UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

FINAL MASTER THESIS

Master in Interdisciplinary and Innovative Engineering

SMOKE PLUME SEGMENTATION OF WILDFIRE IMAGES

Report and Annex

Author: Alba Baldrich Salvadó
Supervisor: Eulalia Planas Cuchi
Co-supervisor: Ronan Gabriel Michel Paugam
Department Chemical Engineering Department

Call: 2023, June

Abstract

This work is framed within the field of study of neural networks in Deep Learning. The aim of the project is to analyse and apply the neural networks that exist today in the market to solve a specific problem. This is about the segmentation of smoke plumes in forest fires.

A study of the neural networks used to solve image segmentation problems and also a subsequent 3D reconstruction of these smoke plumes has been developed. The algorithm finally chosen is the UNet model, a convolutional neural network based on the structure of autoencoders with step connections, which develops self-learning tasks to finally obtain a prediction of the class to be trained, in this case smoke plumes.

Also, a comparison between traditional algorithms and the UNet model applied using deep learning has been carried out, seeing that both quantitatively and qualitatively the best results are achieved by applying the UNet model, but at the same time it involves more computing time. All these models have been developed in the Python programming language using the Tensorflow and Keras machine learning books.

Within the UNet model, multiple experiments have been carried out to obtain the different hyperparameter values most suitable for the project application, obtaining an accuracy of 93.45% in the final model for smoke segmentation in wildfire images.

Resum

Aquest treball s'emmarca dins del camp d'estudi de les xarxes neuronals en Aprenentatge profund. L'objectiu del projecte és analitzar i aplicar les xarxes neuronals que hi ha avui dia en el mercat per resoldre un problema en específic. Aquest és tracta de la segmentació de plomalls de fum en incendis forestals.

S'ha desenvolupat un estudi de les xarxes neuronals utilitzades per resoldre problemes de segmentació d'imatges i també una reconstrucció posterior en 3D d'aquests plomalls de fum. L'algorisme finalment escollit és tracta del model UNet, una xarxa neuronal convolucional basada en l'estructura d'autoencoders amb connexions de pas, que desenvolupa tasques d'autoaprenentatge per finalment obtenir una predicció de la classe a segmentar entrenada, en aquest cas plomalls. de fum.

Posteriorment, una comparativa entre algoritmes tradicionals i el model UNet aplicat fent servir aprenentatge profund s'ha realitzat, veient que tant quantitativament com qualitativament s'aconsegueix els millors resultats aplicant el model UNet, però a la vegada comporta més temps de computació. Tots aquests models s'han desenvolupat amb el llenguatge de programació Python utilitzant els llibres d'aprenentatge automàtic Tensorflow i Keras.

Dins del model UNet s'han dut a terme múltiples experiments per obtenir els diferents valors dels hiperparàmetres més adequats per a l'aplicació del projecte, obtenint una precisió del 93.45 % en el model final per a la segmentació de fum en imatges d'incendis. forestals.

Resumen

Este trabajo se enmarca dentro del campo de estudio de las redes neuronales en aprendizaje profundo. El objetivo del proyecto es analizar y aplicar las redes neuronales que existen hoy en día en el mercado para resolver un problema en específico. Éste se trata de la segmentación de penachos de humo en incendios forestales.

Se ha desarrollado un estudio de las redes neuronales utilizadas para resolver problemas de segmentación de imágenes y también una reconstrucción posterior en 3D de estos penachos de humo. El algoritmo finalmente escogido se trata del modelo UNet, una red neuronal convolucional basada en la estructura de autoencoders con conexiones de paso, que desarrolla tareas de autoaprendizaje para finalmente obtener una predicción de la clase a segmentar entrenada, en este caso penachos de humo.

Posteriormente, una comparativa entre algoritmos tradicionales y el modelo UNet aplicado utilizando aprendizaje profundo se ha realizado, viendo que tanto cuantitativa como cualitativamente se consigue los mejores resultados aplicando el modelo UNet, pero a la vez conlleva más tiempo de computación. Todos estos modelos se han desarrollado con el lenguaje de programación Python utilizando libros de aprendizaje automático Tensorflow y Keras.

Dentro del modelo UNet se han llevado a cabo múltiples experimentos para obtener los distintos valores de los hiperparámetros más adecuados para la aplicación del proyecto, obteniendo una precisión del 93.45 % en el modelo final para la segmentación de humo en imágenes de incendios forestales.

Acknowledgements

I would like to take this opportunity to express my gratitude to the individuals who have played a significant role in the success of this project. This work would not have been possible without the support provided by my thesis director, Eulalia Planas, and my co-director, Ronan Paugman. Their guidance, expertise, and belief in my abilities have been crucial in this work and contributed to the overall quality of the project, a mention for the help received from Center for Technological Risk Studies (CERTEC) department team of Universitat Politècnica de Catalunya.

I would also like to extend my sincere appreciation to Josep Ramon Casas and Montse Pardas from the Signal Theory and Communications department of Barcelona School of Telecommunications Engineering (ETSETB) for their support, advice, and valuable insights, that have been helpful in shaping the technical aspects of this project. Their willingness to share their expertise and engage in the machine learning field have greatly enriched the depth and motivation of my research.

Additionally, I would like to express my thanks to my family and friends, especially Marta, Amaya and Bernat, for their unconditional support and encouragement throughout the duration of this work. Their understanding, patience and support have been a constant source of motivation.

An acknowledgment to be mentioned is also the support provided by the projects that have contributed to the successful completion of this work. The project "Closing gaps" with code RC20_C3_1025 has been helpful in providing resources for this research. Also express the support received from the Agencia Estatal de Investigación from Ministerio de ciencia e innovación, specifically through the projects PID2020-114766RB100 and TED2021-130484B-100.

Glossary

LR – Learning rate

TPR – True Positive Rate

TNR – True Negative Rate

FPR – False Positive Rate

FNR – False Negative Rate

CERTEC – Center for Technological Risk Studies (at Universitat Politècnica de Catalunya)

IR – Infrared camera (or thermographic camera)

MSE – Mean square error

GT – ground truth

CNN – Convolutional Neural Network

DL – Deep learning

DNN – Deep neural network

Mask R-CNN – Regional Convolutional Neural Network

Inference – the- trained deep neural network (DNN) make predictions (or inferences) on new (or novel) data that the model has never seen before.

NeRF – Neural radiance field

Instant NGP - Instant Neural Graphics Primitives

INDEX

INDEX OF FIGURES

INDEX OF TABLES

1. Preface

1.1. Background

This is a final master thesis of the "Interdisciplinary and innovative engineering" master's degree of the Barcelona East School of Engineering (EEBE) at the Diagonal-Besòs UPC campus. Among other subjects it includes a variety of basic applied machine learning techniques. One of the research groups with teaching responsibilities in the master was the Center for Technological Risk Studies (CERTEC). Although not directly involved in machine learning research, they offered the opportunity to solve one of the research questions posed in the frame of the international "Closing Gaps" research project by using some of the techniques learnt during the master's degree.

One open problem on fire studies is to modelise the behaviour and evolutions of fires. One important hint is the analysis of the smoke plume. This includes the volume and 3D shape of it. That information, extended to a video sequence should help on determining fire properties or/and evolutions. This was the starting point of this master thesis where we will address some of the preliminary works to get the final modelization done.

1.2. Motivation

The very first steps to a full modelization of firs are the segmentation of plume smoke on still images, before the 3D volumetric reconstruction. This can be tackled using different methodologies, but nowadays it is known that the state-of-the-art technology are those ones based on Deep Learning approaches. Although this will be one of our priorities, we won't skip to analyse the impact of traditional methods to the current problem. The Interdisciplinary and Innovative Engineering covers some of these topics although does not enter into computer vision details neither into the usage of machine learning with deep neural networks architectures. This is an opportunity to expand the knowledge in this area to round the master formation. Related to the specific work on the problem, the studied points will be traditional image processing techniques, and the more complex convolutional neural networks. Aware of different challenging techniques with higher potential suitable to image processing, I thought this would be a good opportunity to keep developing my knowledge in this field, more precisely on applying those techniques specially adapted to image segmentations as CNN autoencoders.

Overall, a project on autoencoders for smoke detection in a wild environment combines the challenges of environmental variability, limited labelled data, and the need for real-time detection. By leveraging the capabilities of autoencoders, you have the opportunity to make a meaningful contribution towards improving early smoke detection and fire management in these critical environments.

1.3. Requirements

At the very beginning, the project intended to recognize, extract and reconstruct the smoke characteristics in a wild scene. This included both 2D and 3D representation of the smoke plume. Due to the lack of data the project changed its aim to focus mainly to the 2D problem, but maintaining a deep study of what a 3D reconstruction procedure would imply in terms of complexity of both the data and the code. The main requirement of this project is to detect wildfire smoke plumes of the input given, that can be videos or images, segmenting these smoke plumes with the minimum time and maximum accuracy possible, being an ideal situation, the false positive number of predictions equal to zero.

2. Introduction

2.1. Objective

The main objective of this project is to design an image segmentation algorithm for the detection of wildfire smoke plumes. A smoke plume detection algorithm will be designed to detect the presence of smoke plumes, being able to accurately segment smoke plume in images and videos.

To accomplish the objective mentioned, four sub objectives and outcomes are developed:

– To develop the research and analysis of existing image segmentation techniques: The first objective involves research and analysis of current techniques for image segmentation, particularly as they relate to the CNN UNet model. This would involve reviewing relevant literature, studies, and databases, and evaluating the accuracy of existing approaches.

– To design a CNN UNet image segmentation model: Using the findings from the research and analysis in Step 1, the next step would be to develop a CNN UNet model that can accurately segment images and achieve an appropriate accuracy level. This would involve evaluating different approaches, refining the model architecture, and training and testing the model using appropriate datasets.

– To adapt the UNet model for video segmentation application: After developing the CNN UNet model for image segmentation, the next objective would be to adapt it for video segmentation. This would involve working with video data, evaluating the model's performance, and making adjustments to achieve the appropriate level of accuracy. Additionally, it would involve adapting the model's false positive rate (fpr) for the video dataset.

