ses caractéristiques géologiques joue un rôle crucial dans l'avancement de notre compréhension de la composition et de l'histoire de la lune. Cependant, les méthodes traditionnelles d'analyse d'images lunaires sont chronophages et reposent fortement sur l'expertise humaine. Ces dernières années, la disponibilité de jeux de données d'images lunaires haute résolution a ouvert de nouvelles opportunités pour l'application de techniques d'apprentissage profond, plus spécifiquement dans le domaine des techniques de vision par ordinateur en recherche lunaire.

Cette thèse utilise diverses techniques d'apprentissage profond pour la segmentation et la détection d'objets. Pour la segmentation, différents modèles tels que U-Net, VGG16, ResNet et autoencodeur sont utilisés. De plus, des techniques de transfert d'apprentissage sont également appliquées, où des modèles pré-entraînés sur des ensembles de données à grande échelle sont affinés sur l'ensemble de données d'images lunaires pour améliorer les performances de segmentation. En ce qui concerne la détection d'objets, le modèle YOLOv8 a été le meilleur choix pour nous.

En combinant ces techniques de segmentation et de détection d'objets, l'étude vise à obtenir une analyse précise et détaillée de l'ensemble de données d'images lunaires, permettant aux chercheurs d'extraire des informations précieuses sur la surface lunaire et de soutenir les missions futures.

Mots clés: vision par ordinateur, U-Net, transfert d'apprentissage, YOLOv8, détection d'objets, segmentation d'image.

ا

الملخص

يلعب استكشاف سطح القمر وخصائصه الجيولوجية دورًا حاسمًا في تقدم- فهمنا لتكوين القمر وتاريخه. ومع ذلك ، فإن الطرق التقليدية لـ يستغرق تحليل الصور القمرية وقتًا طويلاً ويعتمد بشكل كبير على الخبرة البشرية. فى السنوات الاخيرة، أتاح توفر مجموعات بيانات صور القمر عالية الدقة فرصًا جديدة لـ تطبيق التعلم العميق بشكل أكثر تحديدًا في مجال تقنيات الرؤية الحاسوبية في القمر بحث. تستخدم هذه الأطروحة العديد من تقنيات التعلم العميق في التقسيم واكتشاف الأشياء. بالنسبة للتجزئة ، يتم استخدام نماذج مختلفة مثل U-Net و VGG16 و ResNet و autoencoder بالإضافة إلى ذلك ، تم أيضًا تطبيق تقنيات التعلم الانتقالي ، حيث تم تدريب النماذج مسبقًا على مجموعات البيانات واسعة النطاق ، يتم ضبطها بدقة على مجموعة بيانات الصورة القمرية لتحسين التجزئة الأداء.من حيث اكتشاف الكائن ، كان نموذج YOLOV8 هو الخيار الأفضل بالنسبة لنا. من خلال الجمع بين تقنيات التجزئة والكشف عن الأشياء ، تهدف الدراسة إلى تحقيق ذلك تحليل دقيق ومفصل لمجموعة بيانات الصورة القمرية ، مما يتيح للباحثين استخلاصها رؤى قيمة حول سطح القمر ودعم البعثات المستقبلية

الكلمات المفتاحية: YOLOV8، U-Net ، رؤية الكمبيوتر ، نقل التعلم ، كشف الأشياء ، تجزئة الصورة. .

ACKNOWLEDGEMENTS

First and foremost, we express our heartfelt gratitude to Allah for granting us the strength and guidance to successfully complete this thesis in an outstanding manner.
We would like to thank our supervisor, Dr. Elhadi RAHMANI, for his support throughout this project. Dr. RAHMANI has provided us with invaluable feedback and insights, which have helped us to improve our thesis.
We are also immensely grateful to our parents for their unwavering love, support, and encouragement, which have been the cornerstone of our achievements.We are forever indebted to them for instilling in us the values of perseverance and determination.
Finally, we would like to express our heartfelt appreciation to our friends for their unwavering motivation and support, which have been a constant source of Motivation.
This thesis would not have been possible without the support of all of these people. We are truly grateful for evrything.

Dedication

to all who kindly gave their help and support throughout the development of this thesis.we dedicate this work to you with deep gratitude.

ACRONYMS

AI : *Artificial Intelligence.*
LSTM: *Long short-term memory.*
DEMs: *Digital Elevation Models of the Moon.*
UV: *ultraviolet visible.*
NASA: *National Aeronautics and Space Administration.*
NASA-DOD: *Department of Defense.*
FCNs: *Fully Convolutional Neural Networks.*
CNN : *Convolutional Neural Network.*
ReLU: *Rectified Linear Unit.*
ELU: *Exponential Linear Unit.*
VGG: *Visual Geometry Group.*
AE: *auto encoder.*
ResNet: *Residual Networks.*
RMSprop: *Root Mean Squared Propagation.*
ADAM: *Adaptive Moment Estimation.*

CONTENTS

Dedication **7**

Contents **10**

Page

List of Tables **13**

List of Figures **14**

General Introduction **1**

Problematic and Motivation 1

Thesis Organisation 2

1 Lunar images **3**

1.1 What is lunar images? 3

1.2 What are the difficulties to collect the moon terrain data? 4

1.3 The acheived projects 5

1.4 Why synthetic lunar images are often preferred over real lunar images in deep learning? 9

1.5 Synthetic lunar images 10

1.5.1 Synthetic lunar images types 10

1.5.2 Applications of synthetic lunar images in deep learning 10

1.5.3 The process of the creation 11

1.6 Our Dataset's Images(Connecting the Dataset and Lunar Images) 12

1.7 Conclusion 13

2 Deep Learning for Image Processing **15**

2.1 Current uses and tendencies: 16

2.2 Computer vision: 16

2.3 Deep neural network 17

2.3.1 Basic Concepts: 17

2.3.2 Multi layer perceptron (MLP) 18

2.3.2.1 Forward Propagation 19
2.3.2.2 Back-Propagation 20
2.3.2.3 The loss function 20
2.3.2.4 Gradient decent 20
2.4 Convolutional Neural Network (CNN) 22
2.4.1 Layers 23
2.4.1.1 Activation Layers 23
2.4.1.2 Pooling Layers 23
2.4.1.3 Batch Normalization 23
2.4.2 Convolutional Architectures 24
2.4.2.1 CNNs (LeNet) 24
2.4.2.2 Deep CNNs 24
2.4.2.3 Very Deep CNNs 25
2.4.2.4 Residual CNNs 26
2.4.2.5 Other architectures 26
2.4.3 Training CNNs 27
2.4.3.1 CNN Operations 27
2.4.3.2 Overfitting and underfitting 28
2.4.3.3 Regularization 29
2.4.4 Optimization algorithms 29
2.4.4.1 Data augmentation 30
2.4.4.2 Stochastic Gradient Descent (SGD) 30
2.4.4.3 Adaptive Moment Estimation (Adam) 31
2.5 Image Segmentation using CNNs 32
2.5.1 Semantic segmentation 32
2.5.2 Instance segmentation 32
2.5.3 Evaluation metrics 33
2.6 Object detection using CNNs 33
2.6.1 Object detection algorithms 34
2.6.2 Evaluation metrics 34
2.7 Image classification using CNNs 35
2.8 Transfer Learning 35
2.8.1 Fine-tuning 36
2.9 Conclusion 36

3 Experiments and Results **37**
3.1 Data set description (Artificial Lunar Landscape Dataset) 37
3.2 The used environment 38
3.3 Programming Language 38

3.4 Implementation tools(API) 38
3.4.1 Tenserflow 38
3.4.2 Keras 39
3.4.3 Matplotlib 39
3.4.4 Numpy 39
3.4.5 Sklearn 39
3.4.6 OpenCV 39
3.5 U-Net Architecture 40
3.5.1 Encoder 40
3.5.2 Decoder 41
3.5.3 Skip connections 42
3.6 Experiments And Implementation 42
3.6.1 State-of-the-Art Approaches And Results 42
3.6.2 Semantic Segmentation Models 42
3.6.2.1 Basic U-Net Architecture 42
3.6.2.2 Transfer learning using U-Net 49
3.6.2.3 Autoencoders: 52
3.6.2.4 Segment Anything Model(SAM): 54
3.6.3 Object detection Models 54
3.6.3.1 Big rocks detection(Blue Rocks): 54
3.6.3.2 Grounding DINO: 55
3.6.3.3 YOLOv8: 57
3.6.3.4 Comparison Between YOLOv8 and Grounding DINO : 62
3.6.4 Grounding DINO & SAM for Image segmentation 62
3.7 Conclusion : 63

General conclusion **65**
Future plan 65

Bibliography **67**

List of Tables

Table **Page**

3.1 UNet Architecture Summary 43
3.2 Example of a combination. 46
3.3 VGG16 Model Summary 50
3.4 Validation Results 61

LIST OF FIGURES

FIGURE **Page**

1.1 GEOMETRICAL PERSPECTIVE OF THE FIRST EARTH-MOON PHOTO & NASA chart from 1966 shows how this iconic photo was taken.Image: NASA [65] 5

1.2 ORIGINAL VERSION OF THE LUNAR ORBITER 1 "EARTHRISE" IMAGE Image: NASA [1] 6

1.3 Commander of the Apollo 11 mission, 1969.[46] 6

1.4 Lunar Prospector Orbiter (NASA) [13] 7

1.5 This mosaic from SMART-1 shows a trio of impact craters very near the Moon's north pole. Credit: ESA/SMART-1/AMIE camera team/Space Exploration Institute,CC BY-SA 3.0 IGO [44] 7

1.6 Side by side comparison of the first ever photograph of the lunar far side, from Luna 3, and a visualization of the same view using LRO data. The LRO Moon includes latitude and longitude lines at 15-degree intervals.[47] 8

1.7 Illustration of the Lunar Pathfinder mission in lunar orbit providing its services (image credit: SSTL).[17] 8

1.8 Shown is a screen shot of the In-Situ Resource Utilization (ISRU) Pilot Excavator on the Moon through a 3D software simulation created by Kennedy Software Engineer Kurt Leucht.[43] 11

2.1 The output 17

2.2 Illustration of a single neuron with four input values x1, x2, x3, b. [58] 17

2.3 Common nonlinear activation function [28] 18

2.4 Example for a feed-forward neural network, also known as multilayer perceptron. The inputs are marked with blue circles, whereas the neurons are illustrated with orange circles. While the weight parameters are represented as links between the nodes, the bias parameters are denoted as links coming from additional nodes labeled with "+1". [58] 19

2.5 Illustration of the full process [40] 19

2.6 find The Minimum Value [50] 21

2.7 Gradien descent illustration [59] 21

2.8 Convolution layer [51] 22
2.9 AlexNet Architecture [54] 24
2.10 VGG architecture [62] 25
2.11 Googlenet architecture [24] 25
2.12 Inception module [15] 26
2.13 Convolution operation [51] 27
2.14 Underfitting and Overfitting in ML[51] 28
2.15 Capacity of model [27] 28
2.16 Learning curves [23] 29
2.17 Stochastic Gradient Descent (SGD)[53] 31
2.18 Adam algorithm [22] 31
2.19 Semantic segmentation examples (source: SegNet) [11] 32
2.20 Semantic VS Instance segmentation [61] 33
2.21 Object detection example [4] 34
2.22 How a CNN classifies an image [3] 35

3.1 Artificial Lunar Landscape Dataset 38
3.2 U-Net architecture [7] 40
3.3 Illustration of the Encoder [7] 41
3.4 Illustration of the Decoder [7] 41
3.5 Loss Calculated with Tversky loss function 44
3.6 Loss Calculated with Dice loss function 45
3.7 Whitout Using Data Augmentation 46
3.8 Using Data Augmentation 46
3.9 The Output Image using Unet 47
3.10 The Obtained Results 47
3.11 The Obtained Results Adding Batch Normalisation 48
3.12 Output Image 48
3.13 Image 49
3.14 Mask 49
3.15
Predicted cmap='BrBG' 49
3.16 Output Image RESNET50 & Unet 49
3.17 Output Image VGG16 & Unet 51
3.18 The architecture of autoencoders [29] 52
3.19 Image 53
3.20 Predicted 53
3.21 Output Image Using Autoencoders 53
3.22 the generated mask using SAM 54

3.23 Blue Rocks Detection 55
3.24 DINO's architecture [38] 56
3.25 DINO's Result 56
3.26 YOLOv8 Output Without Additional Training 57
3.27 Annotating 58
3.28 The confusion matrix returned after training 59
3.29 Results Tracked by YOLOv8 59
3.30 Curves 60
3.31 YOLOv8 Output After Additional Training 61
3.32 Output Image 62

General Introduction

Deep learning and neural networks have recently become effective tools in the area of artificial intelligence (AI), revolutionizing a number of fields like computer vision, natural language processing, robotics, and healthcare. These methods have made it possible for computers to learn from enormous volumes of data, identify complex patterns, and make precise predictions, opening the door for huge developments in AI research and applications. moreover Convolutional neural networks (CNNs), in particular, have shown to be very effective in a variety of image processing tasks, including object detection, semantic segmentation, and image restoration. This thesis seeks to investigate and create reliable methods for both lunar images segmentation and rocks detection by using the capabilities of deep learning.

Image segmentation and object detection play vital roles in comprehending lunar images for lunar exploration. Traditional approaches were laborious and subjective, but the advent of deep learning, specifically CNNs, has transformed these tasks. CNN-based segmentation algorithms automate the precise identification of lunar features and terrains. Moreover, object detection algorithms, can effectively identify specific objects(rocks,craters..etc) or artifacts on the lunar surface, such as landers and rovers.

Lunar rovers face challenges in navigating steep and rocky terrains on the moon.as mentioned before that Real-time segmentation and object detection are crucial for safe exploration and scientific analysis, but the Limited annotated datasets and the complex lunar surface add further complexity. Overcoming these challenges is essential for successful rover navigation.

Problematic and Motivation

In the field of space robotics, particularly in the exploration of the moon, we rely on the power of lunar rovers, which are partially or fully autonomous vehicles designed to navigate the lunar surface. However, operating these vehicles in the challenging lunar environment presents various obstacles. One such challenge is the safe navigation of lunar rovers through steep and rocky terrains, which requires precise path planning and obstacle avoidance. Additionally, performing real-time segmentation and object detection on lunar images is essential for the rover to accurately perceive its surroundings and identify potential hazards or scientifically interesting features. These tasks are further complicated by limited annotated datasets for training segmentation and object detection algorithms, as well as the complex nature of the lunar surface with its diverse geological features.Overcoming these challenges is crucial to ensure the successful and safe navigation of lunar rovers, enabling them to effectively explore the moon's terrain, perform accurate segmentation and object detection, and contribute to scientific discoveries.

Thesis Organisation

This thesis is divided into three main chapters as detailed bellow:

- **Lunar images:**The first chapter gives an overview of lunar images and their importance to lunar missions and research. It explores how to produce them if we delve into the synthetic ones, as well as the many techniques used to obtain real lunar images from satellites, rovers, and landers. It also emphasizes the difficulties of doing manual lunar image analysis and provides the foundation for the coming chapters on deep learning for lunar image.
- **Deep Learning:** Deep learning methods for processing images are the main topic of the second chapter. It starts out with a description of deep learning and how it may be used for computer vision applications. It then gets into the principles of CNNs, going into the architecture, parts, and popular models used in image processing applications. Additionally, it goes over deep learning-based techniques for image segmentation and also incorporates object detection using deep learning models.
- **Experiments and Results:** The third chapter describes deep learning's implementation stage. It covers crucial components such as dataset description, model selection, architectural design, training, optimization, evaluation, and performance analysis. The chapter gives perspectives on the actual implementation of deep learning algorithms particularly for lunar image segmentation and detection.

Chapter 1

Lunar images

The moon, Earth's only natural satellite, has long been a subject of fascination and exploration for scientists, astronomers, and the general public alike. From the early days of telescopic observation to the recent lunar exploration missions, the moon has revealed its many mysteries through the lens of advanced imaging techniques.

Lunar images, captured through various imaging techniques which we will discover later in this chapter. These images have been used for scientific research, as well as for artistic and educational purposes what made them a valuable source of information about the moon's geology, topography, and mineralogy.

The first close-up images of the Moon were taken by the Soviet Union's Luna 3 spacecraft in 1959. Since then, numerous spacecraft have been sent to the Moon to take high-resolution images, including the Lunar Reconnaissance Orbiter (LRO), which has been in orbit around the Moon since 2009.

In addition to spacecraft images, many amateur astronomers also take photographs of the Moon using telescopes and cameras. These images can capture details of the lunar surface such as craters, mountains, and valleys. In the same vein, with the advent of machine learning techniques, such as Convolutional Neural Networks (CNNs) and image segmentation, lunar images have become an even more powerful tool for analyzing and interpreting lunar data. In this chapter we will dive into each essential point of this field for more understanding.

1.1 What is lunar images?

Lunar images are photographs or digital images of the moon's surface. These images can be either real, captured by cameras or other instruments aboard spacecraft or telescopes, or synthetic, generated using computer graphics techniques to simulate the appearance of the moon's surface.

Real lunar images are typically captured at different wavelengths of light, such as visible light, infrared, and ultraviolet, to reveal different aspects of the moon's surface features, such as its composition and texture. The resolution of these images varies depending on the imaging technique used, but can range from a few meters to a few millimeters per pixel. For example, images captured by the Lunar Reconnaissance Orbiter Camera (LROC) have a resolution of up to 50 centimeters per pixel.[45]

Synthetic lunar images are computer-generated images that simulate the appearance of the moon's landscape. These images are generated using computer graphics techniques. In contrast to real lunar images, synthetic images can provide consistent and controlled conditions for a specific purpose, as well as the effects of different lighting conditions and viewing angles. To create them, a variety of data sources are used including lunar topographic maps, high-resolution digital elevation models, and spectral data from remote sensing instruments. These data sources are combined with computer graphics algorithms to create realistic models of the moon's surface. The resulting images can be generated at various resolutions and in different lighting conditions to simulate different lunar environments.[39]

Both real and synthetic lunar images are commonly used in the study of the moon's surface features and geology. They can be analyzed using various techniques, such as image segmentation and machine learning algorithms, to extract information about the moon's terrain, geological formations, and mineralogical composition.

in this thesis we used the synthetic images, that is why our focus in this chapter will be all about them.

1.2 What are the difficulties to collect the moon terrain data?

The lunar project faces numerous challenges. One of these challenges involved acquiring the appropriate dataset. Similar to other planetary data, lunar terrain data is both highly sensitive and scarce. This scarcity can be attributed to two main factors. Firstly, it is related to the nature of deep learning models. These models heavily rely on large datasets to yield optimal results. However, in the case of moon terrain data, the availability of such extensive datasets is not feasible. This is due to the fact that a robot records footsteps on the moon's surface, and this information is then transmitted to a satellite and subsequently to the robot's control center. The transfer speed is low, and efficient time management is essential as the robot must balance its work with data transfer. Therefore, there is a need to strike a compromise between these two factors, resulting in limited data availability.The second constraining element arises from the fact that if we're not a member of a space agency we can not have access to an optimum sample of data.