Lastly, the project will explore the possibility of 3D reconstruction of wildfires from the captured images. Overall, the goal of this study is to create a reliable and efficient solution be able to extract the volume of smoke plumes, to be able to validate physical fire simulators. This can help to the research field of smoke detection and reduce the damage caused by wildfires, and the posterior study of the smoke plumes characteristics, for risk analysis applications.

2.2. Scope

The optimal outcome of this project would be to get a real modelization of the smoke plumes, however this is one ambitious goal. Approaching to a preliminary product on laboratory conditions would be already a success. The full problem can be split in four different phases: 1) 2D smoke plume detection and segmentation, 2) multiview capture of a smoke test in laboratory, 3) 3D reconstruction of the smoke and 4) 2D segmentation and 3D reconstruction with temporal coherence.

The specific scope of this project is developing the first phase, where a study of the methodology, the posterior development and results comparison are defined. Also, the preliminary study of phase 3) is also performed, where a literature review, initial working conditions and some tests are performed.

As there is not enough available data on the laboratory data, the project will work with real case data (wildfire smoke). It is expected that the results obtained extrapolate to laboratory data, since the real problem is more complex than in controlled lab conditions. In laboratory conditions, the following factors can be controlled: air currents, lighting, absolute references, type of burnt material (which will define the type of smoke), etc. In the other hand, the application conditions in real cases cannot be controlled.

The dataset that will be used to work in this project consists of diverse smoke plume images captured during forest fires, other scenes such as chimneys or industrial areas will not be used. Forest fire images present challenges due to the dynamic and uncontrolled nature of the smoke plumes. The dataset includes images with various characteristics, including different types of smoke plumes as white, black or grey, varying illumination conditions, images taken in different viewpoints, and diverse image sizes. If we manage to get good results on non-controlled environment, the output in lab conditions should be at least as good as the ones on the trained conditions.

Some traditional segmentation techniques will be applied, however, the project is focused on the implementation of a deep learning image segmentation, where UNet is the one chose for the implementation. Some tests are also performed applying the UNet model created to wildfire video sequences.

Although there are other modern approaches like conditional diffusion model or even Vision Transformers. Both require extensive use of computational resources and a huge amount of data which it's not our initial conditions. Our decision is to keep on the UNet framework as a trade-off between performance capabilities and complexity and cost (both computational and data).

2.3. Tools

In this section the tools used for the development and implementation of this smoke plume segmentation project are presented. *Python* is the programming language used (*Python.Org*, n.d.), while *Keras* provided the framework for building and training the deep learning models (*Keras: Deep Learning for Humans*, n.d.). Moreover, *Wandb* is used to track, visualize and evaluate the performance of the different experiments done with the model (*Weights & Biases Documentation*, n.d.). The softwares used with the aim of performing annotation task are LabelMe (Torralba et al., 2010), for the labelling of the mask images in the dataset, and Labelbox (*Labelbox*, 2023) in order of performing data annotation error tests.

Python: Python is a high-level programming language known for being designed to be easy for humans to read and write, object-oriented features, and dynamic semantics. It offers built-in data structures and supports dynamic typing and dynamic binding, making it ideal for rapid application development and scripting purposes. Python's syntax is simple and easy to learn, it promotes modularity and code reuse through support for modules and packages. Python comes with an extensive standard library that are available for free for major platforms. (VanRossum & Drake, 2010)

Keras: Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow, acting as it's interface (Chollet & others, 2018). It was developed with a focus on enabling fast experimentation, so the main advantages of using Keras is that it's a simple, flexible, and powerful open-source library provided by Python.

Wandb: Weights&Biases (Wandb) platform is a python package that allows the monitorization of the training process of the model designed, in real time. It consists of a dashboard that after configurating with the parameters needed is used to keep track of the hyperparameters, system metrics, and predictions of the models tested, so it permits to easily compare models (Biewald, 2020).

LabelMe: LabelMe is an open-source graphical image annotation tool (Torralba et al., 2010), used to label the image dataset for this project. It provides an intuitive interface where users can manually label regions of interest in an image, such as the smoke plume in this project. LabelMe supports the creation of high-quality ground truth data for training and evaluating segmentation models.

Labelbox: Labelbox is a cloud-based platform that offers a comprehensive set of tools for data labelling (*Labelbox*, 2023), annotation management, and collaboration. LabelBox offers features such as image segmentation, bounding box labelling, and classification, making it suitable for various computer vision tasks. In this project Labelbox is used to make a comparative study of the error between different annotators when labelling a representative sample of the image dataset used in the smoke plume segmentation model.

3. Image segmentation

Image segmentation consists of identifying the different features and properties of an image by dividing the pixels of the image in groups and creating labels for each one of them. Once the segmentation step is completed, the images with its labels identified can be used for both, supervised and unsupervised training. The techniques used can be divided into traditional approaches, where different techniques have been studied for many years until deep learning techniques revolutionized the image segmentation process by achieving more advanced and accurate techniques.

Deep learning segmentation techniques employ Convolutional Neural Networks (CNNs), which are specifically designed for image analysis tasks. CNNs can learn hierarchical representations of the input data by applying convolutional filters to capture local patterns and features. This enables them to effectively segment images by assigning labels to different regions. For this reason, CNN architecture such as UNet is finally implemented in this project and with a posterior comparison of some traditional techniques, to obtain an overview of the performance, the implementation and computational complexity of different methodologies.

For each segmentation techniques there are different algorithms and methods, some of the most important methods for each segmentation technique, for both traditional and deep learning techniques, are mentioned below. The ones highlighted in bold are described in more detail in sections 3.1 and 3.2.

Traditional segmentation techniques:

- Threshold segmentation
  - **Local thresholding: Manual and Otsu's algorithms**
  - **Global thresholding – Adaptative algorithm**
- Edge-based segmentation
  - Gradient based operators
  - Gaussian based operators
- Region-Based segmentation
  - Region growing
  - Region splitting and merging
- Clustering segmentation algorithms
  - **K-means clustering**
  - Hierarchical
  - Gaussian Mixture Models (GMM) + BIC

Deep learning segmentation techniques:

- Neural Networks for segmentation
  - **U-Net**
  - Mask R-CNN
  - Fully Convolutional Network (FCN)
  - DeepLab

3.1. Traditional image segmentation techniques

In this section, we explore various traditional image segmentation techniques that have been widely used in the field of image processing. This section is focused on thresholding, region-based, edge-based and clustering image segmentation techniques, where inside this classification some methods used in each technique are explained.

3.1.1. Threshold segmentation

Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background (Benchamardimath & Hegadi, 2014; Sezgin & Sankur, 2004). This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. Thresholding segmentation can be divided into different methods, the global thresholding techniques use only one threshold value for the whole image, in the other hand, the local thresholding techniques, makes partitions of the images and uses a different threshold value for each partitioned part of the whole image.

Global thresholding

Global thresholding and local thresholding are image segmentation techniques commonly used in digital image processing. Global thresholding calculates a single threshold value that is applied to the entire image to create a binary image. All pixels whose intensity is greater than the threshold value are set to one (foreground), and the rest are set to zero (background). This method is fast and simple, but it may not be suitable for images with non-uniform lighting or with a large variation of object intensities. Global thresholding is simple and fast but may not work well in complex images. Two different approaches can be used to define the threshold:

- **Manual thresholding:** The human sets the threshold value, after analysing the histogram. This is an extremely subjective approach.
- **Otsu's algorithm:** One of the ways to achieve an optimal threshold is Otsu's method. In this method, we find the spread of foreground and background of the pictures for all possible values of threshold. The threshold with the least spread is taken as the optimal threshold. Otsu's is defined as a weighted sum of variances of the two classes that minimizes the intra-

class variance; it operates directly on the grey level image. This method has a simple implementation, but it fails when the global distribution occurs.

Local thresholding

Local thresholding overcomes the limitations of global thresholding by adapting the threshold value to local image characteristics (Gonzalez et al., 2009). Local thresholding methods calculate the threshold value for each pixel or region of pixels based on the information of the surrounding pixels, such as the average intensity or the local standard deviation. This approach is more accurate in dealing with images with non-uniform lighting or with a larger variation of object intensities, but it can be computationally expensive and may require additional parameters to optimize the thresholding. Local thresholding is more accurate but computationally expensive. The Adaptative thresholding is the most used method inside local thresholding techniques.

- **Adaptive thresholding:** A single threshold value may not be sufficient because it may work well in a certain part of the image but may fail in another part. To resolve this limitation, adaptive thresholding can be used. This technique considers each pixel and its neighbourhood, and calculates the threshold of it, by applying the arithmetic mean or gaussian mean. In Gaussian mean, pixel value farther from the centre of the region contributes less in finding the threshold of the region, while in arithmetic mean, all pixel values contribute equally.

3.1.2. Region-based segmentation

Region-based segmentation is another widely used technique for image and video segmentation, where the image is partitioned into regions or segments that share similar properties such as intensity, texture, and colour. There are several region-based segmentation methods, but the two most common methods are region growing and region splitting and merging (Arbelaez et al., 2010).

Region growing

Region growing is a bottom-up approach that starts with a small set of initial pixels, called seed pixels, and gradually expands the region by merging neighbouring pixels that have similar properties. The algorithm repeatedly checks the similarity criteria between adjacent pixels and merges them until the criteria are not met. The advantage of region growing is its ability to capture local structures and details, making it suitable for segmentation in images and videos with regions containing smooth intensity transitions or texture variations.

However, region growing is sensitive to the choice of seed pixels and similarity criteria, which can lead to over-segmentation or under-segmentation. Over-segmentation occurs when the algorithm merges irrelevant pixels and divides the region into several smaller regions. Under-segmentation occurs when the algorithm fails to merge similar pixels, leading to a single region with different properties.

Region splitting and merging

Region splitting and merging is a top-down approach that begins by dividing the image into several sub-regions using clustering or threshold methods. The algorithm then successively applies splitting and merging operations to refine the regions until they satisfy specific stopping criteria. The splitting operation divides a region into smaller sub-regions, while the merging operation combines neighbouring regions that have similar properties.