1.3 The acheived projects

Lunar Orbiter program: This series of five unmanned missions in the mid-1960s was designed to photograph potential landing sites for the Apollo missions. The Lunar Orbiters captured high-resolution images of the lunar surface, including the first photographs of the Earth from the Moon's surface.[1]

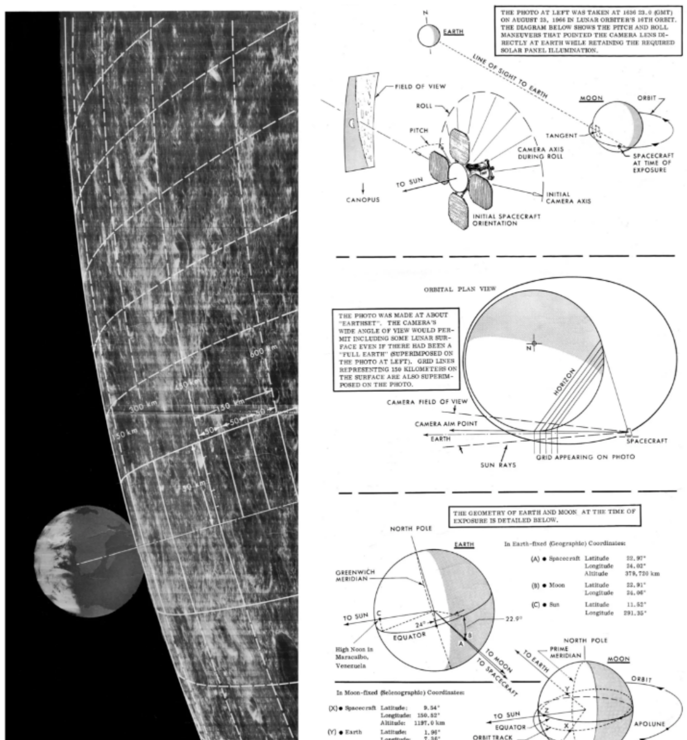


Figure 1.1: GEOMETRICAL PERSPECTIVE OF THE FIRST EARTH-MOON PHOTO & NASA chart from 1966 shows how this iconic photo was taken.Image: NASA [65]

[1]https://www.nasa.gov/feature/50-years-ago-the-lunar-orbiter-program

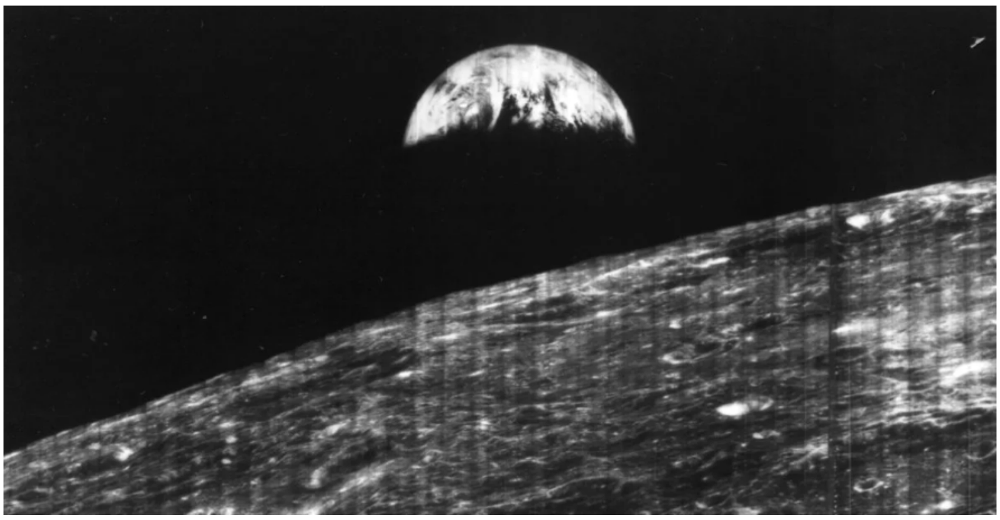


Figure 1.2: ORIGINAL VERSION OF THE LUNAR ORBITER 1 "EARTHRISE" IMAGE Image: NASA [1]

Apollo program: The Apollo missions of the late 1960s and early 1970s brought humans to the Moon and allowed them to capture high-quality images of the lunar surface. The astronauts used handheld cameras to capture images, and also deployed specialized equipment such as panoramic cameras and stereo cameras.[46]

Figure 1.3: Commander of the Apollo 11 mission, 1969.[46]

Clementine: This joint NASA-DOD mission in 1994 captured images of the Moon using a combination of imaging sensors, including a UV/visible camera and a near-infrared camera. The mission produced the first global map of the Moon's topography and composition.[48]

Lunar Prospector: This NASA mission in 1998-1999 used a gamma-ray spectrometer to map the Moon's elemental composition. The mission also captured images of the lunar surface using a camera. [13]

Figure 1.4: Lunar Prospector Orbiter (NASA) [13]

SMART-1: This European Space Agency mission in 2003-2006 used a combination of instruments, including a camera, to study the Moon's topography and composition.[44]

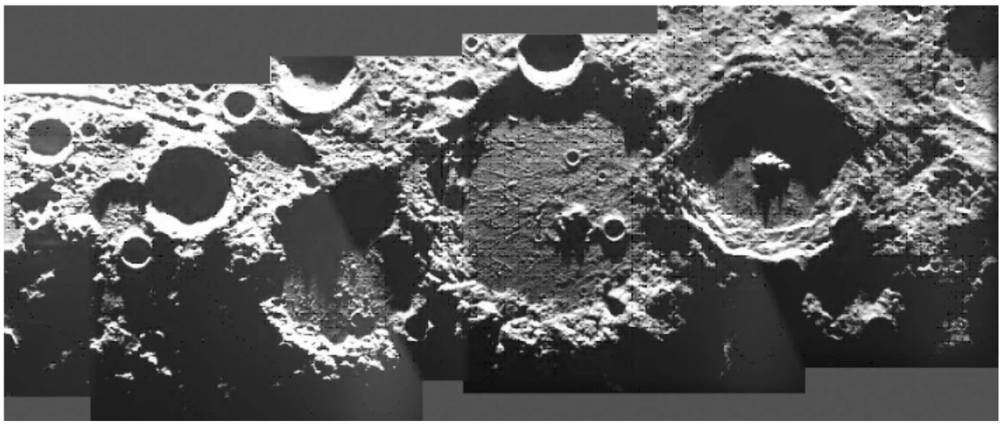


Figure 1.5: This mosaic from SMART-1 shows a trio of impact craters very near the Moon's north pole. Credit: ESA/SMART-1/AMIE camera team/Space Exploration Institute,CC BY-SA 3.0 IGO [44]

Lunar Reconnaissance Orbiter (LRO): This NASA mission, launched in 2009, is still in operation and has captured high-resolution images of the lunar surface using a suite of instruments, including cameras, spectrometers, and a laser altimeter.[47]

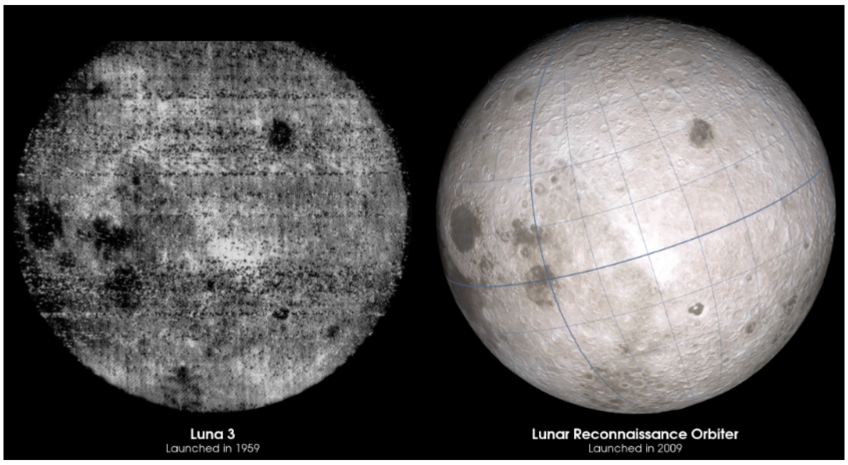


Figure 1.6: Side by side comparison of the first ever photograph of the lunar far side, from Luna 3, and a visualization of the same view using LRO data. The LRO Moon includes latitude and longitude lines at 15-degree intervals.[47]

Change program: Chinas lunar exploration program has included a series of robotic missions, beginning with Change 1 in 2007 and continuing with Change 2, 3, and 4. These missions have included cameras and other instruments for studying the lunar surface.[66]

Lunar Pathfinder: This privately-funded mission, planned for launch in 2022, will use a camera to capture high-resolution images of the lunar surface, as well as other instruments for scientific research.[18]

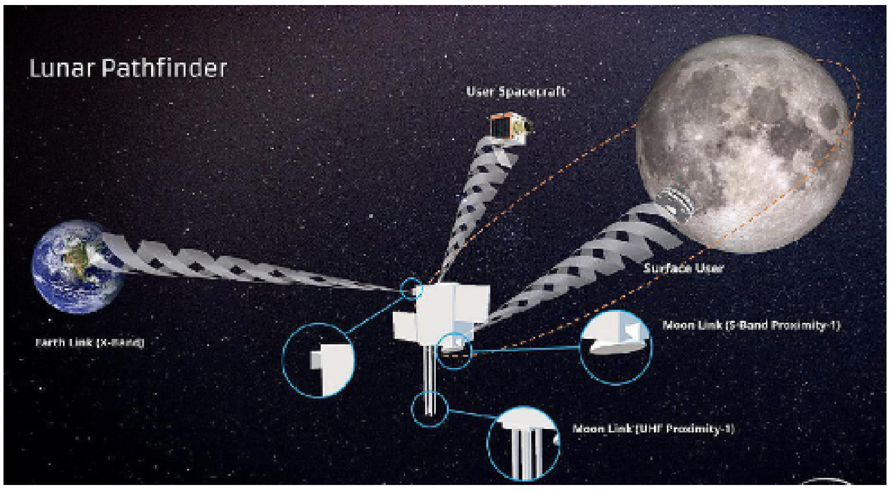


Figure 1.7: Illustration of the Lunar Pathfinder mission in lunar orbit providing its services (image credit: SSTL).[17]

These are just a few examples of the many projects related to lunar imaging over the years. Each mission has contributed valuable information about the Moon's surface features, geology, and history.

1.4 Why synthetic lunar images are often preferred over real lunar images in deep learning?

One of the main reasons is that acquiring real lunar images can be expensive due to the need for specialized equipment, such as lunar rovers or spacecraft, to capture high-quality images. Additionally, the processing and storage of large quantities of real lunar images can also be expensive. Let's now see other reasons:

1. **Availability:** It is often easier and more cost-effective to generate synthetic lunar images than to acquire real lunar images. Acquiring real lunar images can be challenging due to limited availability, the expense of acquiring and processing the images, and the difficulty in obtaining high-quality images that are suitable for use in deep learning models.

2. **Control:** Synthetic lunar images offer more control over the image generation process. Variables such as lighting conditions, terrain type, and atmospheric conditions can be controlled to create images with specific characteristics, which can help to improve the accuracy and robustness of deep learning models.

3. **Ground Truth:** Synthetic lunar images can be generated with known ground truth information. This means that the precise characteristics and features of the image are already known and can be used for training deep learning models. In contrast, real lunar images may be subject to unknown variations and inconsistencies that can make it difficult to establish ground truth information.

4. **Large and Diverse Datasets:** Synthetic lunar images can be generated in large quantities and with diverse characteristics to create datasets that are representative of the range of conditions that might be encountered on the lunar surface. This can help to improve the accuracy and robustness of deep learning models.

5. **Ethics:** There are ethical concerns around the use of real lunar images for deep learning applications. The use of real lunar images may require the removal of scientific data from public databases, which can limit access to this information and impede scientific progress. In contrast, the creation of synthetic lunar images does not require the use of real data and can be done in a way that respects the scientific process.

1.5 Synthetic lunar images

1.5.1 Synthetic lunar images types

There are several types of synthetic lunar images, which are as follows:

1. **Digital Elevation Models (DEMs):** These images provide a 3D representation of the moon's surface that is based on topographic data obtained from laser altimeters, such as the Lunar Orbiter Laser Altimeter (LOLA). DEMs can be used to generate various synthetic images, including shaded relief maps, slope maps, and contour maps.[42]
2. **Normal Maps:** These images provide a representation of the moon's surface that is based on its surface normals. A surface normal is a vector that is perpendicular to the surface at a given point, and normal maps can be used to simulate the appearance of surface features under different lighting conditions.[19]
3. **Albedo Maps:** These images provide a representation of the moon's surface that is based on its reflectivity. Albedo maps can be used to simulate the appearance of the moon's surface under different lighting conditions and to study the distribution of different types of lunar materials.[60]
4. **Synthetic Images:** These images are generated by combining DEMs, normal maps, albedo maps, and other types of data to create a realistic representation of the moon's surface. Synthetic images can be used for various purposes, including simulating lunar missions, testing image processing algorithms, and studying the moon's surface features under different lighting conditions.[39]
5. **Lunar Global Digital Terrain Models (DTMs):** These are synthetic images that provide a detailed representation of the moon's topography. They are created by combining data from various sources, including laser altimetry, stereo imaging, and radar data.[41]
6. **Lunar Reconnaissance Orbiter Camera (LROC) Synthetic Images:** These are synthetic images that are created using data from the LROC Narrow Angle Camera. The images are processed to remove distortions caused by the camera's optics and to improve the resolution of the images.[74]

1.5.2 Applications of synthetic lunar images in deep learning

Synthetic lunar images are also being increasingly used in deep learning applications related to lunar image analysis. Here are some of the applications:

Image Segmentation: Synthetic lunar images can be used to train deep learning models for lunar image segmentation tasks. By generating images with known ground truth segmentation masks, synthetic data can be used to create large, diverse datasets that can improve the accuracy and robustness of image segmentation algorithms.[37]

Object Detection: Synthetic lunar images can be used to train deep learning models for object detection tasks on the lunar surface. This can include detecting and localizing different types of lunar rocks, craters, and other features.[8]

Terrain Classification: Synthetic lunar images can be used to train deep learning models for terrain classification tasks. By generating images with different surface types and terrain features, synthetic data can be used to create large, diverse datasets that can improve the accuracy and robustness of terrain classification algorithms.[12]

Navigation and Mapping: Synthetic lunar images can be used to train deep learning models for navigation and mapping tasks on the lunar surface. This can include creating maps of the surface and using those maps to navigate robots and other equipment.[36]

Simulation of Lunar Environments: Synthetic lunar images can be used to create realistic simulations of lunar environments for testing and training deep learning models. This can help researchers create more accurate models and improve the robustness of algorithms in challenging lunar environments.[6]

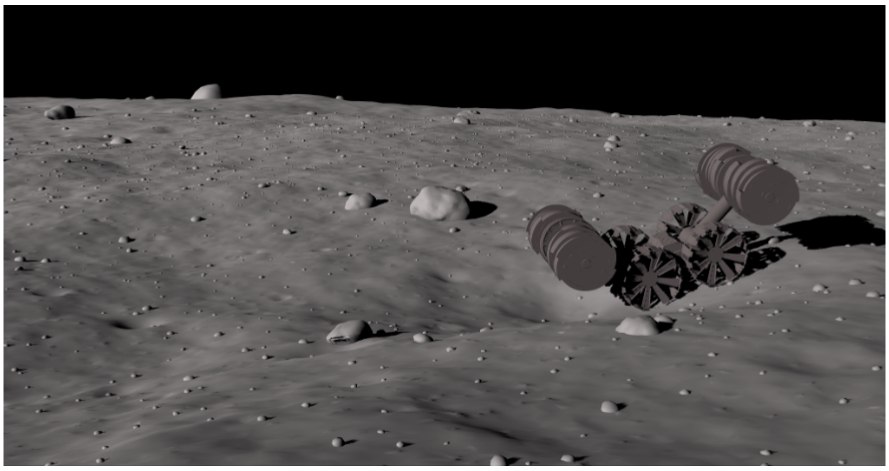


Figure 1.8: Shown is a screen shot of the In-Situ Resource Utilization (ISRU) Pilot Excavator on the Moon through a 3D software simulation created by Kennedy Software Engineer Kurt Leucht.[43]

1.5.3 The process of the creation

The process involves the following steps:

1. **Defining the scene:** The first step is to define the scene that will be rendered in the synthetic image. This includes specifying the terrain, lighting conditions, and any

objects or structures that should be included in the scene.

2. **Creating a 3D model:** Next, a 3D model of the scene is created using specialized software. This model includes the surface features of the moon, such as craters, mountains, and other geological formations. The model can also include any man-made objects or structures that should be included in the scene.

3. **Defining materials:** Once the 3D model is complete, materials are defined for each surface in the scene. This includes specifying the reflectivity, roughness, and other characteristics of the surface materials.

4. **Lighting the scene:** The scene is lit using virtual light sources to create the desired lighting conditions. This includes simulating the shadows and reflections that would be present in the real world.

5. **Rendering the image:** The final step is to render the image by generating a 2D projection of the 3D scene. This is done using specialized rendering software that takes into account the lighting conditions and surface materials to create a realistic-looking image.

6. **Post-processing:** After the image is rendered, it can be post-processed to adjust the colors, contrast, and other aspects of the image to achieve the desired final result.

creating synthetic lunar images requires specialized software and hardware to create the 3D models and render the final images. However, the process can be highly flexible and customizable, allowing for the creation of a wide range of lunar scenes and terrain features.

1.6 Our Dataset's Images(Connecting the Dataset and Lunar Images)

It is a collection of synthetic lunar images that were generated using computer graphics techniques in chapter 2.9 section 3.1 we are going to discuss in detail more information about the used dataset. These synthetic images were created to simulate the appearance of the moon's surface, with varying lighting and terrain conditions.

The Artificial Lunar Landscape Dataset can be related to the use of synthetic lunar images in general, as both involve the creation of computer-generated images that simulate the appearance of the moon's surface.

In particular, the Artificial Lunar Landscape Dataset can be useful for researchers who are interested in developing and testing algorithms for lunar image analysis, such as image segmentation and object detection. By using the dataset as a testbed, researchers can evaluate the performance of their algorithms under different lunar lighting and terrain conditions.

Overall, while this Dataset is a specific example of a synthetic lunar image dataset, it is part of a broader field of research that involves the creation and use of synthetic lunar images for a range of applications in lunar exploration and research.

1.7 Conclusion

In conclusion, the study of lunar images has revolutionized our understanding of Earth's celestial neighbor, the Moon. Throughout this chapter, we have explored the various types of lunar images such as real lunar images that provide a visual record of its surface, while synthetic images simulate different conditions and aid in mission planning. Together, they have enhanced our knowledge of lunar geology, inspired exploration, and guided future missions, ensuring continued progress in lunar research.

Chapter 2

Deep Learning for Image Processing

Research in computer science and engineering has long been active in the area of image processing. Deep learning has revolutionized image processing, allowing computers to analyse and interpret visual data, such as photographs and videos, with impressive speed and precision. Particularly, deep learning has developed into an effective images processing technique that is extensively used in a variety of fields, including computer vision, robotics, healthcare, entertainment, and self-driving automobiles.

The fundamentals of deep learning for image processing will be covered in this chapter. The foundations of deep learning, including its history, essential ideas, and potential applications, will be explored first. The main element of deep learning for image processing, convolutional neural networks (CNNs), will next be covered, along with its construction, training, and optimization.

Due to its capability to automatically learn features directly from the raw pixel data of pictures, CNNs have evolved into the core of many innovative image processing systems. We'll talk about the various CNN models and how they may be used for semantic segmentation, object detection, and image classification. We will also discuss some of the drawbacks and restrictions of deep learning for image processing, for example overfitting, bias, and the need for a large amount of labeled data.

Fully connected networks, and convolutional networks are a few examples of the many deep learning models that we will examine as we go through the chapter. also some of the latest developments in deep learning for image processing will be explored such as transfer learning.

Whether you are a beginner or an experienced practitioner, this chapter will provide you with a solid foundation for understanding deep learning for image processing and its potential for solving real-world problems in a wide range of fields. So, let's get started!

2.1 Current uses and tendencies:

In a short period of time, deep learning techniques' research and application fields have advanced significantly. Deep neural networks first started to achieve amazing results in image classification. Recent advancements in the fields of image classification, single and multiple-object localization, and object detection in videos have all been made possible thanks to the ImageNet challenge, which propelled the research of many groups in this area.

Convolutional neural networks, however, have been employed in a variety of applications, such as those by Gatys, Ecker, and Bethge [21], where these networks may change the content of a photo to fit the picture's aesthetic. The new picture is approximated to both the activations at some layer of the original content and the Gram matrix of the activations of the style image using the pre-trained network in Simonyan and Zisserman [63]. By altering a few hyperparameters in the loss function, the user may change the final image's content and style weights as well as choose the layers that should be taken into consideration during optimization. Ruder, Dosovitskiy, and Brox [57] and Anderson et al. [9] subsequently used this method to analyze video, and during the last year, various other neural art implementations have also surfaced.

Deeper models, increased processing power, and larger datasets have all helped other neural network designs as well. Since a few years ago, recurrent neural networks and LSTM have been applied in the natural language processing disciplines, advancing the state-of-the-art in applications like sentiment analysis, speech-to-text recognition, and language translation. Vinyals et al.'s generative model, which can produce a description of an image provided as input, combined the potent design of convolutional neural networks for image recognition with a recurrent neural network.[75]

2.2 Computer vision:

Computer vision is a scientific field that deals with information extraction from digital pictures. Information gleaned from a picture might range from identification to spatial measures used in applications like augmented reality or navigation.

We can also define it by describing The released applications. Computer vision is the development of algorithms that can comprehend the information of pictures and apply it for various purposes.

There are cameras everywhere, and the amount of photos shared online is increasing dramatically. There are photos on Instagram, movies on YouTube, feeds from security cameras, and visuals from the medical and scientific fields. We need computer vision to filter through these pictures and make it possible for computers to comprehend what they are.Listed here are a few computer vision applications: Self-driving automobiles, augmented reality, face detection, optical character recognition. . . .etc.[67]

2.3 Deep neural network

Artificial neural networks are inspired by biology, more precisely by the human brain's neural net (McCulloch and Pitts [20]) .First and foremost, in order to describe non-linear data, fundamental concepts of neural networks, such as neurons, weights, and activation functions, are explained in this section.

2.3.1 Basic Concepts:

Single neurons are the building blocks of neural networks; these neurons are coupled to one another to create the neural network [35]. A neuron may be thought of as a simple model in and of itself and is the fundamental unit of a neural network. The basic structure of a single neuron is shown in Figure 2.2 In the case of the illustration in Figure 2.2, the neuron processes several inputs (xi) to produce the following output: [58]

$$Y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Figure 2.1: The output

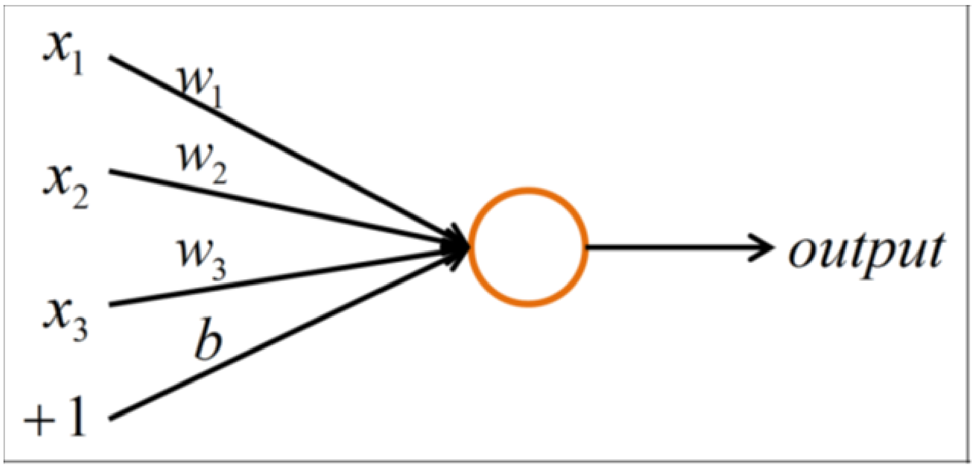


Figure 2.2: Illustration of a single neuron with four input values x1, x2, x3, b. [58]

Where :

- y is the neuron's output.
- F the activation function.
- b is the neuron's bias.

- x_i is the input vector.
- w_i is the weight vector.

Every input value xi is multiplied by an individual weight wi as part of this computing process to give it a weight. The bias b parameter, which enables the addition of an offset to the data, is added to the weighted input values. The result of this linear combination is transformed using a nonlinear activation function to get the final output of the neuron [35].

Neural networks are given non-linearity by using activation functions. It squashes the values in a smaller range viz. Common choices in deep learning for the nonlinear activation function are:

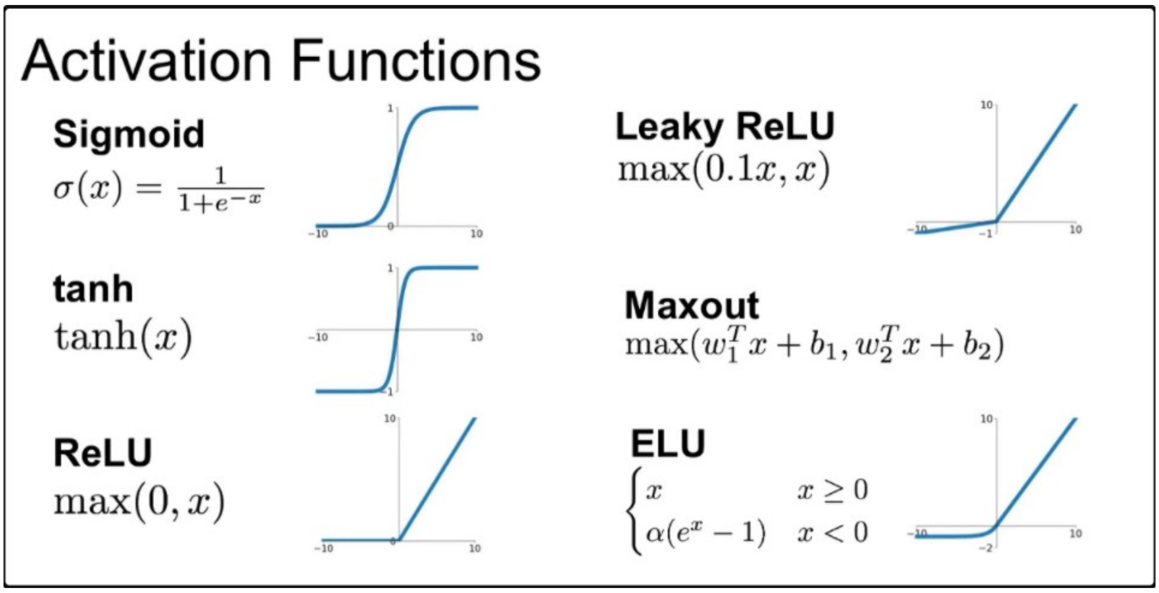


Figure 2.3: Common nonlinear activation function [28]

2.3.2 Multi layer perceptron (MLP)

Neurons that are connected to one another make up a neural network model [35]. The simple example of a neural network model given in Figure demonstrates how the output of one neuron may serve as the input of another neuron. This model is an illustration of a small feed-forward neural network, also referred to as a multilayer perceptron. In this model, the neurons are organized into layers, each layer is fully connected to the one below it (each neuron in a layer is connected with each neuron in the layer below it), the leftmost layer is known as the input layer, and the rightmost layer is known as the output layer. The middle layer is referred to as the hidden layer. [58]

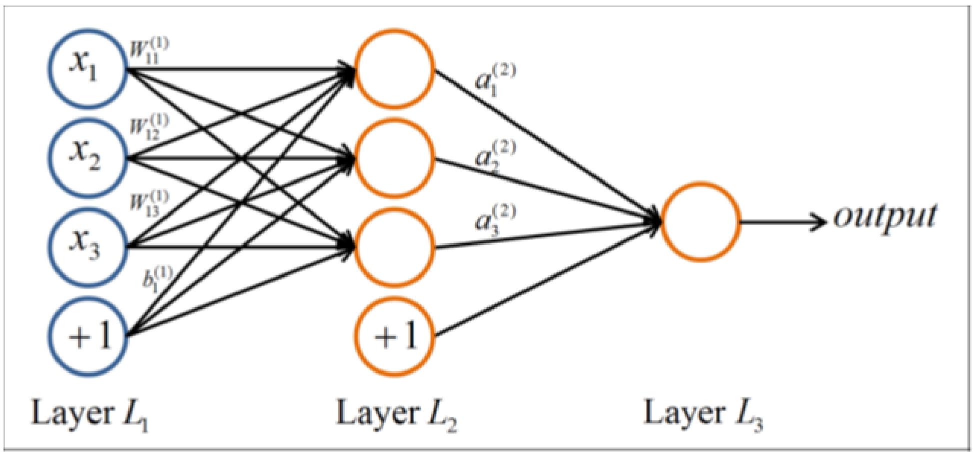
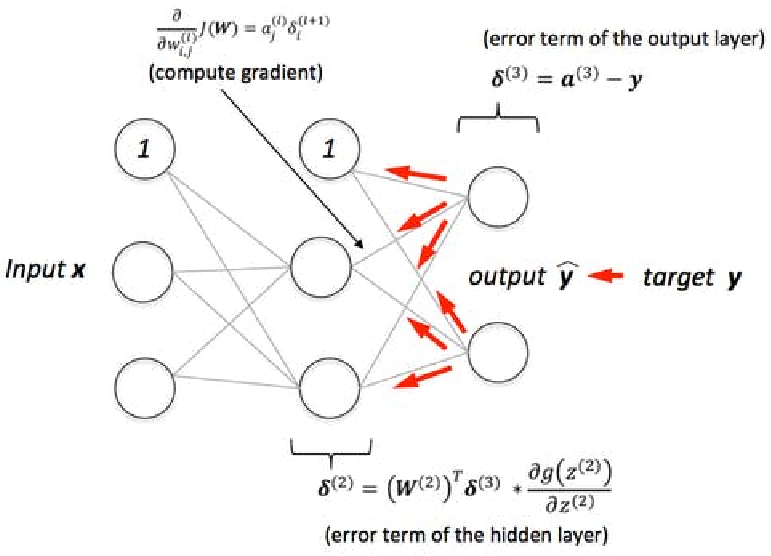


Figure 2.4: Example for a feed-forward neural network, also known as multilayer perceptron. The inputs are marked with blue circles, whereas the neurons are illustrated with orange circles. While the weight parameters are represented as links between the nodes, the bias parameters are denoted as links coming from additional nodes labeled with ”+1”. [58]

2.3.2.1 Forward Propagation

Feeding input values to the neural network during forward propagation results in an output that we refer to as the predicted value. Forward propagation is sometimes called inference. The neural network's first layer performs no operations after receiving the input data. First layer values are taken by the second layer, which then multiplies, adds, and activates them before passing them on to the third layer. The process is repeated for more layers until we reach the final layer, which provides an output value.[40]



$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$
(compute gradient)
(error term of the output layer)
$\delta^{(3)} = a^{(3)} - y$
1
1
Input x
output $\hat{y}$
target y
$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} * \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$
(error term of the hidden layer)

Figure 2.5: Illustration of the full process [40]

2.3.2.2 Back-Propagation

After forward propagation, we obtain an output value that corresponds to the value predicted. Then we compare the predicted value to the actual output value to compute error. To determine the error value, we use a loss function (see below). The derivative of the error value is then calculated respecting each and every weight in the neural network. The chain rule of differential calculus is used in back-propagation. In the chain rule, we first calculate the derivatives of the error value with respect to the last layer's weight values. Calculating the gradients of the second-to-last layer requires using these derivatives, which we refer to as gradients. We keep doing this until every single weight in our neural network has gradients. Then, to lower the error value, we subtract this gradient value from the weight value. By doing this, we descend toward the Local Minima, which represents the least loss.[40]

2.3.2.3 The loss function

We explained that weights and biases are the parameters that need to be changed for the network to learn, these issues are indirectly answered by implementing a loss function, a mathematical equation that connects every possible network variables into one value expressing the distance from the ideal performance, or how much the system is "losing" from its ideal state. The loss function takes on a relatively large value while the network is inaccurate and slowly reduces as the system learns, eventually reaching zero at the end of training. We can extract the gradients needed to update the weights from the loss function. The cost is determined by taking the average of all losses.[40]

2.3.2.4 Gradient decent

Gradient descent is an optimization algorithm used to minimize a cost or loss function in machine learning and deep learning. It works by iteratively adjusting the model parameters in the direction of steepest descent of the cost function. The algorithm involves calculating the gradient of the cost function with respect to the parameters and updating the parameters by taking a step in the direction of the negative gradient. There are two main variants of gradient descent, batch gradient descent and stochastic gradient descent. Gradient descent can converge to a local minimum and can be sensitive to the choice of learning rate, but various extensions and improvements have been developed to address these issues.

Think of the graph below as you going along, and the 'green' dot on the graph represents where you are right now. You try to get to the red dot, the minimum, but from where you are standing, you can't see it.[50]

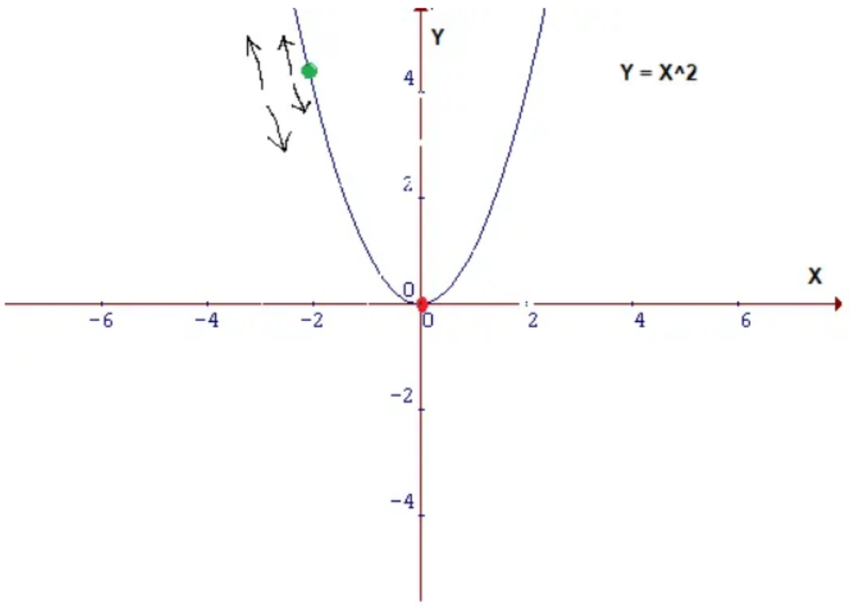


Figure 2.6: find The Minimum Value [50]

To achieve the minimum, you essentially need to know two things: which direction to move in and how large of a step to take.

With the use of derivatives, the Gradient Descent Algorithm enables us to make these judgments quickly and effectively. A derivative is a calculus term that is computed as the graph's slope at a certain position. By adding a tangent line to the graph at the slope's point. Therefore, if we can determine this tangent line, we may determine the preferred path to take to obtain the the minima.

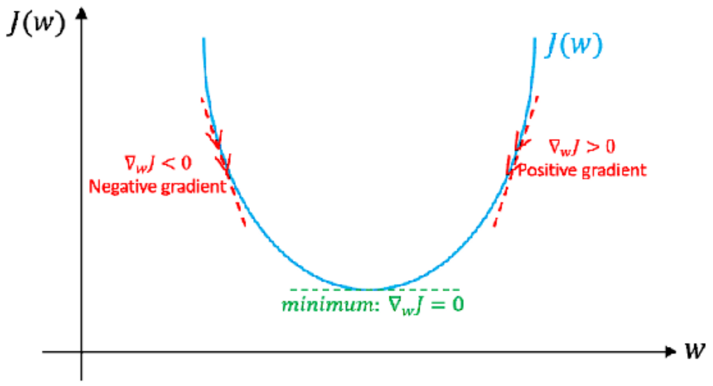


Figure 2.7: Gradien descent illustration [59]

2.4 Convolutional Neural Network (CNN)

The visual cortex of the human vision system serves as a biological model for CNN, which uses it to identify subregions from an input image. Convolutional neural networks (CNN) are a structured version of multilayer neural networks. To learn from the input data in fully connected networks, a relatively high number of parameters are utilized. As an example, a network of 270,000 parameters would be created during the learning stage for color (RGB) images of width and height of 300 pixels. These networks have a complex structure. However, CNNs can use the convolution operation to significantly reduce the number of parameters by applying a kernel or filter window to the input images [51].

CNN is divided into three main parts

1. Input layer

At the input layer, the input data is represented as a tensor with four dimensions that represents the number of images, image width, image height, and channels.

where there are one channel for grayscale images and three channels for color images.[27]

2. Learning layer or convolutional layer

The core part of a CNN is a convolutional layer. With a weight matrix that is organized in a grid called a filter or kernel and a region of neurons from the input layer, the layer performs a dot product and produce one output value for the region [51]. In order to learn various kinds of characteristics during the learning stage, several convolution layers are often interconnected. As shown in figure 2.8.

3. Classification layer

Each class is represented by a neuron in this fully connected layer. It outputs the probability that this input belongs to this class. it may be conceptualized mathematically as a function that transforms an input vector with dimension I into an output vector with dimension O .Usually the layer has a bias parameter. [27]

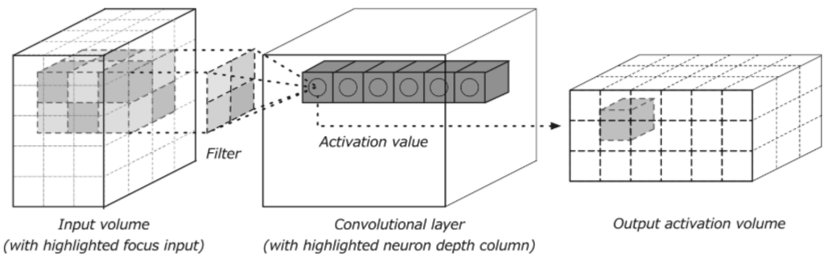


Figure 2.8: Convolution layer [51]

CNN's learning layers and output layer may additionally comprise additional layers, such as:

2.4.1 Layers

2.4.1.1 Activation Layers

Every form of activation function in every type of neural network's main role is mapping the input to the output. The weighted summation of the neuron input and its bias (if present), together with other factors, is computed to produce the input value. By producing the proper output, the activation function, with connection to a certain input, decides whether or not to release a neuron. utilizing CNN's architecture. The activation layers' nonlinear behavior indicates that the input-to-output mapping will also be nonlinear.

Additionally, these layers enable the CNN to learn very complex things. As a crucial characteristic that enables error back-propagation to be utilized to train the network, the activation function must also have the capacity to differentiate.

Since it is assumed that an activation comes right after a convolution, activation layers are sometimes omitted from network architecture diagrams even though they aren't technically "layers" (since no parameters or weights are learned inside an activation layer).

2.4.1.2 Pooling Layers

In between each set of convolutional layers, pooling layers are added. The spatial size of the data representation (width and height) is decreased. This method is used to prevent the network from being overfit. The representation of the data also involves down sampling. A 2x2 pool with stride 2 is the typical size. There are two primary forms of pooling: maximum pooling and average pooling. In contrast to average pooling, which uses the mean of the block, maximum pooling uses the block's maximum value.

2.4.1.3 Batch Normalization

This layer quickly became very popular mostly because it helps to converge faster [14]. It includes a normalization step to make the inputs of each trainable layer comparable across features (moving inputs to zero-mean and unit variance). This keeps the network learning while ensuring a high learning rate.

Additionally, it prevents activation functions like TanH and Sigmoid from getting stuck in saturation mode (for example, with a gradient of 0).

Several CNN architectures have been developed for image processing based on various combinations of the mentioned layers, including VGG16[5], ResNet[21], Xception[22], and others[23].

2.4.2 Convolutional Architectures

Since the 1990s, many convolutional architectures have been created. The most widely recognized architectures are included in this section.

2.4.2.1 CNNs (LeNet)

LeNet

One of the first CNNs, used to recognize the digits in handwritten numbers. consists of two fully connected layers followed by two convolutional layers.

2.4.2.2 Deep CNNs

AlexNet

A deep CNN with eight layers won the 2012 ImageNet Large Scale Visual Recognition Challenge. improved performance by dropout, ReLU activation, and data augmentation.

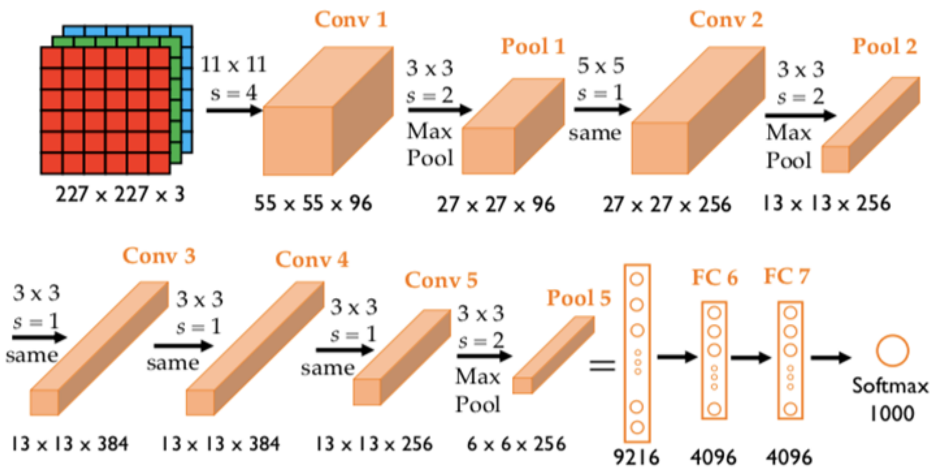


Figure 2.9: AlexNet Architecture [54]

Overfeat or ZFNet

Overfeat and ZFNet are both neural network architectures designed for image recognition tasks. Overfeat was introduced in 2013 and has five convolutional layers followed by three fully connected layers, while ZFNet, also introduced in 2013, has five convolutional layers and two fully connected layers. Both architectures achieved state-of-the-art performance in computer vision benchmarks at the time of their introduction.

2.4.2.3 Very Deep CNNs

VeryDeep or VggNet

"Inception modules" are used by a CNN with a unique architecture to handle input in different scales. 2014 ImageNet competition champion.

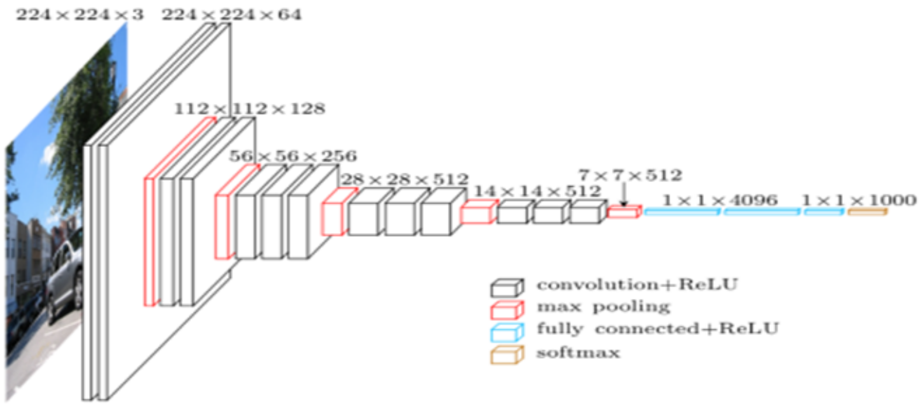


Figure 2.10: VGG architecture [62]

GoogLeNet or Inception

"Inception modules" are used by a CNN with a unique architecture to handle input in different scales. 2014 ImageNet competition champion.

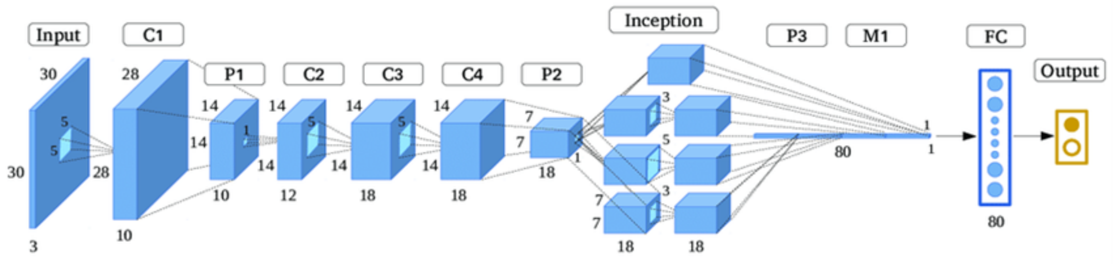


Figure 2.11: Googlenet architecture [24]

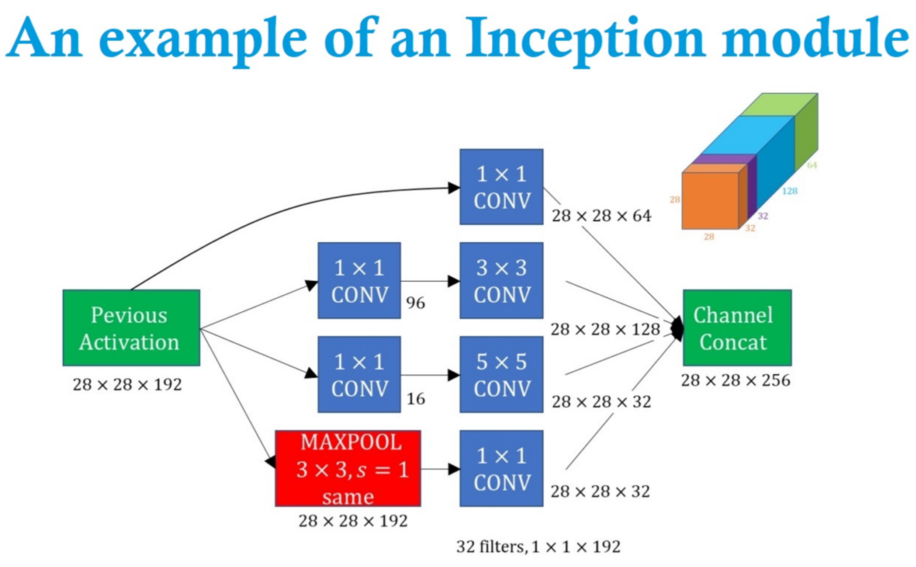


Figure 2.12: Inception module [15]

2.4.2.4 Residual CNNs

ResNet

a highly deep CNN that solves the vanishing gradients issue by using "residual connections". has won several events and may have up to 152 layers.

2.4.2.5 Other architectures

DenseNet

a CNN with very dense connections between each layer and every other layer via the use of feed-forward connections. achieves cutting-edge outcomes with fewer parameters.

MobileNet

a CNN with a smaller model size and less computing power optimized for mobile and embedded vision applications. reduces parameters by using depthwise separable convolutions.

EfficientNet

a family of CNNs that adapts the number of parameters to the required model size automatically. uses fewer parameters than earlier models to achieve state-of-the-art performance on ImageNet.