The advantages of region splitting and merging is Its ability to reduce over-segmentation by grouping similar regions and partitioning different regions, thus producing more coherent and meaningful segments in the image or video. However, region splitting and merging can also be computationally expensive and time-consuming, especially for large images or videos.

Overall, region-based segmentation methods offer a powerful tool for partitioning an image into meaningful segments that can be analysed and processed independently. The choice of a region-based segmentation method depends on the image characteristics, the available data, and the desired level of segmentation. Both region growing and region splitting and merging have their advantages and disadvantages, and their selection depends on the specific application requirements.

3.1.3. Edge-based segmentation

Edge detection is an image segmentation method consisting of detecting regions of discontinuity, by separating the regions when a significant change in the greyscale level is observed (finding boundaries of objects in the image) (Zheng et al., 2010). There are different ways to differentiate the types of edge-based segmentation methods, depending on the type of operator applied.

The advantages of gradient-based operators include their relatively simple implementation and relatively quick execution time. However, their results can be sensitive to image noise and may produce false edges. In the other hand, the advantages of Gaussian-based operators include their ability to better handle image noise and their ability to detect edges of different scales and orientations, but their implementation and execution time can be more complex.

Gradient based operator

Gradient-based operators are one class of edge-based segmentation methods that calculate the gradient approximation of an image by convolving it with a specific kernel. Some of the most commonly used gradient-based operators include the Sobel operator, the Prewitt operator, and the Robert operator.

- **Sobel operator:** It computes the gradient approximation of image intensity function for image edge detection using two 3x3 kernels, one for the horizontal direction and one for the vertical direction, to detect edges in both directions.

- **Prewitt operator:** It is similar to Sobel operator, but it detects additionally vertical and horizontal edges of an image using two separate kernels.
- **Robert operator:** This operator is a simpler alternative to Sobel and Prewitt operators, using a simpler kernel of 2x2 for edge detection. It is the less used operator due to its simplicity.

Gaussian based operator

Gaussian-based operators are another class of edge-based segmentation methods that apply a Gaussian blur to smooth the image before detecting edges. The most used Gaussian-based operators are the Canny edge detector and the Laplacian of Gaussian operator.

- **Canny edge detector:** This operator is a multi-stage algorithm used to detect edges by applying a series of filters to the image. It is widely used due to its ability to detect edges accurately while also reducing noise.
- **Laplacian of Gaussian**: This operator combines the Laplacian filter with a Gaussian kernel to highlight edges in an image. It is useful for detecting edges of different scales and orientations.

3.1.4. Clustering segmentation

Clustering is a technique that groups similar pixels in an image into clusters based on their similarity in terms of colour, texture, or other features. Clustering can be used as a segmentation technique in smoke image and video processing. In this section, we discuss three clustering methods: K-means clustering, hierarchical clustering, and Gaussian Mixture Models (GMM) with Bayesian Information Criterion (BIC).

K-means clustering

K-means is an iterative clustering algorithm that partitions data into K clusters, where K is a user-defined parameter. In the image segmentation context, K-means clustering assigns each pixel in the image to the cluster whose centre is closest to that pixel's colour value. K-means clustering is computationally efficient and easy to implement, but it may not work well for images with overlapping clusters. This clustering method is known for being efficient and easy to implement. Its main advantages are easy detection and implementation, while the main disadvantage is the needs to define the value of cluster.

Hierarchical

Hierarchical clustering is a bottom-up clustering method that creates a hierarchy of clusters. Initially, each pixel is considered as a cluster, and the algorithm iteratively merges the two closest clusters until all pixels are in a single cluster. Hierarchical clustering can produce a more accurate segmentation result than K-means clustering, but it is computationally expensive and sensitive to initialization. Hierarchical clustering method is not as easy to implement as K-means clustering, previously described, but it can produce more accurate results for complex images (Arbelaez et al., 2010).

Gaussian Mixture Models (GMM) + BIC

GMM is a probabilistic clustering method that models the distribution of pixel intensities as a weighted sum of Gaussian distributions. BIC is used to determine the optimal number of Gaussian components in the mixture model. GMM + BIC can produce high-quality segmentation results, but it is computationally expensive and sensitive to noise in the data, it is recommended to use for complex image dataset applications (Huang & Wang, 1995).

3.2. Deep Learning image segmentation techniques

Deep learning refers to a subfield of machine learning that consists of training artificial neural networks with multiple layers, known as deep neural networks (DNN), its aim is to learn and extract patterns from large amounts of data. Inside deep learning there are different methods, the most common one is Multi-Layer Perceptron (MLP), which is not suitable to deal with images. Ie Convolutional Neural Networks (CNN) were created to solve this issue. MLPs consists of several sequential layers connected with neurons, each neuron takes the inputs from the previous layer and applies a set of weights to these inputs before passing the output to the next layer. CNNs use a special type of neural network layer called the convolutional layer, which applies a set of learnable filters to the input image to abstract the input image as a feature map. The convolutional approach allows to create deepest networks (amount of staked layers) keeping the number of parameters to train on a reasonable range. The output of the convolutional layer is then passed through other layers such as pooling layer, that summarize the presence of features in patches of the feature map, and fully connected layers that connect every neuron in one layer to every neuron in another layer to produce the final output. Neural network segmentation methods are used to train data to solve complex problems and easily detect errors. Its main disadvantage is that the training process consumes more time than the traditional segmentation techniques previously exposed, and also the amount of data needed is much more larger.

The main difference between MLP and CNN is that MLPs are designed for processing vector data and CNNs are designed for processing grid-like topology or with spatial relationships data, like images and videos (Kumar, 2021). For this reason CNNs have become the most commonly used deep learning architecture for image segmentation tasks (Alzubaidi et al., 2021).

The initial CNN architectures were created for image classification. That means that the output reduces to a scalar indicating at which class that image belongs. To get an image segmented we need the output to be the same size as the input, i.e. the output is an image where every point indicates at which class belongs this part of the image. An autoencoder is a type of neural network architecture with an encoder-decoder structure, see Figure 1. The aim of autoencoders is to learn to encode, consisting of compressing the input data to a lower dimensional representation to extract relevant features, and then reconstruct it (decoder), providing an output that matches the input size.

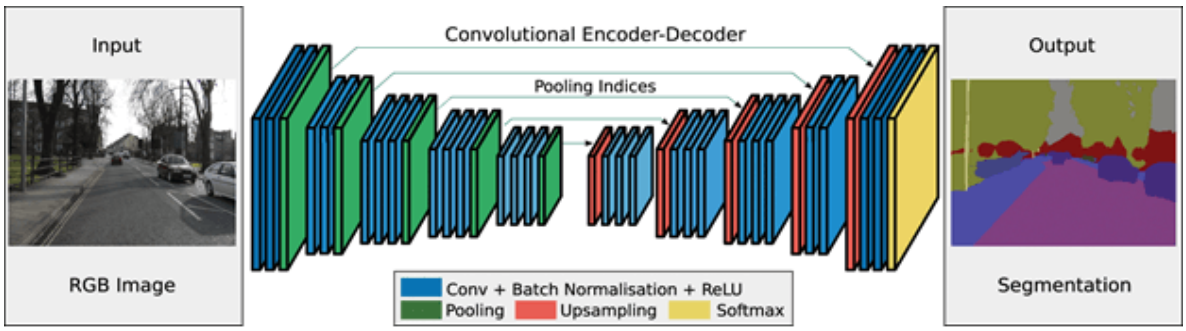


Figure 1. Autoencoder basic structure architecture (Mwiti, 2022)

There are several CNN structures that can be used for image segmentation, by integrating CNNs with autoencoders, UNet is one of the structures that combines the strengths of both approaches, making it effective for image segmentation tasks. The autoencoder is used to learn and encode the input data, while CNN is used for extracting the features and generating the output in resized maps.

3.2.1. UNet

UNet uses the concept of a Fully Convolution Network, but with some variations. It consists on a encoder-decoder cascade structure, where the encoder compresses gradually the information of the image and the decoder, decodes the information to the original dimension of the image, all this process gives a U-shape on the model, giving its name U-Net. The main problem of traditional autoencoders applied to images is that they lack on recovering details. This is the reason to create what is called “skip connections” (grey arrows in Figure 2), which are used to make better predictions by enabling the information flow between encoder and decoder of the model. Skip connections, in a given depth, just concatenate the output in de encoding with the output in the decoder of the previous layer, which explains how the architecture recovers details from the original image but on the new representation of the problem.

Originally UNet model was designed in (Ronneberger et al., 2015), with the purpose to use it in medical images, but nowadays is used for all types of images, such as aerial ones or smoke plumbs of wildfires, as in this case. The main advantages of the U-Net model are:

- Works with a few training samples.
- Global location and context at the same time.
- Not only limited in medical imaging, used also for computer vision.
- Simple structure (repetition of basic building blocks).
- Image size agnostic (because does not have fully connected layers).
- Relatively easy to understand.

And its main disadvantages:

- Takes a lot of time to train (due to the many layers)
- Takes a lot of time on CPU computation.
- High GPU memory footprint.