2.4.3 Training CNNs

2.4.3.1 CNN Operations

Convolution

A mathematical process called convolution combines two kinds of data. A CNN's definition of a convolution operation is a feature detector. It uses a kernel, or grid of weights, to multiply an area from an input image or neurons from the previous layer, and produces convoluted features as output.[51]

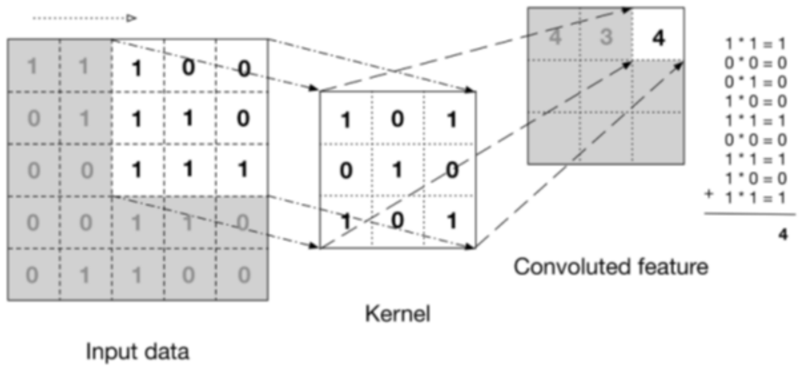


Figure 2.13: Convolution operation [51]

Kernel or filter

It is a three-dimensional tensor that contains the weights used during the convolution process. Throughout the learning stage, its values are updated. According on the feature details desired in the photos, the kernel size is selected. Small kernel sizes like 3x3 or 5x5 are used when looking for localized features, while large kernel sizes like 9x9 or 11x11 are used when looking for generalized features.[27]

Zero-padding

It's used in the convolution layer. Around the input tensor, it adds zero values. Additionally, it enables the dot product between every input pixel and kernel.[27]

Stride

It specifies the movement of the kernel window for each convolution operation. The kernel window is moved one per unit when stride is equal to 1, and so on. When the stride value is low, there is more overlap between the columns in the output layer, leading to greater output. On the other hand, when a greater stride is employed, overlapping is reduced and the amount of output decreases.[27]

Flattening

The flatten operation in a CNN refers to the process of reshaping the output from the convolutional/pooling layers into a 1D vector, which can then be fed into a fully connected layer for classification or regression tasks.

For example, if the output from the last pooling layer in a CNN is a 7x7x64 tensor, the flatten operation would reshape it into a 1D vector of length 3136 (7 x 7 x 64 = 3136), which can then be fed into a fully connected layer for classification or regression. The flatten operation is a common way to transition from the convolutional/pooling layers to the fully connected layers in a CNN.

2.4.3.2 Overfitting and underfitting

Machine learning confronts two challenges: overfitting and underfitting. When the model is unable to generate low error values in the training set, underfitting occurs. When the difference between the training error and the test error is too great, overfitting occurs. When a model is overfit, it memorizes the characteristics or features of the training set's data and may not perform well with the test set.

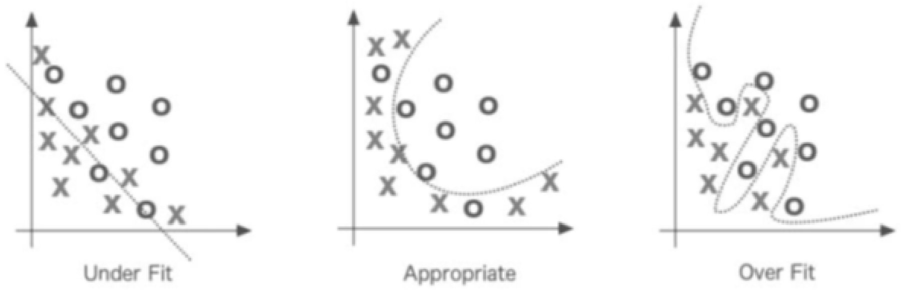


Figure 2.14: Underfitting and Overfitting in ML[51]

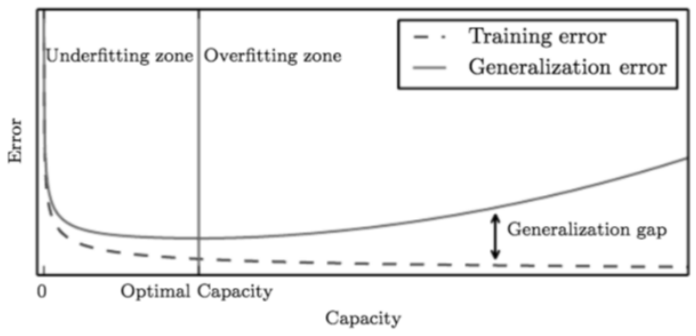


Figure 2.15: Capacity of model [27]

2.4.3.3 Regularization

How to generalize a model such that it does not only perform well on the training set but also on the unseen data is one of the major issues in the machine learning . Regularization is a method for addressing problems with model generalization. Regularization makes attempts to reduce test error, which might increase error in the training set. The goal of regularization is to increase bias while keeping variation under control. For instance, regularization decreases the weights of a deep learning model when the weights grow to be too big by applying L1 or L2 norm.[27]

There are several approaches to regularization. Below we go through early stopping and drop out, two common techniques for regularization in deep learning applications :

Early stopping

Early stopping is a simple hyperparameter that is defined throughout the model-building process. If the validation error does not decrease after the required number of iterations, the parameter requires the training process to end [23]. This method prevents the model from being overfit.[27]

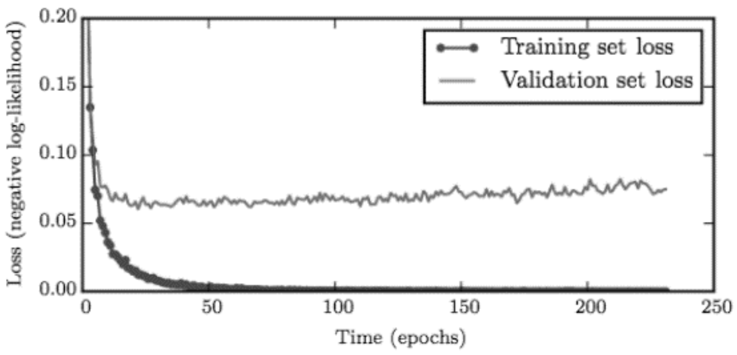


Figure 2.16: Learning curves [23]

Dropout

Dropout is a method of avoiding utilizing all of the network's neurons so that the model does not remember every feature. It is accomplished by multiplying a few neurons in the output of previous layers that were randomly chosen by 0 [23].Using a dropout in the input layer is not recommended since it might result removing parts in the original input dataset [51].

2.4.4 Optimization algorithms

An optimization technique that helps a deep learning model perform better is called an optimizer algorithm. The performance and training speed of the deep learning model are significantly impacted by these optimization techniques or optimizers.

The weights for each epoch must be changed, and the loss function must be minimized, while the deep learning optimizers model is being trained. An optimizer is a process or approach that changes the neural network's properties, such as its weights and learning rates. [25]

2.4.4.1 Data augmentation

Data augmentation is a technique for increasing the size of the training set such that the model cannot learn all of it, by making modified copies of a dataset using existing data. It involves making tiny adjustments to the dataset or creating new data points using deep learning.

Original data is the source of augmented data, with a few small modifications. To increase the size and variety of the training set in the case of image augmentation, we apply geometric and color space modifications (flipping, resizing, cropping, brightness, and contrast). [10]

• **Image Augmentation**

1. Geometric transformations

randomly flip, crop, rotate, stretch, and zoom images. You need to be careful about applying multiple transformations on the same images, as this can reduce model performance.

2. Color space transformations

randomly change RGB color channels, contrast, and brightness.

3. Kernel filters

randomly change the sharpness or blurring of the image.

4. Random erasing

delete some part of the initial image.

5. Mixing images

blending and mixing multiple images.

2.4.4.2 Stochastic Gradient Descent (SGD)

SGD algorithms have shown their efficiency in optimizing huge deep learning models. Instead of using the whole data set for each iteration, a few samples are randomly selected since the word stochastic refers to a system or a procedure associated to a random possibility. Changing the network topology after each training step is how SGD attempts to determine the global minimum. Instead of determining the gradient for the whole dataset, this method simply decreases the error by estimating it for a batch that was selected at random. The dataset is really shuffled at random, and batches are moved through in a series of steps to do random sampling. The objective function may significantly vary thanks to the SGD's frequent high-variance adjustments. [26]

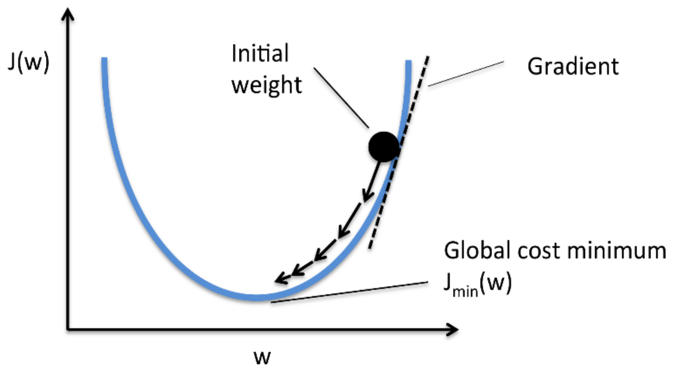
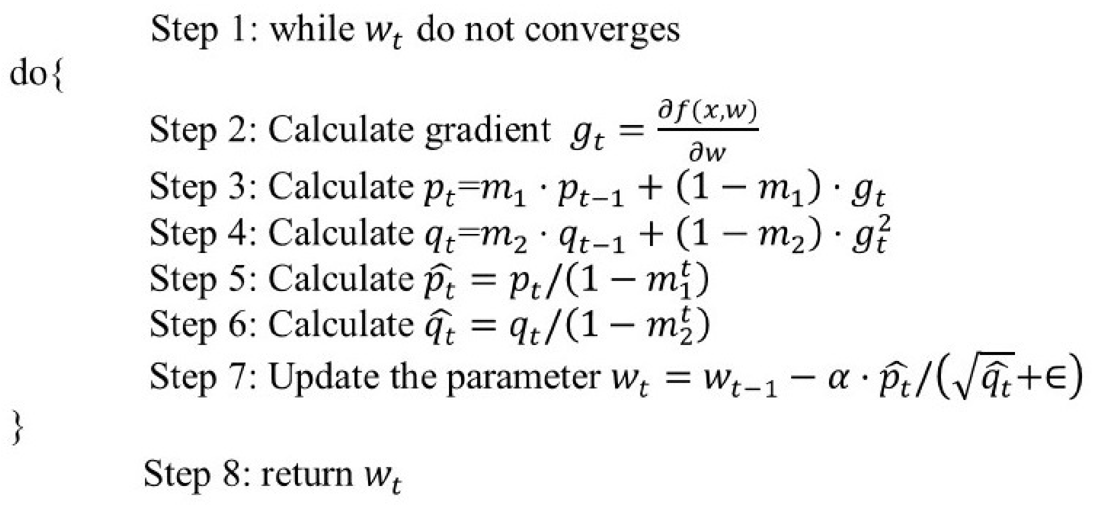


Figure 2.17: Stochastic Gradient Descent (SGD)[53]

2.4.4.3 Adaptive Moment Estimation (Adam)

Adam evaluates each parameter's adaptive learning rates and is an SGD optimization technique. The step-size strategy known as Adam is one of the most popular in the neural network community. Adaptive Moments served as the inspiration for the name. RMSProp and Momentum are combined in it. A bias adjustment technique is offered by the upgrade procedure, which takes into account the smooth gradient variation. Adam is less memory-intensive to execute, has lower processing costs, and is resistant to gradient diagonal rescaling. The learning rate is not treated as a hyperparameter by RMSprop, a gradient-based optimizer, but rather as an adaptive learning rate (LR) that changes over time.[26]



```
        Step 1: while $w_t$ do not converges
do{
        Step 2: Calculate gradient $g_t = \frac{\partial f(x,w)}{\partial w}$
        Step 3: Calculate $p_t = m_1 \cdot p_{t-1} + (1 - m_1) \cdot g_t$
        Step 4: Calculate $q_t = m_2 \cdot q_{t-1} + (1 - m_2) \cdot g_t^2$
        Step 5: Calculate $\widehat{p_t} = p_t / (1 - m_1^t)$
        Step 6: Calculate $\widehat{q_t} = q_t / (1 - m_2^t)$
        Step 7: Update the parameter $w_t = w_{t-1} - \alpha \cdot \widehat{p_t} / (\sqrt{\widehat{q_t}} + \epsilon)$
}
        Step 8: return $w_t$
```

Figure 2.18: Adam algorithm [22]

2.5 Image Segmentation using CNNs

Image segmentation is a widely used method and analysis in digital image processing to divide a picture into various sections, often depending on the properties of the pixels in the image. The technique of segmenting an image involves dividing the foreground from the background or grouping areas of pixels based on their shared color or form. For example, a most common application of image segmentation in medical imaging is to detect and label pixels in an image of a 3D volume that represent a tumor in a patient's lung or other organs.

Other practical application of image segmentation range from filtering of noisy images, medical applications, Locate objects in satellite images, Object detection and Recognition Tasks, Automatic traffic control systems and Video surveillance, etc [30]

2.5.1 Semantic segmentation

the study of an image's infinite detail. Each image pixel is examined, and a distinct class label based on the texture it illustrates is assigned. For instance, Figure 1 shows a scene with two cars, three pedestrians, a road, and the sky. Similar to how the three pedestrians do, the two cars show a similar texture.For each of these textures or categories, semantic segmentation would assign a different class label. The two cars and three pedestrians cannot be distinguished or counted independently in the output of semantic segmentation. Semantic segmentation tools like SegNet, U-Net, DeconvNet, and FCNs are frequently employed. [34]

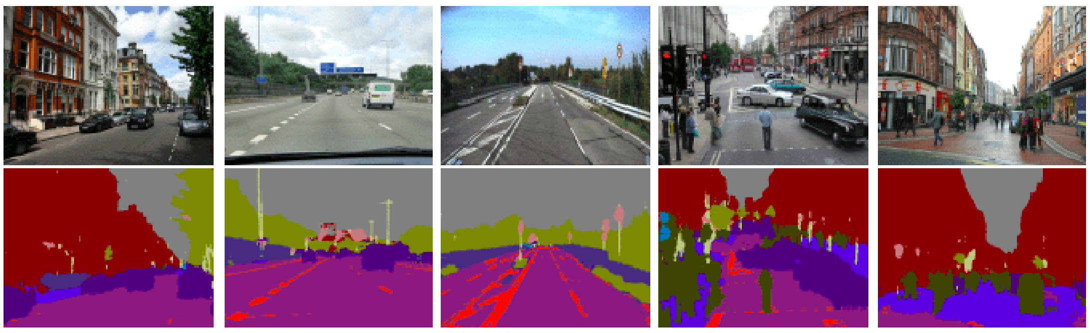


Figure 2.19: Semantic segmentation examples (source: SegNet) [11]

2.5.2 Instance segmentation

usually handles activities that include countable items. It has the ability to identify every item or instance of a class that is present in an image and gives it a unique mask or bounding box. [34]

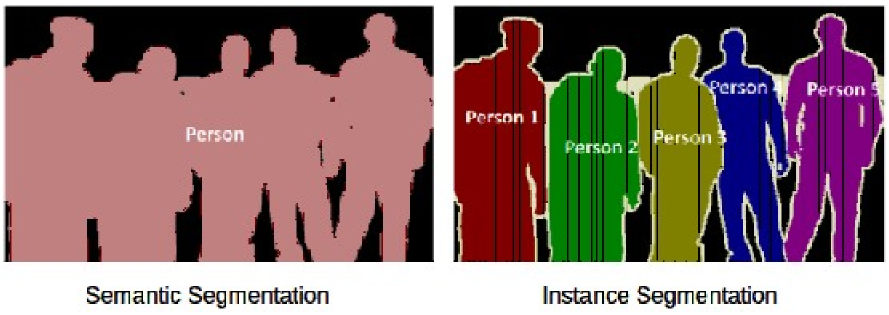


Figure 2.20: Semantic VS Instance segmentation [61]

2.5.3 Evaluation metrics

Each segmentation method evaluates the expected masks or IDs in a scene using a different set of evaluation measures. This is due to the diverse ways that things and items are processed.

The Jaccard Index, also known as the Intersection over Union (IoU) metric, which measures how close the anticipated and actual masks are, is typically used in semantic segmentation. It regulates how much of the two masks' surface overlaps. In addition to IoU, we can also undertake a more thorough review using the dice coefficient, pixel accuracy, and mean accuracy measures. Labels on objects are not taken into account by these measures.

On the other hand, Average Precision (AP) is the preferred evaluation statistic for instance segmentation. For each instance of an item, the AP metric employs the IoU on a pixel-by-pixel basis.[34]

2.6 Object detection using CNNs

The recognition, detection, and localisation of numerous visual occurrences of objects in an image or a video are made easier with the use of object detection techniques. Compared to just basic object classification, it offers a far better understanding of the object as a whole. The number of instances of distinct items can be counted using this technique, and their specific locations can also be marked along with labeling. We can now employ this method for real-time use cases with a huge improvement in speed over time. It provides a comprehensive response to the query, "What object is where and how much of it is there?"

Every object will have its own characteristics, which is the fundamental idea driving this approach. These characteristics can assist us in separating certain objects from others. Those features are used in object detection methodology to categorize the items. Things like face detection, fingerprint detection, etc. use the same principle.[73]

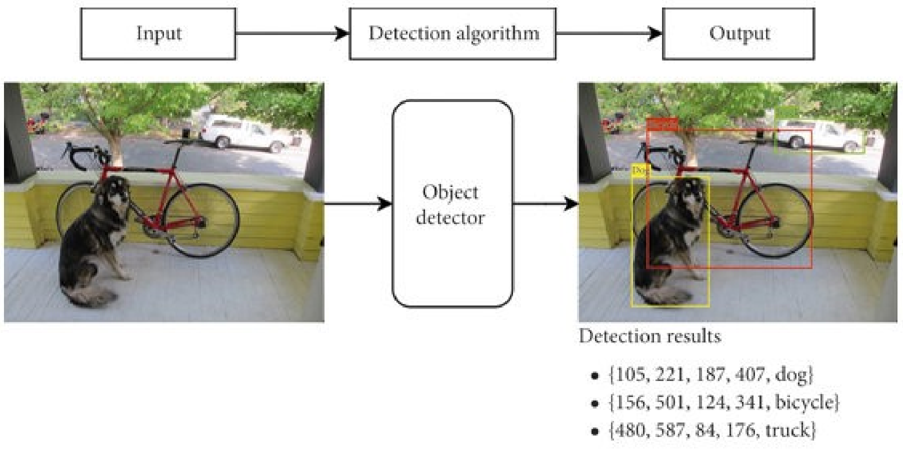

Input
Detection algorithm
Output
Object detector
Dog
Detection results
• {105, 221, 187, 407, dog}
• {156, 501, 124, 341, bicycle}
• {480, 587, 84, 176, truck}

Figure 2.21: Object detection example [4]

2.6.1 Object detection algorithms

Region Propositions (R-CNN, Fast R-CNN, Faster R-CNN) The region proposal layer outputs bounding boxes around the image's objects as part of the region proposal network in this method. Through this process, the image is divided into a few superpixels and combined near the area. [64]

YOLO: You Only Look Once This real-time method aids in the recognition of diverse items in images. Regression is a technique that this algorithm use to help determine class probabilities for the topic image. It operates by dividing the image into N grids with equal-sized SxS squares.[31]

2.6.2 Evaluation metrics

Object detection models are evaluated using a variety of metrics, including mAP, IoU, accuracy, recall, precision, and F1 score.

Mean Average Precision (mAP) is a metric for evaluating object detection models. It is calculated by averaging the precision-recall curves for each object class. A higher mAP indicates that the model is better at detecting objects. [49]

Recall is a metric for evaluating object detection models. It measures the fraction of objects that the model correctly detects. A higher recall indicates that the model is better at detecting all of the objects in an image or video.[49]

Precision is a metric for evaluating object detection models. It measures the fraction of objects that the model correctly detects and classifies. A higher precision indicates that the model is better at avoiding false positives.[49]

F1 score is a metric for evaluating object detection models. It measures the balance between precision and recall. A higher F1 score indicates that the model is better at both detecting objects and avoiding false positives.[49]

The choice of metric depends on the specific application. For example, if the goal is to identify as many objects as possible, then recall may be a more important metric than precision. If the goal is to avoid false positives, then precision may be a more important metric than recall.

2.7 Image classification using CNNs

The process of classifying an entire image is known as image classification. Images are anticipated to have just one class per image. Models for image classification take an image as input and produce a prediction of the class to which the image belongs.

The final output of an image classifier, which can be either a single class or a probability of classes that best characterize the image, is obtained by passing the numerical pixel values of an image through the CNN. This is how the procedure appears:

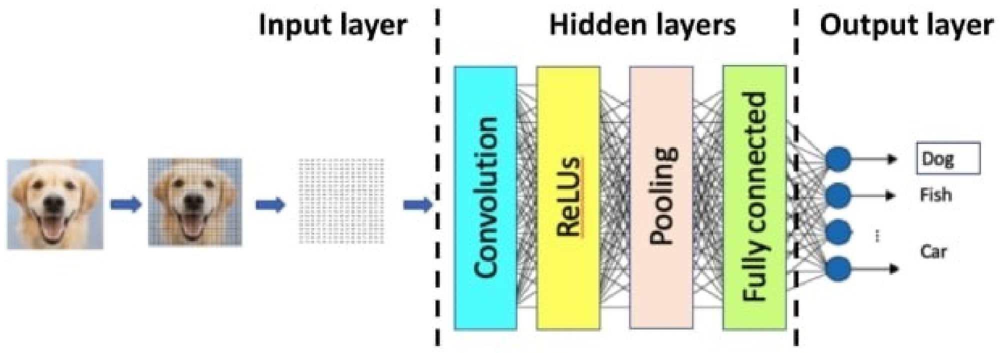


Figure 2.22: How a CNN classifies an image [3]

The "magic" takes place inside the CNN's secret layers. Each CNN layer type completes a particular function on the path from the input of a picture to the output of a class.[3]

2.8 Transfer Learning

The capacity of a system to detect and use information and abilities obtained in earlier activities to new tasks (or other domains).

Transfer learning is often used in deep learning when there is an absence of labeled data for a new task. With transfer learning, a pre-trained model's expertise may be utilized to fine-tune the new data and use the limited labeled data for the job. also The performance of a pre-trained model that was developed for a job that is comparable to the new task may be enhanced by fine-tuning it using the new data. On new data set of various dog breeds, for example, a model that was trained on a dataset of dogs and cats may be adjusted.[71]

2.8.1 Fine-tuning

It is a way of applying or utilizing transfer learning.There are a number of fine tuning techniques that can be used to improve the performance of a pre-trained model on a new task [14]. Some of the most common techniques include:

Freezing layers: When fine tuning, you can choose to freeze some of the layers of the pre-trained model. This means that the weights of these layers will not be updated during training. Freezing layers can help to prevent the model from overfitting to the new data.

Unfreezing layers: If you have frozen some of the layers of the pre-trained model, you can choose to unfreeze them during fine tuning. This means that the weights of these layers will be updated during training. Unfreezing layers can help the model to learn the new data and adapt to the new task.

Changing the learning rate: The learning rate is a hyperparameter that controls how much the weights of the model are updated during training. When fine tuning, you may need to change the learning rate to improve the performance of the model.

2.9 Conclusion

In this chapter we presented the basics of deep neural netwok, specifically convolutional neural network, as we have highlight on its characteristics and the common methods in this feild.

Chapter 3

Experiments and Results

This chapter will cover our model's application to the u_ net architecture from an implementation perspective.For a better understanding, it is first necessary to provide a brief description of the data set that was used. Then, we will discuss the working environment that was employed, the API, and the programming language.moreover the u _ net architecture will be detailed in order to show our proposed model with assessment and results.Finally, after completing an experiment in terms of hyperparameters and optimizers to determine the best model, it is crucial to compare the models to determine how effective it is the chosen one.

3.1 Data set description (Artificial Lunar Landscape Dataset)

This dataset was created by Romain Pessia and Genya Ishigami of the Space Robotics Group, Keio University, Japan. Because lunar images are rare and frequently lack annotation, it can be challenging to run any kind of machine learning experiment on them. This dataset works to give a sample of fake but realistic lunar landscapes to the general public, which may be used to train algorithms for the identification of rocks. On real lunar images or other images of rocky terrain, those trained algorithms are able to go to the test. Currently, the dataset has 9,766 realistic renderings of rocky lunar landscapes along with their segmented counterparts (the three classes are the sky, smaller rocks, and larger rocks). It also includes processed, cleaned-up ground truth photos and a database of bounding boxes for all bigger rocks. [5]

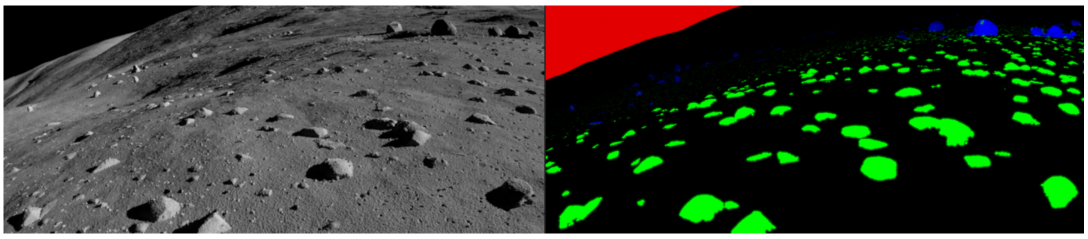


Figure 3.1: Artificial Lunar Landscape Dataset

3.2 The used environment

Google Colab

A Jupyter notebook-based runtime environment called Google Colab enables you to execute programs fully in the cloud. This is important because it allows you to train large-scale ML and DL models even without a powerful computer or high-speed internet connectivity. Due to the computing boundaries of local computers, Google Colab offers both GPU and TPU instances, making it the ideal tool for deep learning and data analytics enthusiasts. A Colab notebook is appropriate for both personal and professional use since it can be accessed remotely via a browser from any device.[16]

3.3 Programming Language

Python

A powerful programming language is Python. Its ecosystem of libraries, frameworks, and tools is expanding. These programs and libraries include prewritten patterns that enable users to execute a wide range of tasks without having to spend a lot of time creating new code. Python has created an emphatic place for itself as the industry has focused on building for the future where data plays a central role, and Python, with its prowess, has become the first language of choice for everyone. AI and ML professionals. [52]

3.4 Implementation tools(API)

3.4.1 Tenserflow

is a complete, open-source deep learning framework that Google developed and released in 2015. It is known for its support for many abstraction layers, scalable production and deployment options, and compatibility for several platforms, including Android.

The most effective usage of TensorFlow, a symbolic math framework for neural networks, is for dataflow programming across a variety of tasks. It provides a range of abstraction levels for model construction and training.[69]

3.4.2 Keras

Google created the high-level Keras deep learning API to build neural networks. It is used to make the implementation of neural networks simple and is developed in Python. Additionally, multiple backend neural network computation is supported. [70]

Between Keras and Tenserflow

A high-level neural network framework called Keras works on top of TensorFlow, which is an open-sourced end-to-end platform and library for various machine learning applications. Both provide high-level APIs for quickly constructing and training models, while Keras is more approachable because to its Python integration.[32]

3.4.3 Matplotlib

To display data, the scientific charting package matplotlib is often used. Importantly, data analysis requires visualization. Histograms, scatter plots, lines, etc. may all be plotted.[33]

3.4.4 Numpy

In simple terms, NumPy offers n-dimensional array objects. Additionally, NumPy offers mathematical functions that can be applied to numerous calculations. [33]

3.4.5 Sklearn

Tools for data analysis and data mining are provided by scikit-learn, which is based on NumPy, and matplotlib. It includes methods for classification and clustering that are built-in, as well as training datasets like the iris dataset, the Boston housing market dataset, the diabetes dataset, etc.[33]

3.4.6 OpenCV

A Python package called OpenCV makes it possible to execute out image processing and computer vision tasks. It offers a variety of capabilities, including as tracking, facial recognition, and object detection. [68]

3.5 U-Net Architecture

The U-Net architecture, which was initially released in 2015, has completely changed the state of deep learning. The design easily succeeded in several categories of the 2015 International Symposium on Biomedical Imaging (ISBI) cell tracking challenge. One of their projects was the segmentation of neural structures in transmitted light and electron microscopy stacks.

With this U-Net architecture, segmentation of images with a size of 512X512 can be computed quickly on a modern GPU. Due to its incredible success, this architecture goes through multiple variations and modifications.[72]

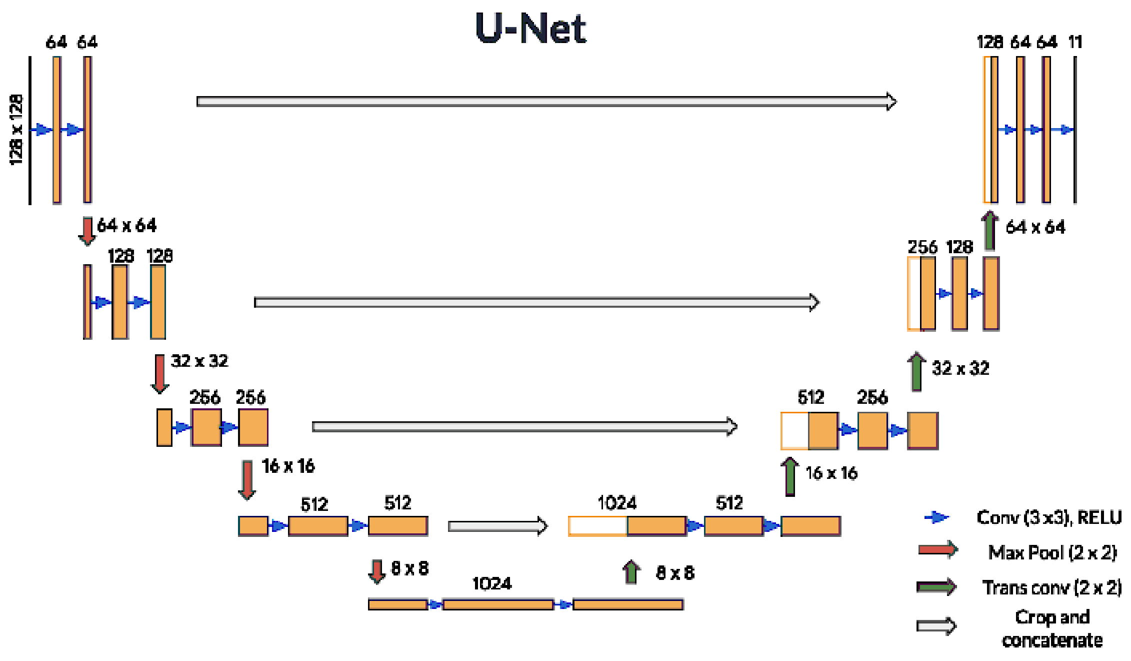


Figure 3.2: U-Net architecture [7]

3.5.1 Encoder

The first part of the architectural diagram is the encoder. In order to encode the input picture into feature representations at many different levels, it is often a pre-trained classification network like VGG/ResNet where convolution blocks are applied followed by a maxpool downsampling. [2]

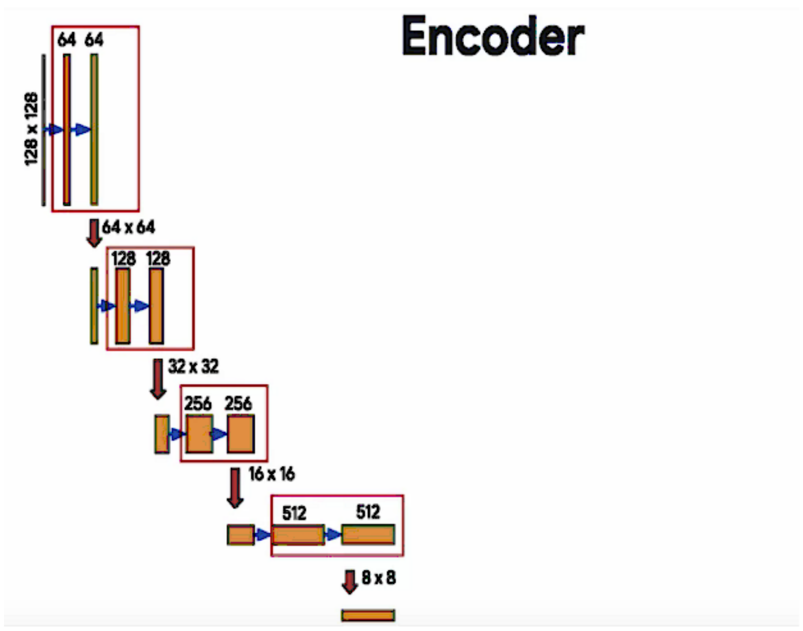


Figure 3.3: Illustration of the Encoder [7]

3.5.2 Decoder

The architecture's second component is the decoder. The objective is to get a dense classification by semantically projecting the discriminative features (lower resolution) learned by the encoder around the pixel space (higher resolution). Upsampling, concatenation, and standard convolution processes make up the decoder. [2]

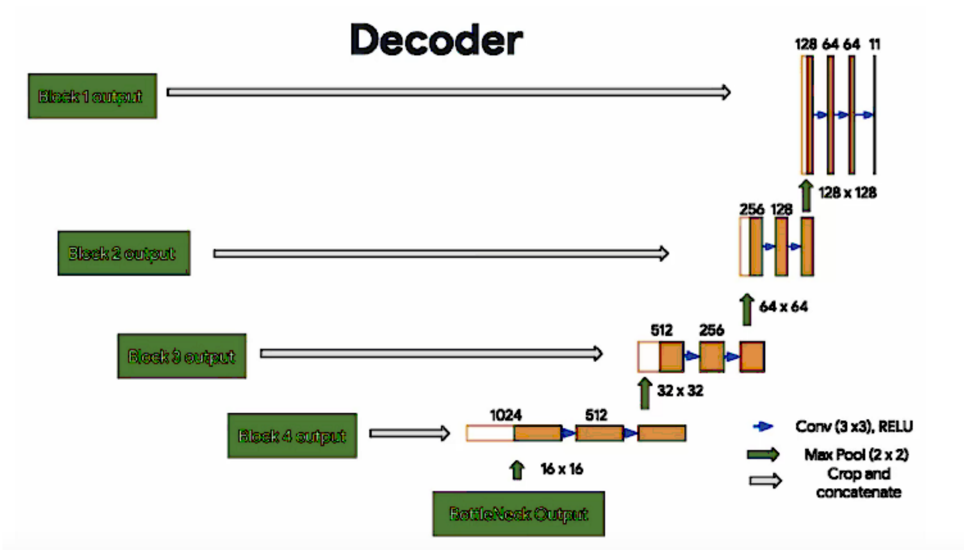


Figure 3.4: Illustration of the Decoder [7]

3.5.3 Skip connections

In order to maintain the loss from previous layers and make them more strongly represent the total values, this skip connection is an essential concept. Additionally, it has been demonstrated scientifically that they lead to faster model convergence and better results.[72]

3.6 Experiments And Implementation

3.6.1 State-of-the-Art Approaches And Results

The paper entitled "Image Segmentation and Object Detection of Lunar Landscape" by Chen Wu and Ziyang Yuan presents an approach for segmenting and detecting objects in lunar images. The authors employ image segmentation techniques to partition the lunar landscape images into distinct regions or segments. This step aims to separate different objects or regions of interest from the background. (they used FCN-8s and DeepLabv3)) After segmenting the images, they focus on object detection within the segmented regions. Object detection involves identifying and localizing specific objects of interest within an image. (they used YOLOv2 algorithm,Adam optimizer and mean Average Precision (mAP) for evaluation)

3.6.2 Semantic Segmentation Models

While the state-of-the-art approaches for the task involved the utilization of DeepLab and FCN architectures, our objective was to explore and evaluate alternative architectures and models. The models were basically based on the UNET architecture. Autoencoders have also been tested, and finally SAM.

3.6.2.1 Basic U-Net Architecture

Our experimentation began with the adoption of the UNet architecture. The table bellow provides a complete overview of the layers in this architecture, including their output shapes, the number of parameters, and the connections to previous layers.

Layer	Output Shape	Param #	Connected to
input_2	(None, 128, 128, 3)	0	
conv2d_19	(None, 128, 128, 16)	448	input_2
dropout_9	(None, 128, 128, 16)	0	conv2d_19
conv2d_20	(None, 128, 128, 16)	2320	dropout_9
max_pooling2d_4	(None, 64, 64, 16)	0	conv2d_20
conv2d_21	(None, 64, 64, 32)	4640	max_pooling2d_4
dropout_10	(None, 64, 64, 32)	0	conv2d_21
conv2d_22	(None, 64, 64, 32)	9248	dropout_10
max_pooling2d_5	(None, 32, 32, 32)	0	conv2d_22
conv2d_23	(None, 32, 32, 64)	18496	max_pooling2d_5
dropout_11	(None, 32, 32, 64)	0	conv2d_23
conv2d_24	(None, 32, 32, 64)	36928	dropout_11
max_pooling2d_6	(None, 16, 16, 64)	0	conv2d_24
conv2d_25	(None, 16, 16, 128)	73856	max_pooling2d_6
dropout_12	(None, 16, 16, 128)	0	conv2d_25
conv2d_26	(None, 16, 16, 128)	147584	dropout_12
max_pooling2d_7	(None, 8, 8, 128)	0	conv2d_26
conv2d_27	(None, 8, 8, 256)	295168	max_pooling2d_7
dropout_13	(None, 8, 8, 256)	0	conv2d_27
conv2d_28	(None, 8, 8, 256)	590080	dropout_13
conv2d_transpose_4	(None, 16, 16, 128)	131200	conv2d_28
concatenate_4	(None, 16, 16, 256)	0	conv2d_transpose_4, conv2d_26
conv2d_29	(None, 16, 16, 128)	295040	concatenate_4
dropout_14	(None, 16, 16, 128)	0	conv2d_29
conv2d_30	(None, 16, 16, 128)	147584	dropout_14
conv2d_transpose_5	(None, 32, 32, 64)	32832	conv2d_30
concatenate_5	(None, 32, 32, 128)	0	conv2d_transpose_5, conv2d_24
conv2d_31	(None, 32, 32, 64)	73792	concatenate_5
dropout_15	(None, 32, 32, 64)	0	conv2d_31
conv2d_32	(None, 32, 32, 64)	36928	dropout_15
conv2d_transpose_6	(None, 64, 64, 32)	8224	conv2d_32
concatenate_6	(None, 64, 64, 64)	0	conv2d_transpose_6, conv2d_22
conv2d_33	(None, 64, 64, 32)	18464	concatenate_6
dropout_16	(None, 64, 64, 32)	0	conv2d_33
conv2d_34	(None, 64, 64, 32)	9248	dropout_16
conv2d_transpose_7	(None, 128, 128, 16)	2064	conv2d_34
concatenate_7	(None, 128, 128, 32)	0	conv2d_transpose_7, conv2d_20
conv2d_35	(None, 128, 128, 16)	4624	concatenate_7
conv2d_36	(None, 128, 128, 16)	2320	conv2d_35
conv2d_37	(None, 128, 128, 3)	51	conv2d_36

Table 3.1: UNet Architecture Summary

Splitting The Data :

the data was split into training and validation sets, 10% of the data will be used for testing, and the remaining 90% will be used for training.

The Implemented Loss Functions

Tversky loss function It is a variant of the Dice coefficient used in image segmentation. It measures similarity between predicted and ground truth masks, allowing adjustment of false positives and false negatives. It is popular in medical imaging and provides a continuous and differentiable loss for neural network training.

The Tversky Index equation:

$$\text{Tversky Index} = \frac{TP}{TP + \alpha \cdot FP + \beta \cdot FN}$$

where:

$$TP : \text{True Positives}$$
$$FP : \text{False Positives}$$
$$FN : \text{False Negatives}$$
$$\alpha, \beta : \text{Weighting parameters controlling importance of FPs and FNs}$$

The Tversky Loss equation:

$$\text{Tversky Loss} = 1 - \text{Tversky Index}$$

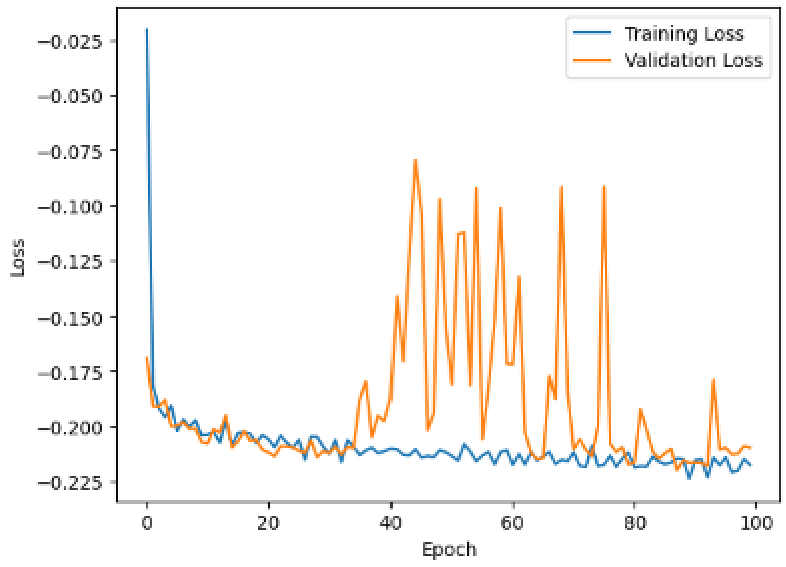


Figure 3.5: Loss Calculated with Tversky loss function

Dice loss function It is used in image segmentation to measure dissimilarity between predicted and ground truth masks. It is derived from the Dice coefficient and is calculated as 1 minus the Dice coefficient.

The Dice coefficient equation:

$$\text{Dice coefficient} = \frac{2TP}{2TP + FP + FN}$$

where:

$$TP : \text{True Positives}$$
$$FP : \text{False Positives}$$
$$FN : \text{False Negatives}$$

The Dice loss equation:

$$\text{Dice Loss} = 1 - \text{Dice coefficient}$$

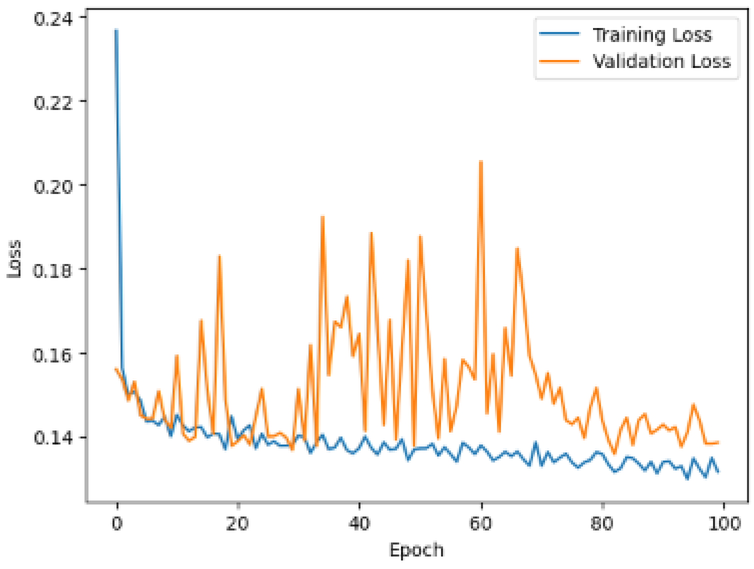


Figure 3.6: Loss Calculated with Dice loss function

Data Augmentation

After training our model on the data for 47 epochs,the model run of data. We suspected that this was because the training set was too small. To address this, we decided to test data augmentation.