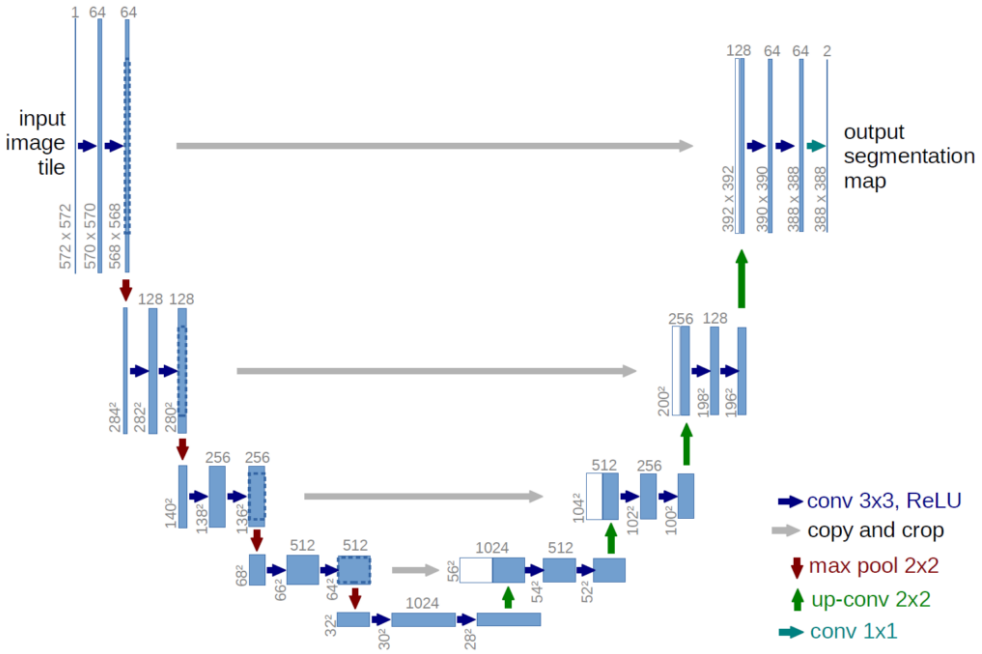


Figure 2. UNet model architecture

3.2.2. Basic concepts of training process

One of the main points to take into account in deep learning is how to make the network to accomplish a given task. Usually this is done my minimizing a function related to the task at hand. In this case we are doing segmentation, which is a specific case of classification (we classify each pixel), and the standard function to minimize is the binary cross entropy loss. However, another loss (function) that can be minimized to achieve the same task is the inverse of the dice coefficient, also known as dice loss. We define as loss any function used to optimize a machine learning algorithm.

Losses

Binary cross entropy measures the differences of information content between the actual and predicted masks of the image (Dr.A. Usha Ruby, 2020).

$$\boldsymbol{Binary\ cross\ entropy} \rightarrow \boldsymbol{L(y)} = -\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{y}\,\mathbf{log}(\boldsymbol{p_i}) + (\mathbf{1}-\boldsymbol{y_i})\,\mathbf{log}(\mathbf{1}-\boldsymbol{p_i})) \qquad \text{(Eq. 3.2.1)}$$

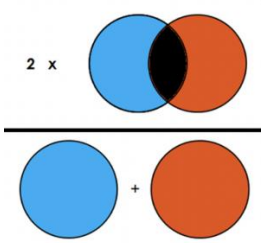
Being p_i the output probability of having class 1 and $(1 - p_i)$ is the probability of obtaining class 0, which is inside a range value between 0 and 1.

The *Dice* is a commonly used loss function that calculates the similarity between images, the dice coefficient (Eq. 3.2.4) is adapted to obtain the dice loss.

$$\boldsymbol{Dice\ loss} = \mathbf{1} - \boldsymbol{DSC} \qquad \text{(Eq. 3.2.2)}$$

Were DSC is known as *Sørensen–Dice*, which consists on an statistical equation that measures the similarity between two datasets, in image segmentation algorithms it is used to compare the ground truth mask data with the predicted masks of the images dataset, a value between 0 and 1 is obtained and gives us information about the learning of the model (Zhao et al., 2020).

$$\boldsymbol{DSC = 2 * \frac{mask_{GT} \cap mask_{pred}}{mask_{GT} + mask_{pred}}} \quad \text{(Eq. 3.2.3)}$$



Epochs

In deep learning, an epoch refers to one complete pass of the entire training dataset through a neural network. It represents the number of times the entire dataset is used to train the model.

Learning rate

The learning rate is a hyperparameter that determines the step size at which the model updates its weights during the training process. It controls how quickly or slowly the model learns from the training data.

Batch size

Batch size refers to the number of training examples (data samples) that are processed together in one forward and backward pass during each iteration of the training process. It affects the speed and efficiency of training as well as the memory requirements.

Dropout

Dropout is a regularization technique in deep learning that helps prevent overfitting. It randomly drops out a proportion of the neurons during training, forcing the network to learn more robust representations by preventing overreliance on specific neurons.

Weights

In deep learning, weights refer to the learnable parameters of a neural network that are adjusted during the training process to minimize the difference between predicted outputs and ground truth labels. The weights determine the strength of connections between neurons and are crucial for the network to learn and make accurate predictions.

Transfer learning

Transfer learning is a technique in deep learning where a pre-trained model, which has been trained on a large dataset for a related task, is used as a starting point for a new task. The knowledge and learned representations from the pre-trained model are transferred to the new model, usually by fine-tuning or feature extraction.

Backbone

In deep learning, the backbone refers to the main architecture or network that forms the core structure of a model. It typically consists of multiple layers and is responsible for extracting meaningful features from the input data. The backbone serves as the foundation upon which additional layers or modules are built for specific tasks.

4. Image Dataset

The wildfire smoke plume image dataset was collected from various sources, including satellite imagery, aerial photography, and ground-based image capture during real wildfire incidents. Special attention was given to capturing diverse scenarios, including different smoke plume sizes, shapes, and environmental conditions. The dataset was provided by the Centre for Technological Risk Studies (CERTEC), a research group of the Universitat Politècnica de Catalunya (UPC) (*CERTEC*, 2023). A total of 190 images of wildfires are used, with a different shape each image, for this reason in the pre-processing step of the segmentation project, a reshape of all the images must be done to obtain the same shape. See Figure 3 and Figure 4 as an example of five images of the train and test dataset respectively.

Figure 3. Example of 5 images of train dataset

Figure 4. Example of 5 images of test dataset

4.1. Ground truth labelling

Ground truth labelling is the process of manually labelling data by humans, called annotators, to create an accurate, reliable, and robust dataset for being used in a machine learning training model. It's a time-consuming and indeed expensive process, but also crucial to ensure achieving a high accuracy performance of the model.

In order to extract the features of each raw image LabelMe has been used the manual annotation of each smoke plume of the image dataset was done, a total of 190 images. LabelMe gives the annotation labels in *.json* format, to import it in the U-Net algorithm, these file formats are converted from *.json* to a masks file in *.png* to read and load it easily in python.

JSON files (Maeda, 2012) can also be converted to Pascal VOC, COCO or TFRecord format. The mask with extension file *.png* is the one recommended to use when applying a UNet model. To obtain this *.png* mask images (black the background and red the smoke plume) the Pascal VOC conversion is done. The obtained data after performing the conversion from LabelMe annotations are: the *.jpg* original raw image, the *.png* mask image, a file with the segmentation class of each image and an image with the class visualization of each image in black and white. An example of the mask image obtained is shown in Figure 5.

Figure 5. Image labelling example. Left: mask image. Right: class visualization of the annotation.

4.2. Image annotation error

Annotation is the labelling of data by human effort (Tseng et33la, 2020). For a good performance of the algorithm training and the model, it is necessary to have high quality data, such as an extensive dataset of good quality images properly labelled. Low-quality human annotations result in wrong labels for retraining automated classifiers and indirectly contribute to the creation of inaccurate classifiers (Pandey et al., 2022).

There are different causes of image annotation errors:

- **Annotator burnout.** To prevent this, a sample of 40 images of the database of this project is chosen to perform this study, another mitigation strategy for burnout is to limit the working time of each annotator or limit the unit of labels in a certain working time.
- **Human error:** The human error on the annotating process can be divided into two types.
    - Mistakes: errors due to incorrect or incomplete knowledge (Metcalfe, 2017)
    - Slips: errors in the presence of correct and complete knowledge (Metcalfe, 2017), when the annotator knows the correct category but selects an incorrect one, when this error is recurrent, it's normally due to burnout (Pandey et al., 2019).

One of the main challenges for data annotation is to establish high-level end goals, which for this project the following ones have been defined: labelling the images in order to create an image segmentation algorithm to identify smoke in wildfires and compare the annotations between a team of different annotators to analyse the annotation human error that it can be produced while implementing the process.

There are different measures allowing to quantify the quality of the annotations for the image dataset, the goal is to measure how correct a label is and how often it is correct.

- **Honeypot:** this measures the disagreement between the annotator's labels and the ground-truth labels (*How to Make Annotation Painless*, 2023).
- **Consensus:** is a disagreement metric that measures the similarity between the same labels from multiple annotators.
- **Review:** used to identify low-accuracy and inconsistencies in the labelling process, carried out by trusted labelling experts.

Several software's and applications for labelling images have been considered, finally the Labelbox application (*Labelbox*, 2023) has been the chosen tool, for being intuitive, easy to access and use, it has free access to the services without the need of downloading anything, the team will work on an online environment. An annotation guideline has been created, so the annotation team can follow the instructions of how to use the Labelbox tool and the annotation criteria to apply in this study.

The consensus of 7 different annotators (members of the CERTEC group) for the same images has been calculated for 40 original images, so in total 280 annotation labels will be obtained, and compared the disagreement between 7 masks, made by 7 different workers, from the same image. The results in Figure 6 shows a histogram of the quality scores for the labels created, with an average quality score of 77%.

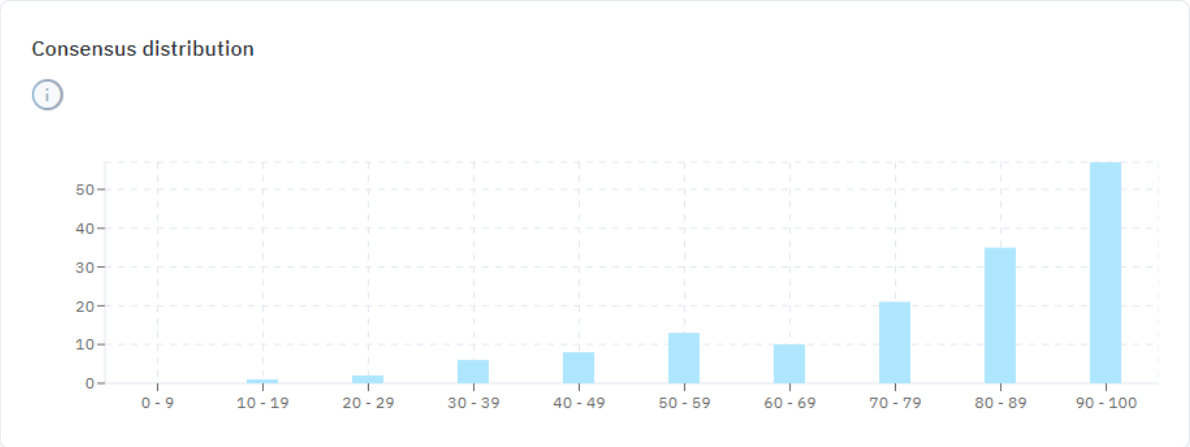


Figure 6. Consensus distribution for image annotation bias dataset

In conclusion, the nature of smoke plumes has several challenges regarding the annotation process, and it is expected that some errors may be obtained on the model's training due to the variability of smoke and the difficulty in determining precise boundary limits of the smoke plume.