`ImageDataGenerator` configuration:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
        rescale=1.0/255.0,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        vertical_flip=True
    )
```

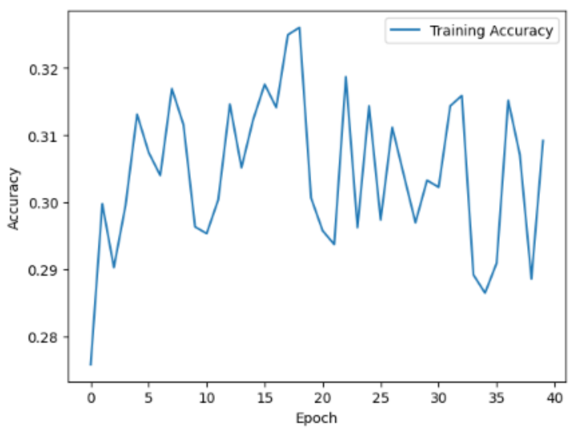


Figure 3.7: Whitout Using Data Augmentation

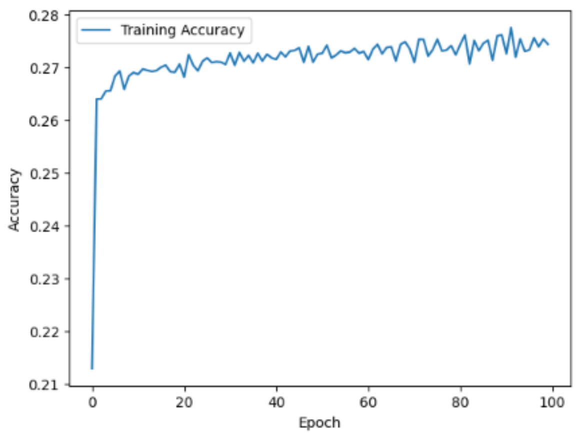


Figure 3.8: Using Data Augmentation

Optimizers/Learning Rate/Batch Size

To test all the possible combinations of the choice of optimizer, learning rate, and batch size, we used a for loop to iterate over all the possible combinations. For each combination, we can train a model and evaluate its performance. so in the end the combination with the best performance can then be used to train the final model.

Batch Size	Optimizers	Learning Rate	number of Epochs	model
64	SGD	0.001	48	U_ NET

Table 3.2: Example of a combination.

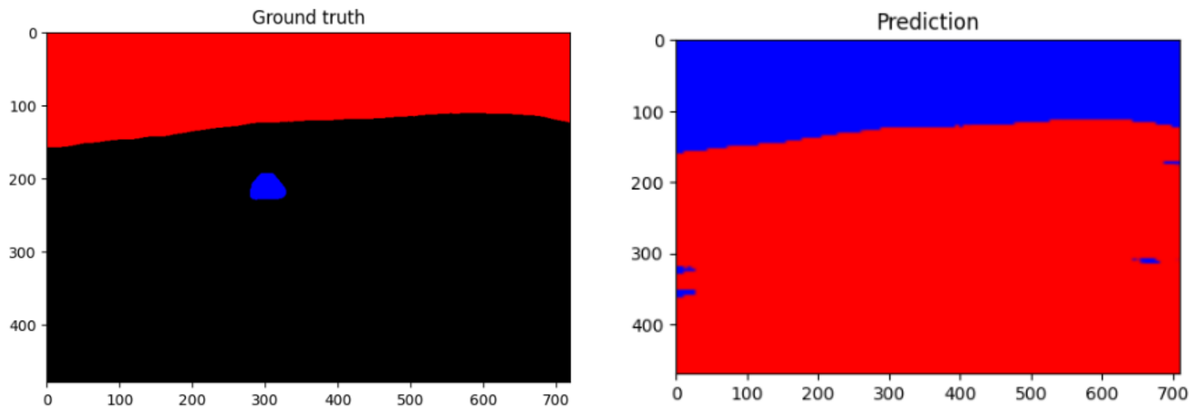


Figure 3.9: The Output Image using Unet

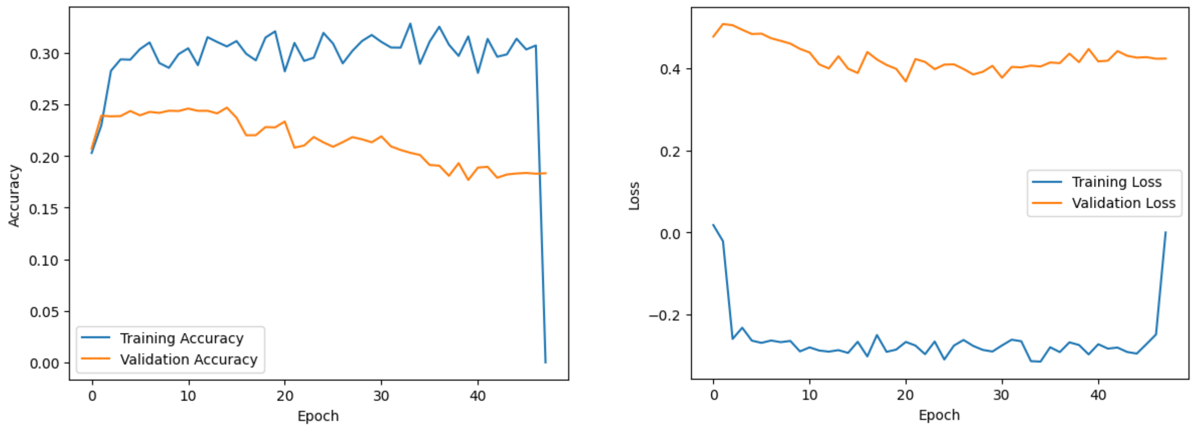


Figure 3.10: The Obtained Results

Observation :
The results were not good, and there are many reasons for this, which we will identify at the end.

Adding Batch Normalisation:

We have added batch normalization to our model in an attempt to make it more stable. Batch normalization is a process that normalizes the inputs to a layer, which can help to prevent the model from becoming unstable. We have tried placing the batch normalization layer in different positions in our model to see if it has any effect on the stability of the model.

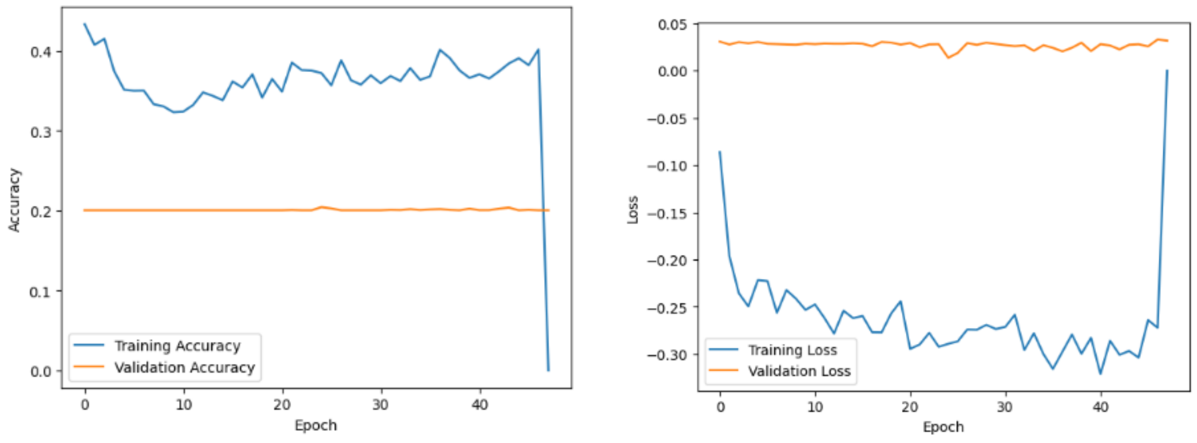


Figure 3.11: The Obtained Results Adding Batch Normalisation

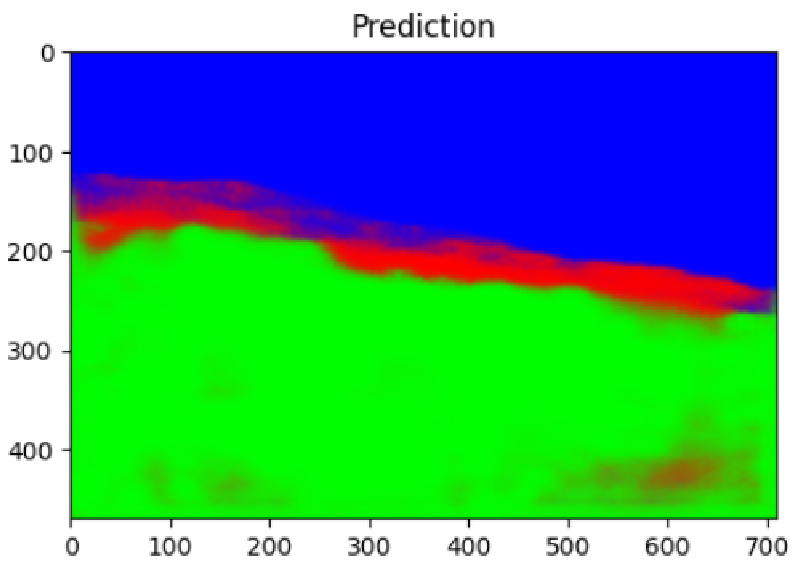


Figure 3.12: Output Image

Clarification:

In addition to the mentioned experiments, several other approaches were attempted to improve the model's performance. These included changing the filters, adding blocks to the U-Net model, using k-fold cross-validation, and applying various preprocessing methods. However, unfortunately, these attempts did not yield satisfactory results.

Certain approaches encountered difficulties due to the limitations imposed by the Colab environment. For example, k-fold cross-validation and resizing the images to a larger size proved challenging due to the amount of data and the complexity of the dataset.

3.6.2.2 Transfer learning using U-Net

Using RESNET50 As a Backbone : Convolutional neural network ResNet-50 has 50 layers total.It is a classic neural network used as a backbone for many computer vision tasks.

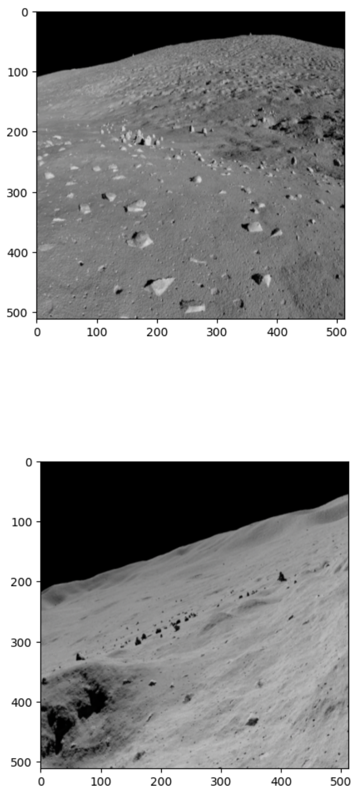


Figure 3.13: Image

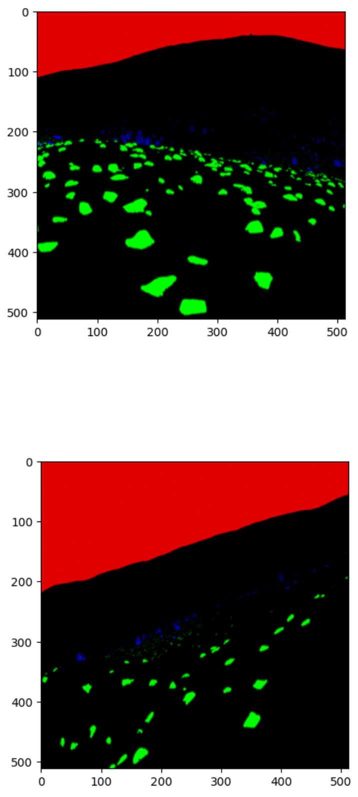


Figure 3.14: Mask

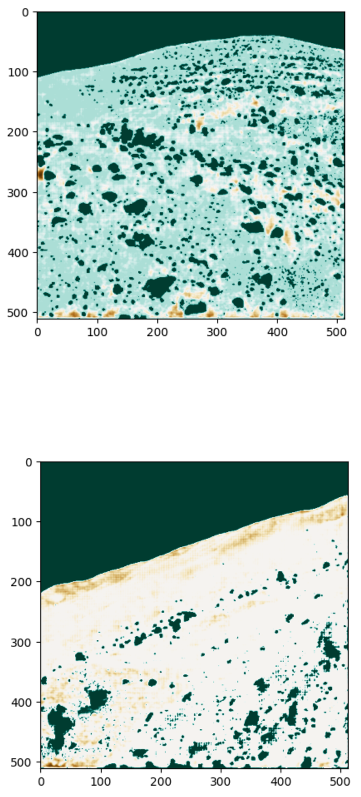


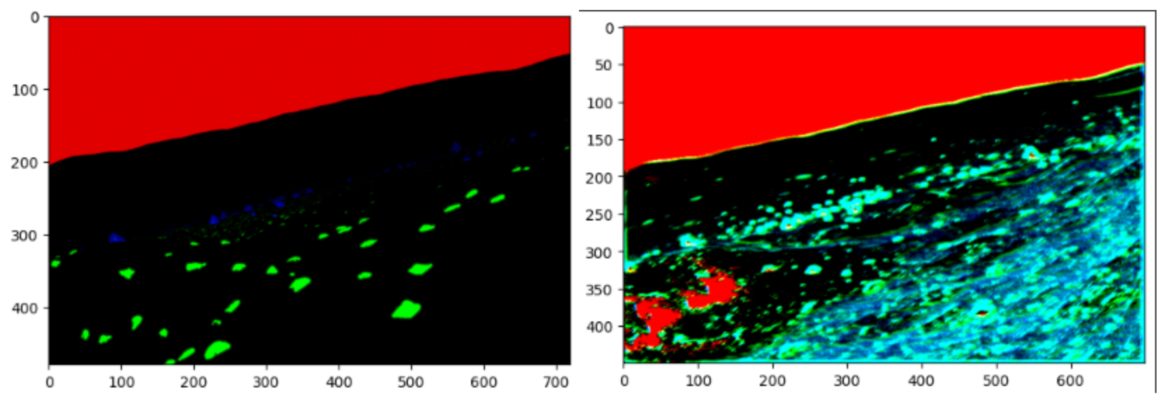
Figure 3.15: Predicted cmap='BrBG'

Figure 3.16: Output Image RESNET50 & Unet

VGG16 as a Backbone : VGG16 is a 16-layer deep neural network, as its name indicates. As a result, VGG16 is a large network with 138 million parameters in total—it's huge even by today's standards.

Layer (type)
input_1 (InputLayer)
block1_conv1 (Conv2D)
block1_conv2 (Conv2D)
block1_pool (MaxPooling2D)
block2_conv1 (Conv2D)
block2_conv2 (Conv2D)
block2_pool (MaxPooling2D)
block3_conv1 (Conv2D)
block3_conv2 (Conv2D)
block3_conv3 (Conv2D)
block3_pool (MaxPooling2D)
block4_conv1 (Conv2D)
block4_conv2 (Conv2D)
block4_conv3 (Conv2D)
block4_pool (MaxPooling2D)
block5_conv1 (Conv2D)
block5_conv2 (Conv2D)
block5_conv3 (Conv2D)
block5_pool (MaxPooling2D)

Table 3.3: VGG16 Model Summary



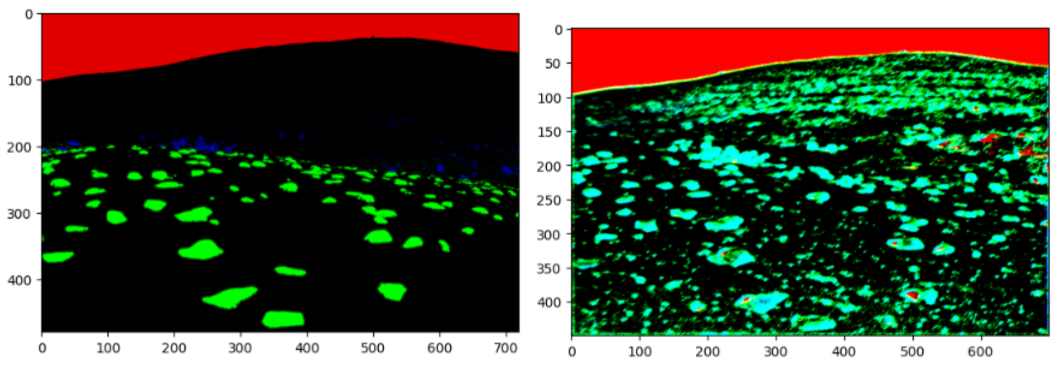


Figure 3.17: Output Image VGG16 & Unet

Clarification:
After conducting numerous experiments (using VGG16 & RESNET50 as a backbones) involving the alteration of loss functions, optimizers, and hyperparameters, we unfortunately did not achieve favorable results. This outcome can be attributed to the various limitations encountered throughout the process. However, it is important to note that the segmented images displayed promising characteristics, exhibiting a multitude of distinctive features. Despite the challenges faced, the quality of the segmented output provides encouragement for further exploration and refinement of the approach.

3.6.2.3 Autoencoders:

An autoencoder is an unsupervised neural network that learns to compress data by capturing important features in a lower-dimensional representation. Autoencoders consist of 3 parts:

Encoder: The encoder is the first part of an autoencoder. It takes the input data and compresses it into a smaller representation. This representation is typically much smaller than the original data, which makes it easier to store and process.

Bottleneck: The bottleneck is the middle part of an autoencoder. It contains the most important information from the input data. This information is compressed into a very small space, which makes it difficult to extract.

Decoder: The decoder is the last part of an autoencoder. It takes the compressed representation from the bottleneck and reconstructs the original data. This is done by expanding the compressed representation into a larger space.

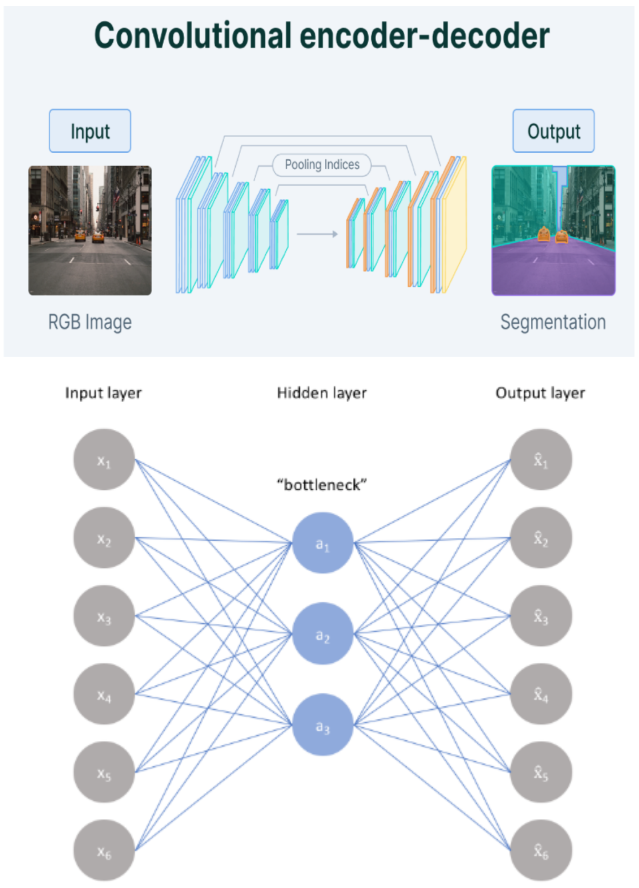


Figure 3.18: The architecture of autoencoders [29]

Output Image:

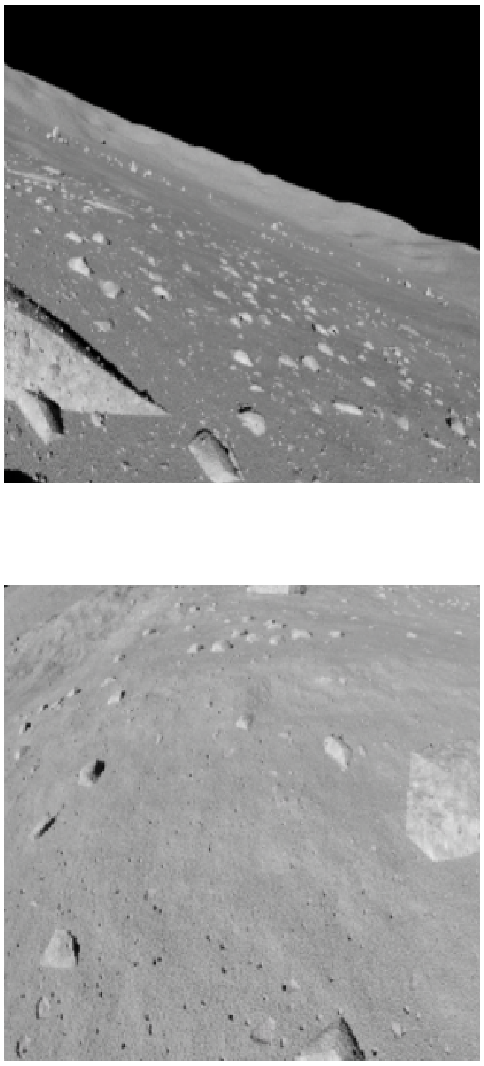


Figure 3.19: Image

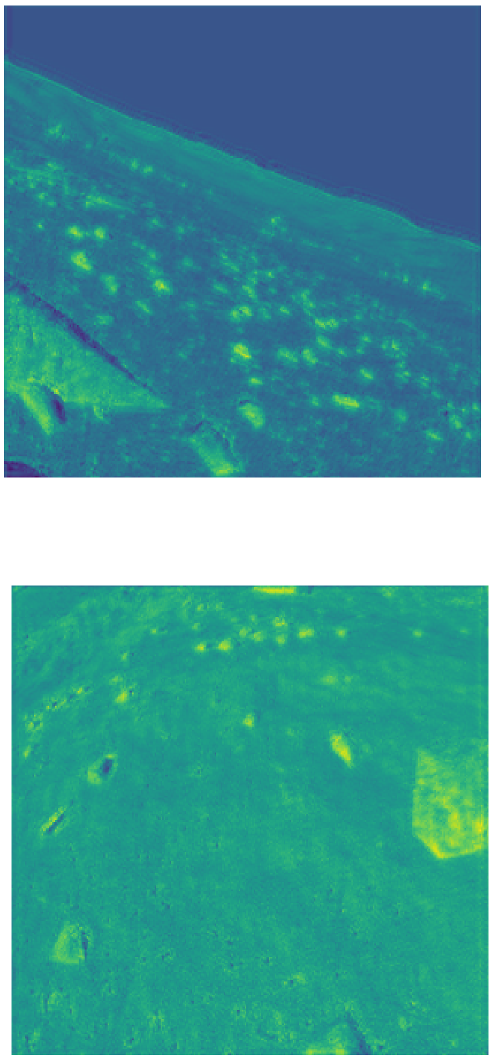


Figure 3.20: Predicted

Figure 3.21: Output Image Using Autoencoders

3.6.2.4 Segment Anything Model(SAM):

Since we are working on semantic segmentation, SAM must be tested before this section is concluded. SAM is a deep learning model (transformer based).it is a self-learning (does not need for additional training) segmentation system that can segment objects in images, even if it has never seen those objects before.

The `sam_model_registry` module provides a way to find and load SAM model checkpoints. The `SamAutomaticMaskGenerator` class can be used to generate masks for objects in images automatically. The `SamPredictor` class can be used to segment objects in images using a SAM model checkpoint.

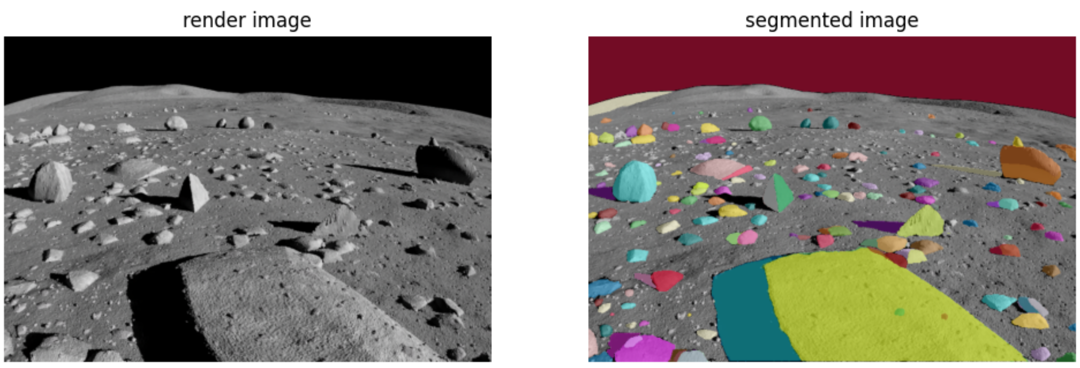


Figure 3.22: the generated mask using SAM

3.6.3 Object detection Models

When working on object detection , it was quickly realized that the existing dataset was not sufficient. The dataset only contained bounding boxes of clearly visible blue rocks with a minimum size of 20x20 pixels as mentioned below in 3.6.3.1. This meant that the model would not be able to detect rocks that were smaller, further away, or a different color. To address this issue, a lot of research was done and a number of tests were conducted3.6.3.3.All of these are detailed on the following pages.

3.6.3.1 Big rocks detection(Blue Rocks):

Using a simple code, we could get the output image 3.23. The code works by first loading a CSV file containing the bounding boxes of rocks. It then loads the segmented image and displays its bounding boxes.

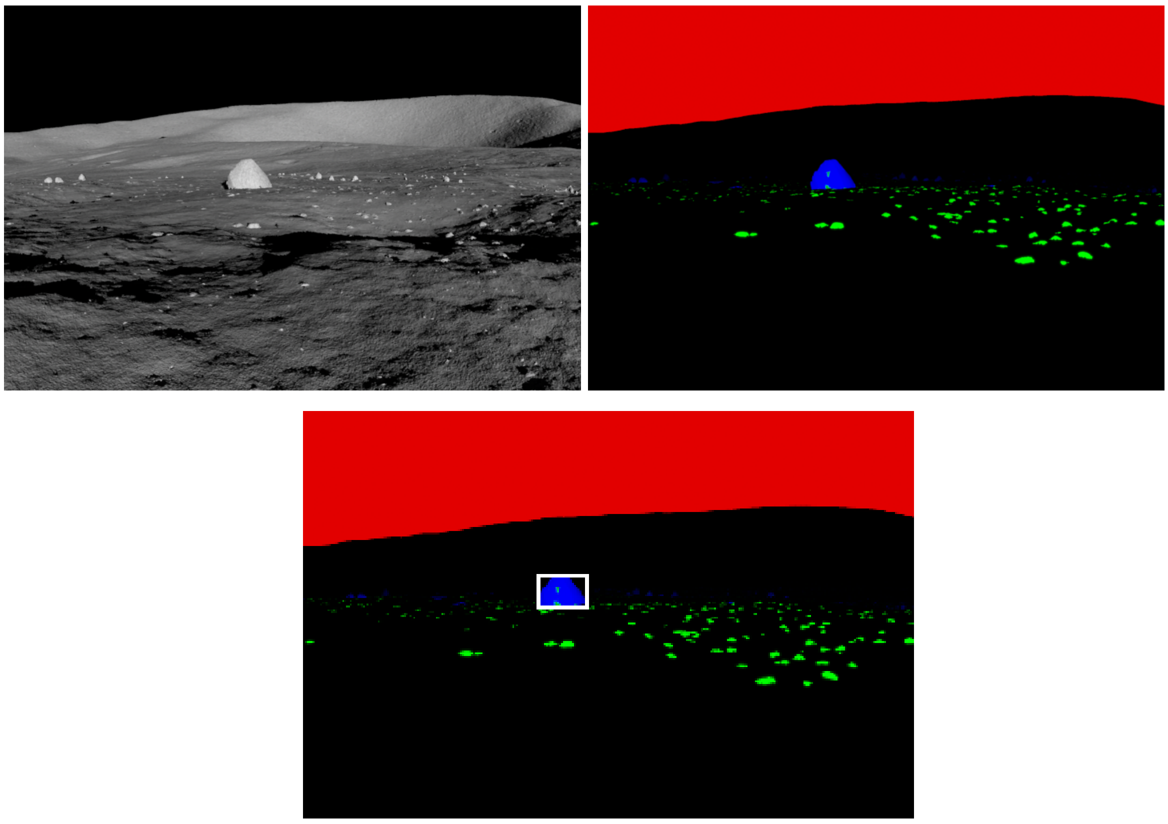


Figure 3.23: Blue Rocks Detection

3.6.3.2 Grounding DINO:

Grounding DINO is an open-set object detector that integrates language inputs with DINO and grounded pre-training. It aims to detect arbitrary objects specified by human language. The model leverages Transformers for seamless fusion of image and language data, simplifies the model design, and achieves strong closed-set object detection performance. With its ability to generalize to unseen objects, Grounding DINO demonstrates promising applications in various domains. It combines the power of language and vision to enhance open-set object detection and supports referring expression comprehension.[38]

Grounding DINO's architecture:

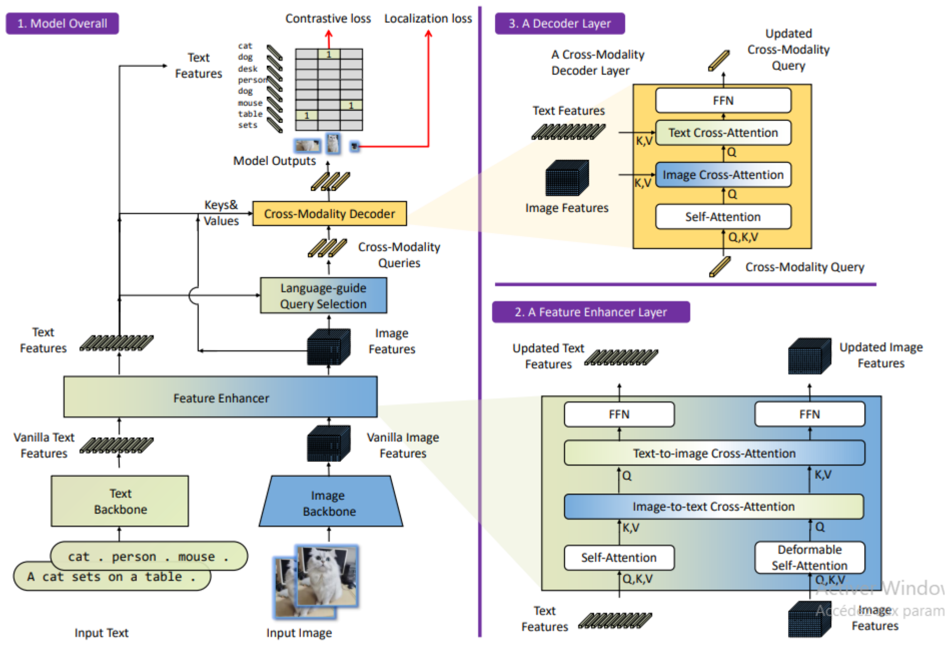


Figure 3.24: DINO's architecture [38]

Results:

We used the Grounding DINO for the object detection task giving it the following prompt: ['black sky','gray big ground','rocks']. here is the output image :

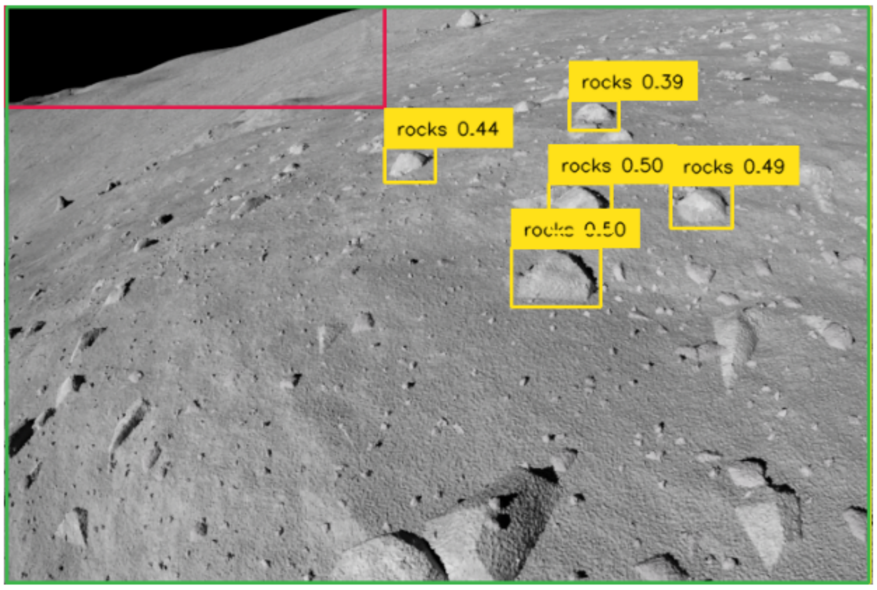


Figure 3.25: DINO's Result

The given results are not satisficing since it did not detect all the rocks, which can be a problem in the future.

3.6.3.3 YOLOv8:

YOLOv8 [55]is a fast, accurate, and versatile object detection algorithm that can be used for a wide range of applications. It is based on the CSPDarknet53 architecture and has been pre-trained on the COCO (Common Objects in Context) dataset[56]. YOLOv8 uses an anchor-free approach to object detection, which means that it does not need to pre-define a set of anchor boxes. This makes YOLOv8 more flexible and allows it to better handle objects of different sizes and shapes. YOLOv8 also predicts an objectness score for each bounding box, which indicates the probability that the box contains an object. This can be used to filter out false positives. YOLOv8 is trained on images at multiple scales, which helps it to better handle objects of different sizes.

Inference with Pre-trained COCO Model:
Even though the YOLOv8 model is pre-trained on the COCO dataset, which contains 80 different types of objects, it is important to test it on our dataset of lunar images to see how it performs at detecting rocks on the moon.

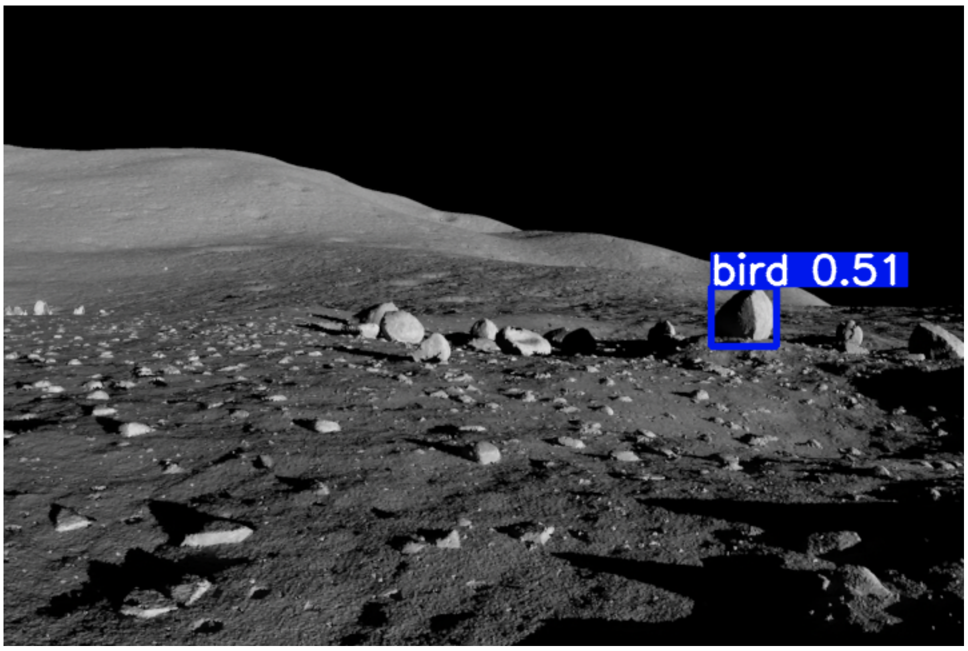


Figure 3.26: YOLOv8 Output Without Additional Training

Figure 3.26 shows that the YOLOv8 model is not able to detect rocks as accurately as it could if it were trained on a dataset of lunar images. This is because the COCO dataset, which the YOLOv8 model was trained on, does not contain any lunar images.

Preparing The Dataset :

USING ROBOFLOW : Roboflow is a computer vision platform to build accurate computer vision models. It provides tools and services for data collection, preprocessing, and model training, which can save users time and effort. Labeling tools of this platform were used in the process of preparing the dataset. Despite the challenges of annotating a dataset of lunar images, which are often complex and full of rocks, the annotation was completed manually by drawing bounding boxes around each rock and shadow.

the challenges of annotating lunar images

-Lunar images are very complex, with many different objects and features. This is what make it difficult to identify and annotate all of the objects in the image.
-Rocks and shadows can look very similar in lunar images, which can make it difficult to distinguish between them.
-Manual annotation is a tedious and time-consuming process

Due to the large number of images (9766), it was not possible to annotate all of them. Therefore, only 200 images were annotated.The dataset was then exported to the right format to train the model.

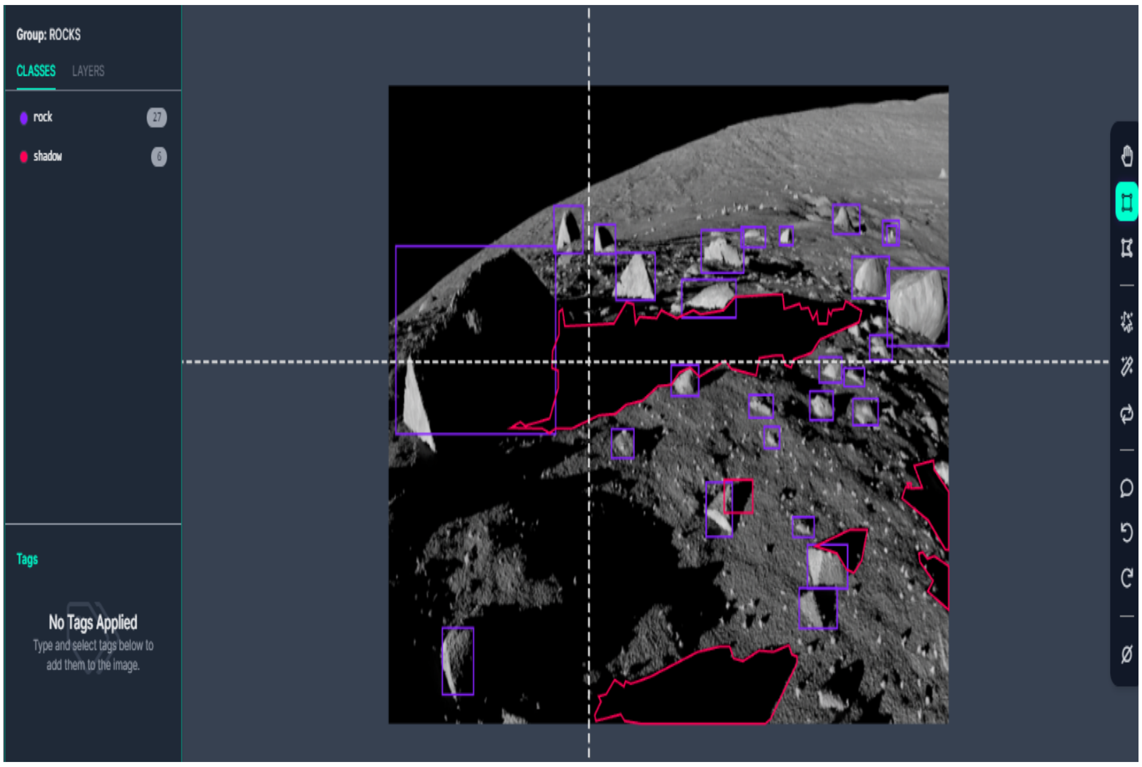


Figure 3.27: Annotating

Model/ Results :

The command `pip install ultralytics` enables the installation of YOLOv8.
After loading the dataset and trained the model, we obtained the following **results**:

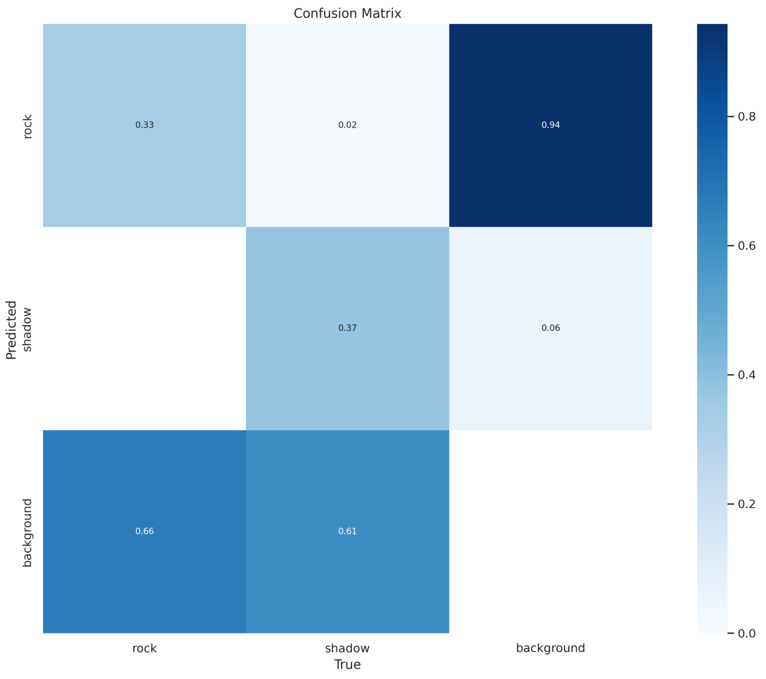


Figure 3.28: The confusion matrix returned after training

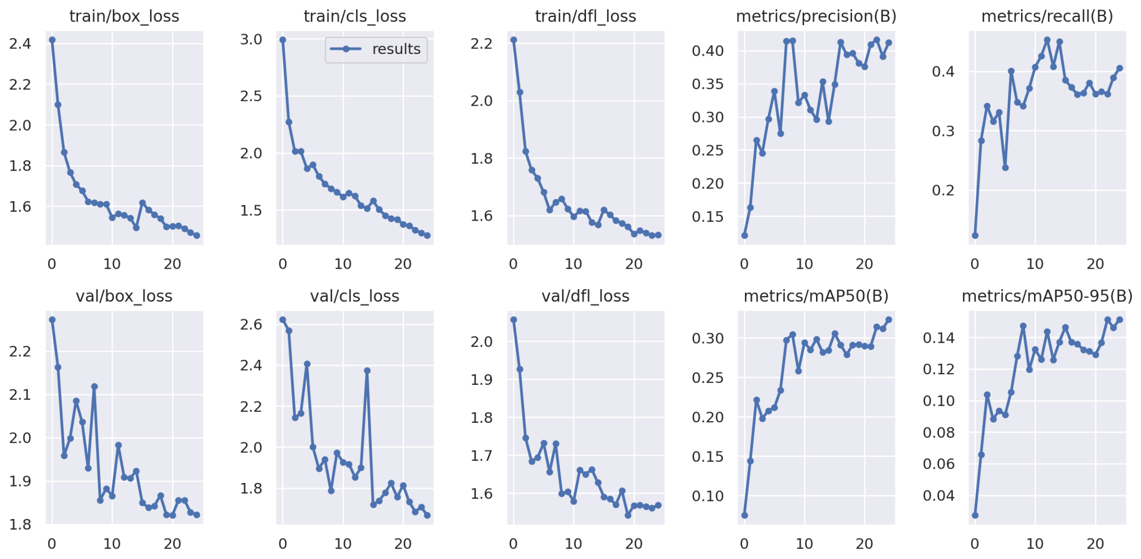
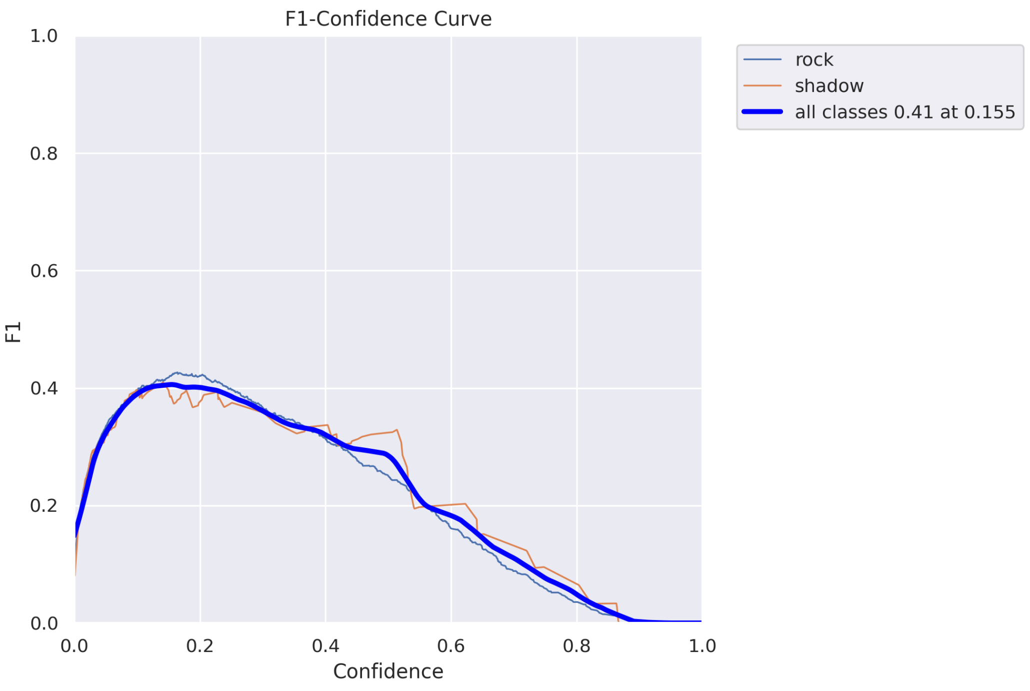


Figure 3.29: Results Tracked by YOLOv8



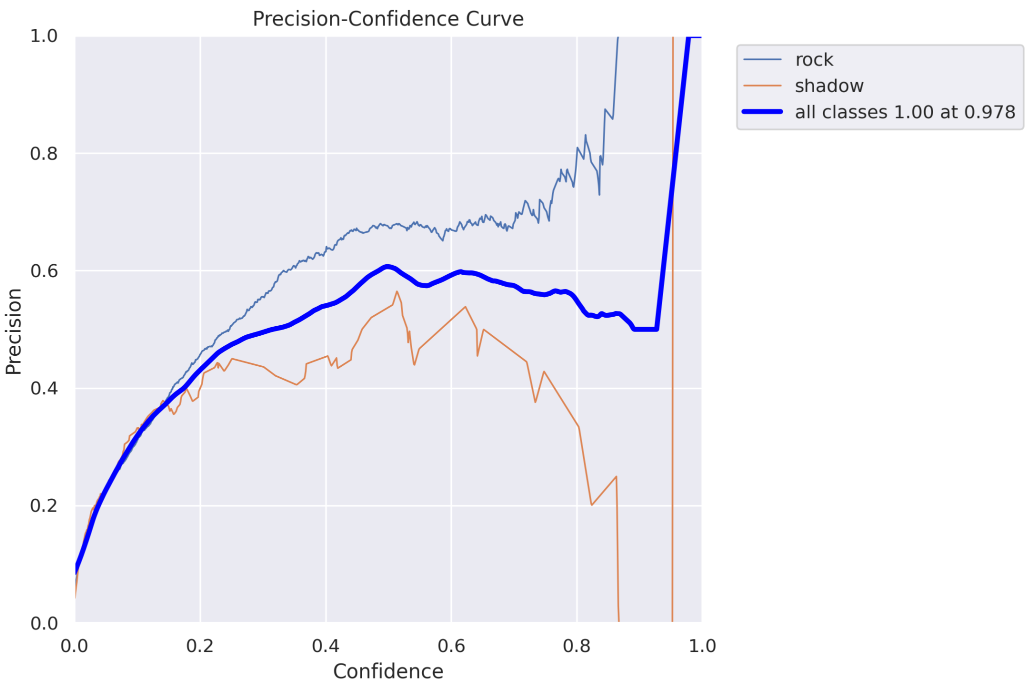


Figure 3.30: Curves

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	59	987	0.409	0.406	0.324	0.151
rock	59	931	0.426	0.42	0.364	0.171
shadow	59	56	0.392	0.393	0.284	0.132

Table 3.4: Validation Results

Ouput Images :

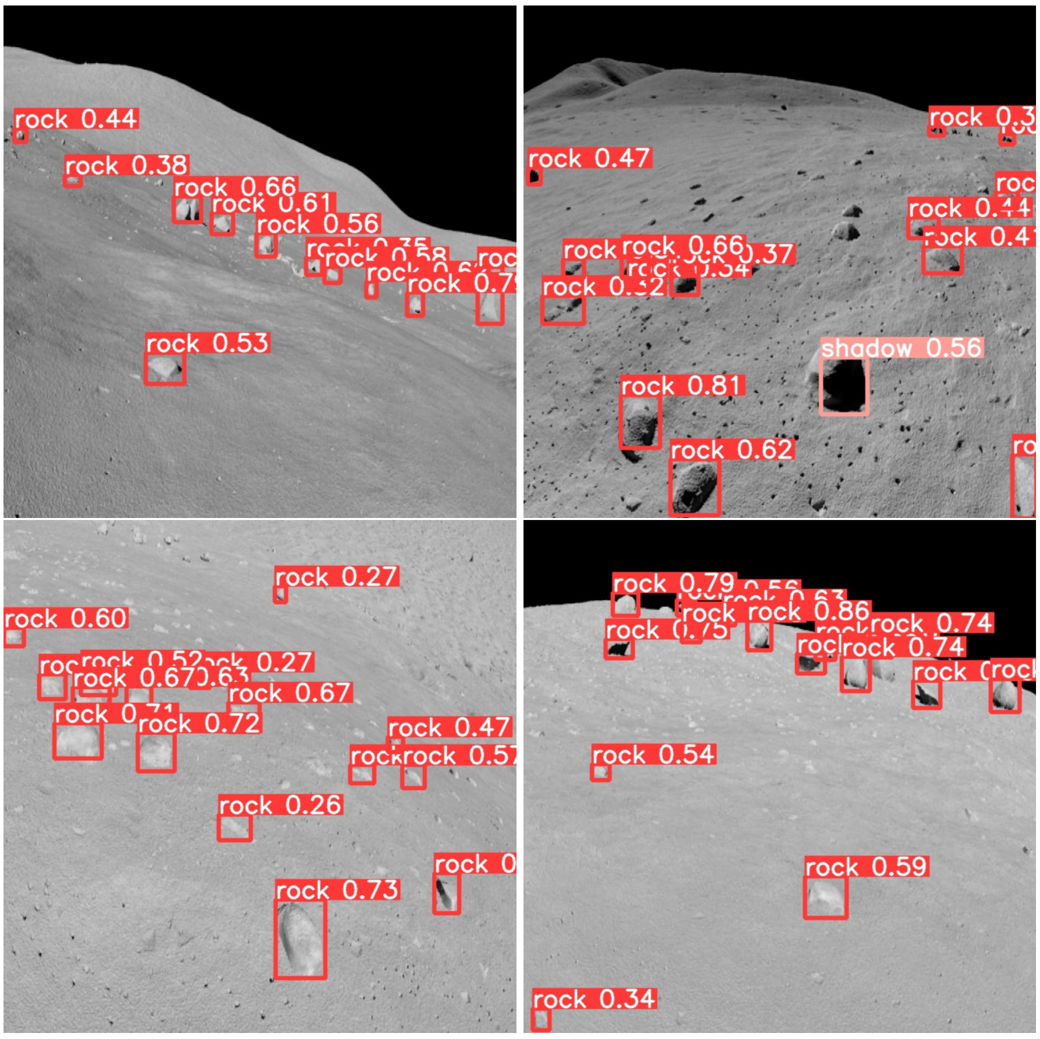


Figure 3.31: YOLOv8 Output After Additional Training

The results are promising even when using only 200 images. The YOLOv8 model is now able to detect rocks in lunar images with a high degree of accuracy.

3.6.3.4 Comparison Between YOLOv8 and Grounding DINO :

We explored two object detection models, namely YOLOV8 and Grounding DINO, for the purpose of detecting rocks in lunar images. YOLOV8 exhibited promising performance, although it required additional training to achieve optimal outcomes. With further refinement, YOLOV8 demonstrated the ability to accurately detect rocks in the lunar images with a reasonable level of accuracy. On the other hand, the implementation of Grounding DINO did not produce satisfactory results. Despite numerous efforts to optimize the model, it encountered difficulties in effectively identifying and localizing rocks within the lunar images. This divergence in performance underscores the varying effectiveness of different object detection approaches when applied to the specific task of rock detection in lunar imagery.

3.6.4 Grounding DINO & SAM for Image segmentation

We will be using both Grounding DINO & SAM for image segmentaion, this implementation requires two steps. The first step that emphasizes on using Grounding DINO for object detection was already done in the Grounding DINO implementation 3.25. In the second step, we will pass the result with its bounding box to the Segment Anything Model (SAM) for the image segmentation. Here is an example of the results of this task:

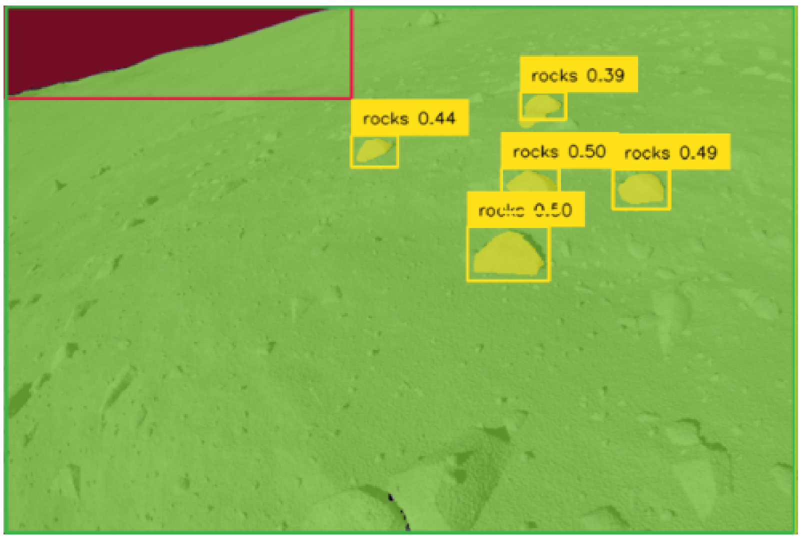


Figure 3.32: Output Image

3.7 Conclusion :

In this chapter we presented our dataset and a review of literature.we have detailed architectures and methods used in our implementation, moreover, we have shown and discuss the results for each method.

General Conclusion

In conclusion, this thesis has presented an in-depth exploration of various approaches in deep learning for object detection and image segmentation in the context of an artificial lunar landscape dataset. Extensive research and experimentations were conducted to investigate different architectures, loss functions, optimizers, and preprocessing techniques. Despite these efforts, the results of the image segmentation task did not meet the desired value of the evaluation metric. Several limitations were identified, including the complexity of lunar images and the constraints imposed by the available data.

However, it is important to note that significant progress was achieved in accurately segmenting and detecting objects within the lunar images, demonstrating the potential of deep learning techniques in lunar exploration. The developed models showcased promising capabilities and highlighted the importance of feature extraction and representation for accurate detection.

While the outcomes may not have been optimal, the conducted experiments provide valuable insights for future research in the field. The limitations encountered in this thesis, such as the constraints of the Colab environment, can be addressed by leveraging more advanced computational resources and exploring alternative techniques.

Finally, this thesis has contributed to the growing body of knowledge in applying deep learning for lunar landscape analysis. It serves as a foundation for future studies, inviting researchers to delve deeper into improving the results and robustness of image segmentation and object detection techniques for lunar imagery. The advancements made in this thesis pave the way for future applications of computer vision in lunar exploration and beyond.

Future plan

Moving forward, there is a strong commitment to further improve the results obtained in this thesis,Based on the knowledge acquired during the experiments.future endeavors will focus on refining the existing approaches and exploring new avenues for enhancement. One promising direction is the integration of reinforcement learning techniques, which can potentially enhance the performance and adaptability of the models.

BIBLIOGRAPHY

[1] *50 years ago, The Lunar Orbiter Program*.

[2] *How U-net works? | ArcGIS API for Python*.

[3] *Image Processing: How Do Image Classifiers Work?*

[4] *Object detection in an example image.*

[5] *Artificial Lunar Landscape Dataset*, 6 2019.

[6] M. ALLAN, U. WONG, P. M. FURLONG, A. ROGG, S. MCMICHAEL, T. WELSH, I. CHEN, S. PETERS, B. GERKEY, M. QUIGLEY, ET AL., *Planetary rover simulation for lunar exploration missions*, in 2019 IEEE Aerospace Conference, IEEE, 2019, pp. 1–19.

[7] AMANCHADHA, *coursera-deep-learning-specialization/Image*$_segmentation_Unet$.

[8] Y. AMIT, P. FELZENSZWALB, AND R. GIRSHICK*,* Object detection*, Computer Vision: A Reference Guide, (2020), pp. 1–9.*