While an annotation guide has been provided to ensure that all annotators follow the same criteria, these inherent characteristics of smoke plumes contribute to the potential for annotation errors. It is important to acknowledge the limitations in achieving perfect annotation accuracy for smoke plume segmentation. Facing challenges in determining the exact boundary between smoke plumes and their surroundings, especially in cases where the plumes are diffuse, irregularly shaped, or overlapping with other objects or background elements.

5. Evaluation techniques

In the field of wildfire smoke plume segmentation, evaluating the performance of segmentation models is crucial to assess their accuracy and effectiveness. To accomplish this, various evaluation techniques and metrics are explained in this section for a posterior implementation in the different models tested. These metrics provide information of the model's ability to correctly identify smoke plume regions and distinguish them from the background.

One metric to evaluate segmentation methods is a confusion matrix, which provides information about the model's performance. It is commonly used in binary classification or segmentation problems where there are two classes, as in this work in which we have a positive (smoke) and a negative (non-smoke) class. The results are summarized in a 2x2 matrix, the confusion matrix (see Table 1) summarizes the results of the model's predictions on a set of test data, comparing the predicted labels with the true labels.

The values shown in the confusion matrix are the following:

- True positive (TP): A smoke pixel that has been correctly segmented as pattern.
- True negative (TN): A background pixel that has been correctly segmented as background.
- False positive (FP): A background pixel that has been mistakenly segmented as pattern (false alarm, Type I error).
- False negative (FN): A smoke pixel that has been mistakenly segmented as background (omission, Type II error).

Table 1. Confusion matrix for binary segmentation

	Ground truth Positive	Ground truth Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

With the values explained above, obtained when computing the confusion matrix, different measures of error can be defined, as the true positive rate and true negative rate (the ones that are studied in this report), also false positive rate and false negative rate.

Other metrics implemented to evaluate the model's performance in this project are:

- **True positive rate (TPR):** also called sensitivity or recall, it is defined as the probability of a positive prediction of the test, conditioned on truly being positive.

$$TPR = \frac{TP}{TP + FN} \qquad \text{(Eq. 5.3.1)}$$

- **True negative rate (TNR):** also named specificity or selectivity, defined as the probability of a negative test, conditioned on truly being negative.

$$TNR = \frac{TN}{TN + FP} \quad \text{(Eq. 5.3.2)}$$

- **False negative rate (FNR):** it is defined as the probability that a true positive is missed by the test.

$$FNR = \frac{FN}{FN + TP} = 1 - TPR \quad \text{(Eq. 5.3.3)}$$

- **False positive rate (FPR):** it is the probability of obtaining a positive result when in fact the true Value is negative.

$$FPR = \frac{FP}{FP + TN} = 1 - TNR \quad \text{(Eq. 5.3.4)}$$

- **Accuracy:** this metric is used to measure the performance of the model, accuracy score consists of the number of correct predictions obtained.

$$Accuracy = \frac{Nº\ of\ correct\ predictions}{Total\ nº\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(Eq. 5.3.5)}$$

- **Precision:** measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives plus false positives).

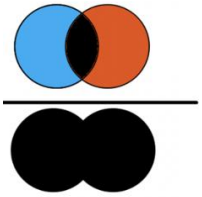
$$Precision = \frac{TP}{TP + FN} \quad \text{(Eq. 5.3.6)}$$

- **Recall:** measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives plus false negatives).

$$Recall = \frac{TP}{TP + FP} \quad \text{(Eq. 5.3.7)}$$

- **Intersection over union (IOU):** Also named Jaccard similarity coefficient, it is used to evaluate the performance of deep learning algorithms by dividing the overlap between the predicted and ground truth annotation by the union of these, which gives the information about how the predicted mask matches the ground truth mask. The greater the region of overlap, the greater the IOU (Rahman & Wang, 2016). As we obtain a higher IOU score, a better segmentation prediction is obtained.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad \text{(Eq. 5.3.8)}$$



6. Traditional segmentation

In this section, we present the results of implementing traditional smoke plume segmentation techniques. The focus is on threshold segmentation and clustering segmentation methods, which have been widely used in image processing and segmentation tasks. The evaluation metrics for each segmentation technique, including TPR, FPR and DSC are presented.

6.1. Threshold segmentation implementation

Different segmentation algorithms are analysed and tested with the images on Figure 7, to choose the best solution for smoke segmentation (Senthilkumaran & Vaithegi, 2016). The first one is the threshold segmentation technique, which is one of the easiest and quick to implement, but it can only be performed on grayscale images, thresholding is the process of creating binary images from the grayscale ones. These binary images have only two colours 0 or 255, black and white respectively, when the threshold value is selected either manually or automatically, the pixels below the threshold value (*T*) will be converted to black (0) and the ones above this value become white (255).

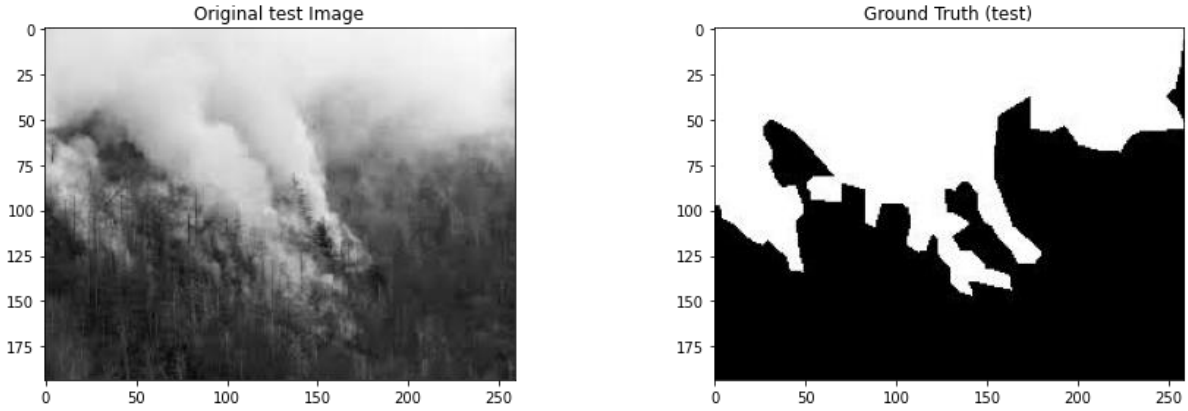


Figure 7. Tested image and mask for segmentation technique evaluation

From the thresholding segmentation technique, different methods are tested, first the manual thresholding, where a histogram is processed and the threshold value extracted manually by humans, which may lead to interpretation and human errors. The steps to be followed for manual image threshold segmentation are:

1. Convert the image to grayscale and visualize the histogram.
2. Since two clear clusters are obtained in the histogram (see Figure 8), the threshold value is defined by eye, by finding the value that separates the two clusters. Afterwards this threshold values is applied to the image.
3. Find the image contours (edges).
4. Create a mask using the largest contour.
5. Apply the mask on the original image to remove the background (see Figure 9)

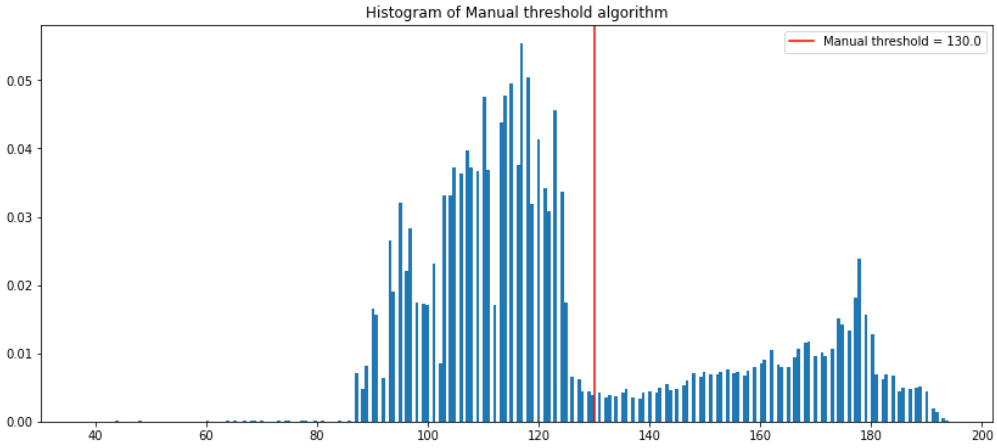


Figure 8. Histogram and thresholded value of manual thresholding technique

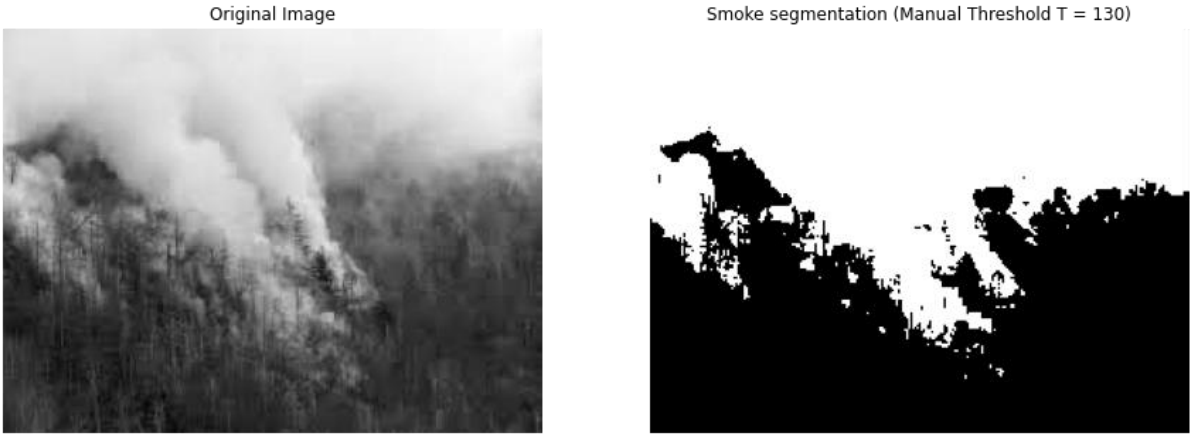


Figure 9. Smoke segmentation output for manual threshold segmentation

The evaluation metrics results from the manual thresholding method are shown in Table 2.

Table 2. Manual thresholding evaluation metrics

Manual segmentation results	
TPR	90.82 %
FPR	11.87 %
DSC	89.14 %

The Otsu's algorithm is also tested, which consists of an automatic segmentation approach that finds the optimum threshold value by the maximum inter-class variance (σ^2).

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \qquad \textbf{(Eq. 6.1.1)}$$

The steps to be followed to apply the Otsu's algorithm, in order to obtain the segmentation of Figure 11, are the following:

1. Compute the histogram and probabilities of each intensity level, where for an n-bit grayscale image (see Figure 10).
2. Set up initial weights.

3. Step through all possible thresholds.
4. The desired threshold is obtained by the maximum inter-class variance.

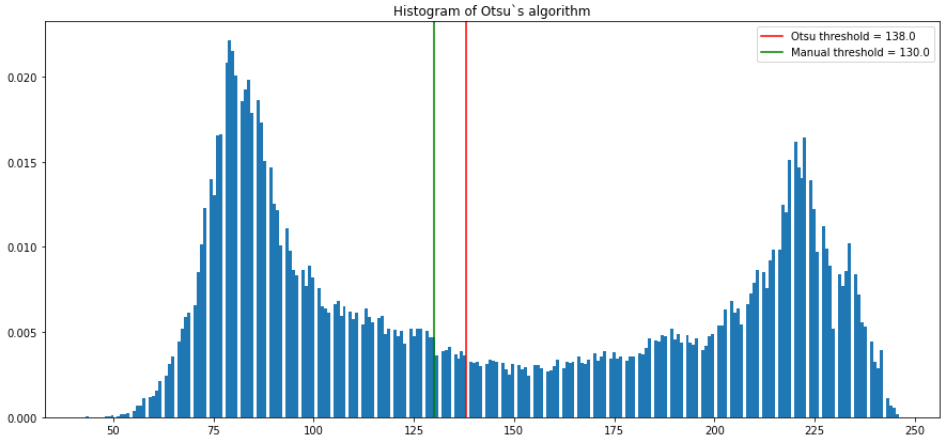


Figure 10. Histogram and threshold value obtained for Otsu's algorithm.

Figure 11. Smoke output for Otsu's threshold segmentation

The evaluation metrics results from the manual thresholding method are shown in Table 2.

Table 3. Otsu's thresholding evaluation metrics

Otsu's segmentation	
TPR	88.45 %
FPR	9.31 %
DSC	89.08 %

6.2. Clustering segmentation implementation

After testing segmentation thresholding methods, clustering segmentation algorithms was also tested, in this case applying K-means technique. To obtain the segmentation of Figure 12, the steps to be followed for K-means algorithm are:

1. Specify the number of clusters k to be found in data.
2. Set initial values for the cluster centroids (at random or prior knowledge).

3. Assign each observation to the nearest cluster (Euclidean distance).
4. Recompute the centroid of each cluster from the assigned observations.
5. Repeat steps 3-4 until no change in the centroids. Provide final clustering, where n is the number of iterations.

Figure 12. Smoke segmentation output for K-means segmentation

The evaluation metrics results from the manual thresholding method are shown in Table 4.

Table 4. K-means clustering evaluation metrics

Kmeans segmentation	
TPR	88.16 %
FPR	9.02 %
DSC	89.05 %

Results of traditional implementation

In conclusion, this section investigated and evaluated traditional smoke plume segmentation methods. The threshold segmentation techniques, including manual thresholding and Otsu's algorithm, provided reasonable results with respect to TPR, FPR, and DSC. Clustering segmentation using the K-means algorithm also yielded comparable performance. However, it is important to note that these traditional approaches have their limitations, such as the reliance on manually selecting thresholds or assumptions about cluster centroids. Therefore, further exploration and improvement are needed to enhance the accuracy and robustness of smoke plume segmentation.

Table 5. Performance comparison table for each traditional method tested

Evaluation metric	Segmentation method		
	Manual threshold	Otsu's threshold	K-means
TPR	90.82 %	88.45 %	88.16 %
FPR	11.87 %	9.31 %	9.02 %
DSC	89.14 %	89.08 %	89.05 %

7. Methodology of deep learning implementation

The methodology to implement deep learning for image segmentation, specifically using the UNet model, follows a defined pipeline explained in this section. This section also provides an overview of the different steps involved in the definition and implementation of the UNet model such as the image pre-processing done in the smoke dataset and an explanation of the encoder backbones that will be used afterwards in different tests.

7.1. Pipeline of UNet model development

This section explains the model development of a CNN for image segmentation, after researching the advantages and disadvantages of each architecture mentioned in section 3.2. Deep Learning image segmentation, UNet model is the one chosen to apply in this project. The model development follows the workflow in Figure 13. This workflow involves several steps, with each step contributing to the overall success of the project.

The first step, as detailed in Section 4, consists of collecting a suitable dataset of wildfire smoke images and creating the ground truth for each one of them to generate the dataset, in total 190 images with their corresponding ground truth masks configure the dataset. Afterwards this dataset is manually divided into two subfolders, with the criteria of obtaining a dataset with variability in the images in both cases, the train dataset, with 175 images and their corresponding masks, and the test dataset, with 15 images and their corresponding masks.

Further division of the training dataset is done to establish the validation and training subsets, this step is done inside the code algorithm. The validation subset is created by allocating 10% of the train dataset, corresponding to 17 images and their ground truth masks. The remaining 90% of the training dataset is assigned to the training subset, which has 158 images and their ground truth masks.

To prepare the dataset for training the neural network, a series of pre-processing steps are applied. These steps include resizing the images to a consistent size to all sub-datasets (training, validation and testing). On the other hand, we explore data augmentation, as it is particularly important because it increases the dataset size by generating additional variations of the original image, so the model has more variability to achieve a better training result. This data augmentation should only be applied to the training subset. To evaluate the performance of the system, both training and validation subsets are used in the model training process, using the defined metrics in Section 5. These metrics serve as objective measures to assess the accuracy and effectiveness of the model's predictions.

The output metrics obtained during the training process are of significant importance for making model evaluations and comparisons. They include quantitative results such as a plot illustrating the history metrics for each epoch, enabling a visual understanding of the model's progress, a confusion matrix and qualitative results, being the prediction outputs for the different datasets created.

By following this implementation workflow, the project aims to develop an accurate and robust UNet model for smoke plume segmentation in wildfires. The workflow ensures the utilization of a well-prepared dataset, proper model training and evaluation, and post-processing measures to achieve reliable and effective image segmentation results.

To implement this pipeline, different script files has been created in order to run the code of the unet algorithm. Where each one has a different function such as loading and creating the dataset, where the images are pre-processed, crate the training process, the UNet model or to extract the image predictions. The python script files created for the project are the following: *main*, *utils*, *dataset*, *unet_model*, *train*, *predict*, *evaluation*, *hyper_search*.

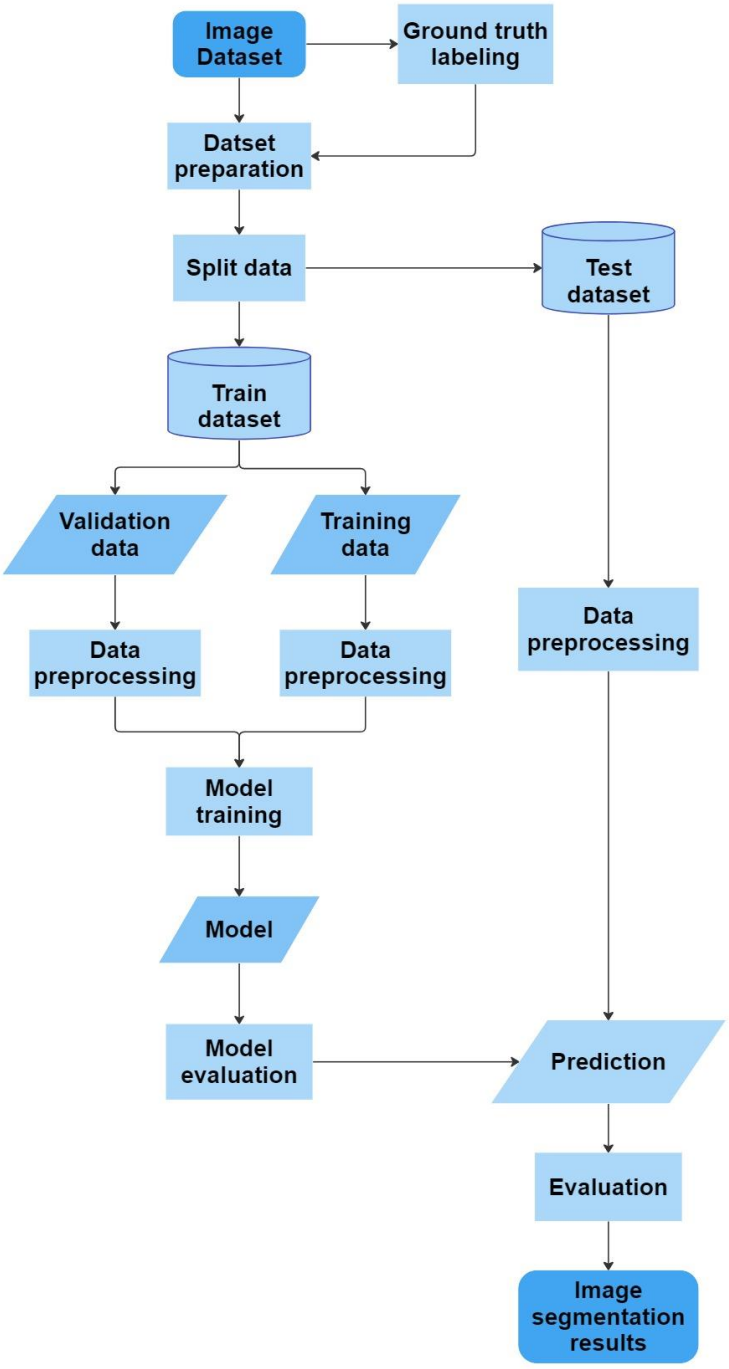


Figure 13. Image segmentation workflow

7.2. Image pre-processing

Image pre-processing is a fundamental step in image segmentation algorithms, aimed at enhancing the quality and suitability of the input images for segmentation. It involves applying a series of techniques to address challenges such as noise, contrast, illumination, image size, and irrelevant information. By reducing noise, enhancing contrast, and normalizing illumination, the pre-processing stage improves the accuracy and reliability of segmentation results. Additionally, resizing and scaling the images ensure consistency, while removing irrelevant information helps the algorithm focus on the desired regions or objects. For this purpose, several functions were created, which are described in the following paragraphs:

- *LoadImages*
- *CrearDataset*
- *DataAugmentation*
- *ResizeDataset*
- *DataToFloat*
- *LoadTestDataset*

The image pre-processing is done in the *dataset.py* script file, where the *LoadImages* function loads the images to a dictionary with its corresponding masks, a dictionary is a built-in data structure (associative array) that stores a collection of key-value pairs. When we have the image stored, they are converted to RGB, masks converted to grayscale, and all of them are normalized.

The validation and training subsets are computed using the *CrearDataset* function. The first step of this function after loading the data is to divide the dataset into training and validation subsets as mentioned above. Once both dictionaries are obtained, data augmentation is applied only in the training subset using the function called *DataAugmentation*. Afterwards, in order to have all the images with the same size (256, 256), resizing of the images and masks is done for both subsets by applying the function called *ResizeDataset*. Finally, images are converted to float variables, using the *DatasetToFloat* function. These datasets are then obtained in the *main.py* script by calling the *CrearDataset* function, having the *TrainDataset* and *ValDatset* variables as the output, which will be the ones used for training the model.

The test dataset is also called in the *main.py* script by applying the *LoadTestDataset* function, for the test images and masks the only pre-processing applied is resizing and then converting it to float variable, as the validation dataset, using the functions explained above, the output obtained is called *TestDataset*.

As explained, data augmentation is applied in the *TrainDataset* in order to enhance the dataset and introduce more diversity into the training process. Data augmentation techniques are widely extensive because it helps solving the small dataset limitations, it artificially expands the dataset by generating

additional images with variations. For this purpose, *Albumentation* library is used to implement several augmentation methods (*Albumentations: Fast and Flexible Image Augmentations*, n.d.), which include:

- Cropping the image randomly. The image is cropped to focus on specific regions of interest, enabling the model to learn from different spatial contexts.

- Adding Gaussian noise. By introducing noise to the images, the augmented dataset becomes more robust, which helps having a better model for capturing variations in smoke plumes.

- Blurring the image. Applying blur to the images helps in simulating different levels of smoke dispersion, improving the model's ability to generalize and segment smoke plumes accurately.

- Flipping the image horizontally. The images are horizontally flipped, expanding the dataset by providing mirrored versions. This aids in training the model to recognize and segment smoke plumes from various orientations.

- Changing the brightness and shadow of the images. The brightness and shadow levels of the images are adjusted, mimicking different lighting conditions. This allows the model to learn to segment smoke plumes effectively under varying environmental settings.

By applying these augmentation methods, the initial dataset is transformed into a larger and more diverse dataset (see Figure 14). This augmented dataset provides a richer set of training samples, enabling the UNet model to learn and generalize better in the task of smoke plume segmentation, and enhances the model's ability to accurately segment smoke plumes.

Figure 14. Augmentation examples. First column original images, other columns are random augmentation images.

7.3. UNet backbone architectures

This section presents the detailed structure and design choices of the UNet model architecture implemented in this thesis. The UNet model has a U-shaped architecture, which consists of a contraction path (encoder) and an expansive path (decoder).

The contraction path, or encoder, captures the contextual information and extracts high-level features from the input image. Initially the basic structure of UNet (Ronneberger et al., 2015) shown in Figure 2 was implemented, but did not achieve acceptable results for smoke segmentation. The basic contraction path of this UNet architecture applied begins with the input layer, followed by several convolutional blocks. Each convolutional block consists of two consecutive convolutional layers, batch normalization, and rectified linear unit (ReLU) activation. These blocks help capture and abstract high-level features at different levels of abstraction. This encoder part of the UNet model will be tested in section 8.1 and 8.6.3.

The basic block in the encoding is not enough to get the smoke segmentation task done properly. One way of solving this is using some previous CNN architecture already trained on millions of images for an image classification task. Fortunately, the modern classification architectures are built on blocks that are repeated until the networks reach a compact representation before taking the final classification decision. This is just the encoding part of the UNet approach, which takes this compact representation and grows it up until it reaches the initial dimensions. The fact that they are built in blocks allows us to connect the encoder and decoder after each block. It should be noted that the decoder blocks are much simpler than the encoders blocks that are responsible of extracting meaningful information. This scheme allows to improve the model's performance. The point, now, is to select a specific backbone architecture as the encoding pathway. Figure 15 shows a ball chart representing the accuracy related with the number of operations, being the computational complexity, and the model complexity of the backbones in the market. A trade-off between size and performance decision must be done to decide which approach to use. For this thesis a ResNet50 was initially chosen, which has a quite high accuracy and is not one of the most computationally complex architectures to implement. Afterwards a test to compare the performance of two different backbones is done in section 8.6.

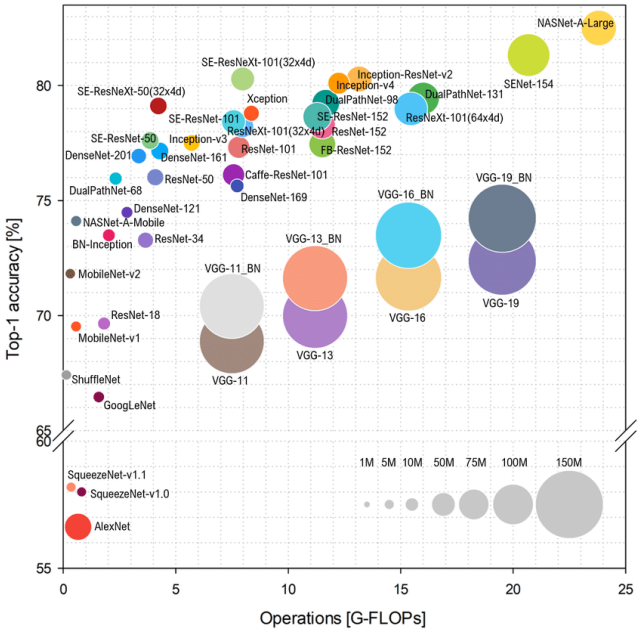


Figure 15. Ball chart of the accuracy vs the computational complexity of different deep neural network architectures, ball size represents the model complexity (Bianco et al., 2018).

The modification performed consists on replacing the encoder by a pre-trained CNN based on ResNet50 model (Wu et al., 2018), which has been trained on ImageNet (a large-scale dataset for image classification tasks). By using a pre-trained encoder, the UNet model can benefit from the learned representations and feature extraction capabilities of the ResNet50 model.

The ResNet50 is a CNN of 50 layers consisting of several sequential blocks, each block has multiple layers, including a convolutional, a batch normalization and an activation (ReLU), this structure is repeated three times. The main idea behind ResNet is the use of residual connections, also called skip connections. This connection allows the residual learning making easier to optimize the network's performance.

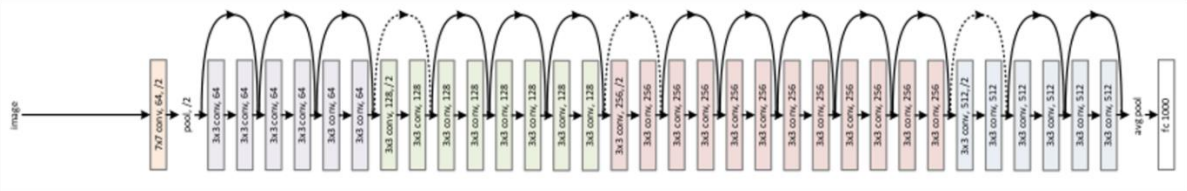


Figure 16. ResNet50 architecture (Wu et al., 2018)

On the other hand, the MobileNetV2 backbone was also considered for the encoder architecture. MobileNetV2 (Sandler et al., 2018) backbone is a CNN architecture specifically designed to have an efficient performance on mobile devices, due to it's low computational complexity. It was developed as an extension of the original MobileNet architecture, with the aim of improving efficiency and accuracy in image classification and other computer vision tasks.

The key concept of MobileNetV2 is that it introduces depth-wise separable convolutions, which splits the convolution process into separate depth-wise and point-wise convolutions. The depth-wise convolution applies a separate filter to each input channel, reducing computational cost. The point-wise convolution combines the filtered channels, allowing cross-channel information exchange. This architecture achieves a balance between computational efficiency and accuracy by reducing parameters and computations while capturing important spatial and cross-channel information. MobileNetV2 also incorporates inverted residual blocks, the function of these is to capture low-level and high-level features effectively. Shortcut connections are used to enable the learning of residual functions and optimize the network's performance.