[9] A. G. ANDERSON, C. P. BERG, D. P. MOSSING, AND B. A. OLSHAUSEN*,* Deepmovie: Using optical flow and deep neural networks to stylize movies*, May 2016.*

[10] A. A. AWAN*,* A complete guide to data augmentation*, Nov 2022.*

[11] V. BADRINARAYANAN, A. KENDALL, AND R. CIPOLLA*,* Segnet: A deep convolutional encoder-decoder architecture for image segmentation*, IEEE transactions on pattern analysis and machine intelligence, 39 (2017), pp. 2481–2495.*

[12] C. BAI, J. GUO, L. GUO, AND J. SONG*,* Deep multi-layer perception based terrain classification for planetary exploration rovers*, Sensors, 19 (2019), p. 3102.*

[13] K. CARROLL, H. SPENCER, J. ARKANI-HAMED, AND R. ZEE*,* Lunette: An affordable canadian lunar farside gravity mapping mission*, 09 2005.*

[14] E. CETINIC, T. LIPIC, AND S. GRGIC*,* Fine-tuning convolutional neural networks for fine art classification*, Expert Systems with Applications, 114 (2018), pp. 107–118.*

[15] D. DATAHACKER.RS, 017 cnn inception network, *Nov 2020.*

[16] H. DWIVEDI, How to use google colab for deep learning - complete tutorial, *Apr 2023.*

[17] EOPORTAL, Lunar Pathfinder - Minisatellite Mission.

[18] EUROPEAN SPACE AGENCY (ESA), Lunar Pathfinder, *September 2021.*

[19] FILTER FORGE, Normal Map Generator, *2014.*

[20] F. B. FITCH, Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133.: The journal of symbolic logic, *Mar 2014.*

[21] L. A. GATYS, A. S. ECKER, AND M. BETHGE, A neural algorithm of artistic style, *Sep 2015.*

[22] GEEKSFORGEEKS, ML ADAM Adaptive Moment Estimation Optimization, *GeeksforGeeks, (2021).*

[23] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, Deep learning, *MIT press, 2016.*

[24] Z. GUO, Q. CHEN, G. WU, Y. XU, R. SHIBASAKI, AND X. SHAO, Village building identification based on ensemble convolutional neural networks, *Sensors, 17 (2017), p. 2487.*

[25] A. GUPTA, A Comprehensive Guide on Optimizers in Deep Learning, *Analytics Vidhya, (2023).*

[26] S. H. HAJI AND A. M. ABDULAZEEZ, Comparison of optimization techniques based on gradient descent algorithm: A review, *PalArch's Journal of Archaeology of Egypt/Egyptology, 18 (2021), pp. 2715–2743.*

[27] S. O. ISMAEL, Deep learning and image processingfor handwritten style recognition: Deep learning and image processingfor handwritten style recognition, *2020.*

[28] S. JADON, Introduction to different activation functions for deep learning, *Feb 2022.*

[29] A. JAIN, C. VERMA, N. KUMAR, M. S. RABOACA, J. N. BALIYA, AND G. SUCIU, Image geo-site estimation using convolutional auto-encoder and multi-label support vector machine, *Information, 14 (2023).*

[30] K. JEEVITHA, A. IYSWARIYA, V. RAMKUMAR, S. M. BASHA, AND V. P. KUMAR, A review on various segmentation techniques in image processsing, *European Journal of Molecular & Clinical Medicine, 7 (2020), pp. 1342–1348.*

[31] P. JIANG, D. ERGU, F. LIU, Y. CAI, AND B. MA, A review of yolo algorithm developments, *Procedia Computer Science, 199 (2022), pp. 1066–1073.*

[32] F. J. JOHN JOSEPH, S. NONSIRI, AND A. MONSAKUL, Keras and TensorFlow: A Hands-On Experience, *07 2021, pp. 85–111.*

[33] N. JOSHI, Machine Learning libraries (NumPy, SciPy, matplotlib, scikit-learn, pandas), *© 2012-2022 Dotnetlovers, (2018).*

[34] V. KOOKNA, Semantic vs. Instance vs. Panoptic Segmentation - PyImageSearch, *6 2022.*

[35] A. KRENKER, J. BEŠTER, AND A. KOS, Introduction to the artificial neural networks, *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech, (2011), pp. 1–18.*

[36] B. J. KUIPERS AND T. S. LEVITT, Navigation and mapping in large scale space, *AI magazine, 9 (1988), pp. 25–25.*

[37] J. KURUVILLA, D. SUKUMARAN, A. SANKAR, AND S. P. JOY, A review on image processing and image segmentation, *in 2016 international conference on data mining and advanced computing (SAPIENCE), IEEE, 2016, pp. 198–203.*

[38] S. LIU, Z. ZENG, T. REN, F. LI, H. ZHANG, J. YANG, C. LI, J. YANG, H. SU, J. ZHU, ET AL., Grounding dino: Marrying dino with grounded pre-training for open-set object detection, *arXiv preprint arXiv:2303.05499, (2023).*

[39] LUNAR RECONNAISSANCE ORBITER CAMERA (LROC), 731: Big Island.

[40] MATELABS$_a i, Everythingyouneedtoknowaboutneuralnetworks$,2017.

[41] NASA, How to: Digital Terrain Models of the Moon.

[42] ——, NASA Multimedia Highlights: September 2010, *2010.*

[43] ——, Nasa project takes off with new 3d lunar simulation, *May 17 2023.*

[44] ——, SMART-1, *n.d.*

[45] NASA JET PROPULSION LABORATORY, Moon Image Collection.

[46] NASA LANGLEY RESEARCH CENTER, Apollo.
https://www.nasa.gov/centers/langley/news/factsheets/Apollo.html, Unknown.
Accessed: May 14, 2023.

[47] NASA Scientific Visualization Studio, SVS Gallery: Earth from Space, *n.d.*

[48] NASA Solar System Exploration, Clementine. *https://solarsystem.nasa.gov/missions/clementine/in-depth/, Unknown. Accessed: May 14, 2023.*

[49] R. Padilla, S. L. Netto, and E. A. Da Silva, A survey on performance metrics for object-detection algorithms, *in 2020 international conference on systems, signals and image processing (IWSSIP), IEEE, 2020, pp. 237–242.*

[50] P. Pandey, Understanding the Mathematics behind Gradient Descent., *(2021).*

[51] J. Patterson and A. Gibson, Deep learning: A practitioner's approach, *" O'Reilly Media, Inc.", 2017.*

[52] W. Python, Python, *Python Releases Wind, 24 (2021).*

[53] S. Raschka, What are gradient descent and stochastic gradient descent?, *4 2023.*

[54] A. Rastogi, Alexnet-the net that surpassed cnns, *May 2022.*

[55] D. Reis, J. Kupec, J. Hong, and A. Daoudi, Real-time flying object detection with yolov8, *arXiv preprint arXiv:2305.09972, (2023).*

[56] S. Rostianingsih, A. Setiawan, and C. I. Halim, Coco (creating common object in context) dataset for chemistry apparatus, *Procedia Computer Science, 171 (2020), pp. 2445–2452.*

[57] M. Ruder, A. Dosovitskiy, and T. Brox, Artistic style transfer for videos, *Oct 2016.*

[58] P. Seeböck, Deep learning in medical image analysis, *PhD thesis, Citeseer, 2015.*

[59] N. Seth, Is Gradient Descent sufficient for Neural Network?, *Analytics Vidhya, (2021).*

[60] ShaderMap, Creating an Albedo Map from a Photographic Texture, *n.d.*

[61] P. Sharma, Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code), *Analytics Vidhya, (2021).*

[62] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556, (2014).*

[63] ——, Very deep convolutional networks for large-scale image recognition, *Apr 2015.*

[64] S. SINGH, U. AHUJA, M. KUMAR, K. KUMAR, AND M. SACHDEVA, Face mask detection using yolov3 and faster r-cnn models: Covid-19 environment, *Multimedia Tools and Applications, 80 (2021), pp. 19753–19768.*

[65] P. SOCIETY, Earthrise from Lunar Orbiter 1, *4 2020.*

[66] SPACE.COM, Chang'e Program: China's Lunar Missions Explained, *n.d.*

[67] G. STOCKMAN AND L. G. SHAPIRO, Computer vision, *Prentice Hall PTR, 2001.*

[68] G. L. TEAM, OpenCV Tutorial: A Guide to Learn OpenCV in Python, *10 2022.*

[69] J. TERRA, Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning, *Simplilearn.com, (2023).*

[70] ——, Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning, *Simplilearn.com, (2023).*

[71] L. TORREY AND J. SHAVLIK, Transfer learning, *in Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, IGI global, 2010, pp. 242–264.*

[72] K. TREBING, T. STACZYK, AND S. MEHRKANOON, Smaat-unet: Precipitation nowcasting using a small attention-unet architecture, *Pattern Recognition Letters, 145 (2021), pp. 178–186.*

[73] P. VADAPALLI, Ultimate Guide to Object Detection Using Deep Learning [2023], *upGrad blog, (2022).*

[74] T. R. WATTERS, M. S. ROBINSON, R. A. BEYER, M. E. BANKS, J. F. BELL III, M. E. PRITCHARD, H. HIESINGER, C. H. VAN DER BOGERT, P. C. THOMAS, E. P. TURTLE, ET AL., Evidence of recent thrust faulting on the moon revealed by the lunar reconnaissance orbiter camera, *Science, 329 (2010), pp. 936–940.*

[75] M. YAGÜES GOMÀ, Image recognition with deep learning techniques and tensorflow, *Master's thesis, Universitat Politècnica de Catalunya, 2016.*