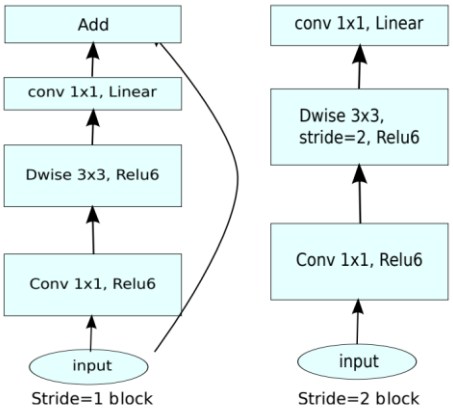


Figure 17. Convolutional blocks of MobileNetV2 architecture (Sandler et al., 2018)

When comparing ResNet50 and MobileNetV2, there are trade-offs to consider. ResNet50 offers higher accuracy due to its deeper architecture and utilization of skip connections, which allows for better feature extraction. However, it is computationally more complex compared to MobileNetV2. MobileNetV2, on the other hand, prioritizes computational efficiency and is designed for mobile devices. It achieves lower complexity by employing depth-wise separable convolutions, which reduce the number of computations while maintaining accuracy performance. The performance of these backbones will be compared and evaluated through experiments and a comparison analysis to determine which one is better for the requirements of the smoke plume segmentation task.

The encoder part of the UNet is the only one that will be changed in the tests where the UNet architecture is changed by a backbone. Regarding the bridge layer and the decoder part, will keep the same as the original UNet model for all the tests. The bridge layer is used as a connection between the contraction and expansive parts of the UNet model. The output of the last convolutional block from the contraction path is obtained and connected to the decoder.

The expansive path, or decoder, is responsible for up sampling the encoded features and recovering the spatial information to generate the final segmentation map. It consists of a series of transposed convolutional layers, which gradually increase the spatial resolution of the feature maps. The decoder path also incorporates skip connections that concatenate feature maps from the corresponding layers in the contraction path. These skip connections enable the decoder to access low-level and high-level features, facilitating precise localization and accurate segmentation.

The expansive path starts with the up sampling of the bridge layer using transpose convolutions. The up-sampling process increases the spatial dimensions while preserving the learned features. The up sampled layers are then concatenated with the corresponding layers from the contraction path, promoting the integration of low-level and high-level features. Following the concatenation, convolutional blocks are applied to refine and learn the representations in the expansive path. Each convolutional block in the expansive path follows the same structure as those in the contraction path, consisting of two convolutional layers, batch normalization, and ReLU activation. Finally, a 1x1 convolutional layer with a sigmoid activation function is employed to generate the final segmentation

mask. The sigmoid activation ensures that the output values are within the range of 0 to 1, representing the probability of each pixel belonging to the smoke plume.

The resulting UNet model is compiled and trained using the defined evaluation metrics and loss functions. Several utility functions to evaluate the model are defined in the *unet_model.py* script. These functions include TPR (True Positive Rate), TNR (True Negative Rate), IOU (Intersection over Union), Dice coefficient, and Dice loss. These functions and the once already provided by Keras, accuracy, precision and recall serve as evaluation metrics and loss functions to assess the performance and guide the training process of the model, explained in more detail in Section 5.

8. Results of the deep learning implementation

To achieve the best performance of the UNet model for this project, different tests and modifications were carried out. In the first tests, the original UNet model was applied (Ronneberger et al., 2015), the accuracy obtained was decreasing when epochs increased, which is the opposite performance expected when a model is learning weights, called *overfitting effect*. To improve the percentage of validation accuracy of the model, different modifications in the original U-Net architecture model have been done.

Adding a batch normalization layer after every convolution layer also improves accuracy. Batch normalization for the training process consists on normalizing the contributions to each layer instead of in the raw data, it is applied in order to be able of reducing the number of epochs in the training model, and a bigger learning rate can be used, making easier and faster the learning process.

8.1. Basic structure tests

Once the 'basic' structure of the UNet model is defined, the original architecture seen in Figure 2. UNet model architecture, a first test is performed with the aim of making a comparison between the performance of the UNet 'basic' structure and a UNet model with a ResNet50 backbone as the encoder structure. Both tests are performed with the same configuration parameters of learning rate value of 0.001, batch size of 16. As it is seen in Table 6 the test using ResNet50 encoder as a backbone in the UNet model obtains better performance than applying the original UNet model architecture. The validation metrics obtained for ResNet50 test achieves an accuracy of 91.99 % and 0.4389 loss value. In contrast, the original UNet test has a validation accuracy of the 86.73 % and a validation loss value of 0.7936, an 80.79 % higher than the one for ResNet50 test. The graphics of this metrics during each epoch, Figure 18 and Figure 19, also show that the original UNet model is more instable than the ResNet50, where the validation and training loss and accuracy are plotted.

Table 6. Performance metrics obtained for UNet model architecture comparing the original UNet structure and the UNet with ResNet50 backbone. Both tests trained for 200 epochs with a learning rate value of 0.01 and a batch size of 16.

Test	Loss	Val loss	Accuracy	Val accuracy	IOU	Val_IOU
Original UNet	0.05053	0.79361	0.98091	0.8673	0.9077	0.60074
ResNet50 UNet	0.00626	0.43897	0.99746	0.91993	0.98761	0.71701

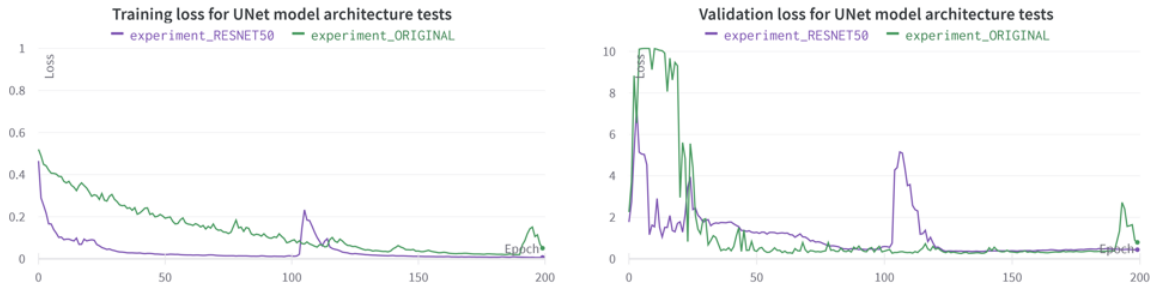


Figure 18. Training and validation loss performance of UNet model architecture test. In green the original UNet model structure. In purple the UNet model with a ResNet50 as a backbone.

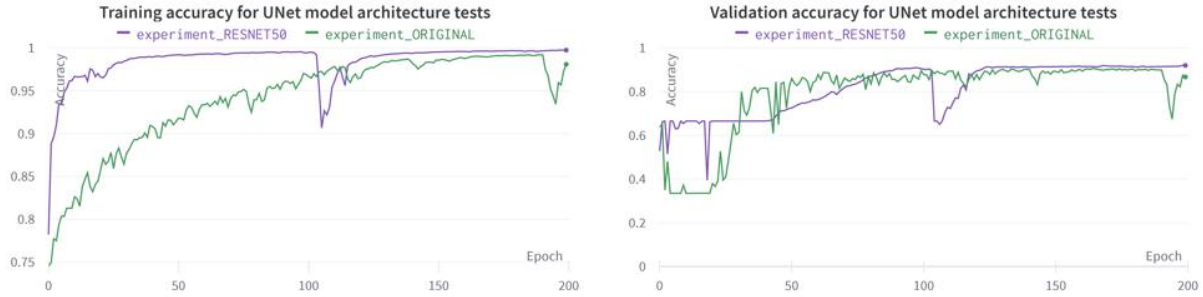


Figure 19. Training and validation accuracy performance of UNet model architecture test. In green the original UNet model structure. In purple the UNet model with a ResNet50 as a backbone.

The best performance results obtained in this comparison is from the UNet model using ResNet50. However, the metric values obtained are not enough to ensure a correct and reliable image segmentation algorithm, so the UNet model with ResNet50 architecture will be the one used in the following tests in order to improve model's performance. An example of an output prediction image from the test dataset for both tests is shown below, Figure 20 from original UNet test and Figure 21 from ResNet50 UNet test.

Figure 20. Prediction from test dataset for the basic structure test: Original UNet. Applying the original UNet architecture with a lr=0.001 and batch size=16.

Figure 21. Prediction from test dataset for the basic structure test: ResNet50 UNet. Applying the UNet architecture with a ResNet50 backbone with a lr=0.001 and batch size=16.

8.2. Loss function tests

Once the structure of the UNet model is defined, another test is carried out to decide the best metric to use for as a loss function in the model. A comparison is done between the binary cross entropy loss, which is the one generally used in machine learning basics and dice loss which is recommended when applying to image segmentation models.

To ensure the best loss function option, two different tests with different UNet model parameters are performed. A first comparison of the two loss functions with a constant learning rate of 0.001 and another one to ensure the decision of the loss function to be used is done with a learning rate scheduler, a function used to decrease the learning rate value every 1000 steps of the training process, starting from the initial value of 0.01 learning rate.

The parameters defined on the first test for both binary cross entropy and dice loss are the following:

- Batch size of 16
- 200 number of epochs
- ResNet50 as the backbone encoder
- Constant learning rate of 0.001

Changing the learning rate and the number of epochs, the second test is performed with the next parameters:

- Batch size of 16
- 200 number of epochs
- ResNet50 as the backbone encoder
- Adaptive learning rate (LR) starting from 0.01, which reduces the LR on a factor every 1000 steps of the training process.

In both comparison tests between the performance when using the binary cross-entropy loss function versus the performance of using the dice loss function, the models where the dice loss is applied gave better results than when applying binary cross-entropy. See Table 7, with the accuracy and loss obtained in all the tests. For the first test conducted, applying a constant learning rate of 0.001, the

loss obtained for the dice, with a value of 0.0062, is much smaller than the loss obtained for the binary cross-entropy, with a value of 0.036. Regarding the validation loss values obtained, the ones from dice loss also obtained better performance than the ones from the binary cross-entropy loss, but with less difference between them, obtaining a 0.1922 and 0.2788 respectively. For the accuracy values obtained in this first test comparing the loss functions, the training accuracy is higher in the case of dice loss function but the validation accuracy for binary cross-entropy test is slightly higher than the dice test.

A second test is performed to assure the results and the decision of which loss function chose for the application of UNet model for smoke plume segmentation, in this test a learning rate scheduler has been used, where the learning rate curve keeps decreasing every 1000 steps. The results obtained in this test showed similar values for both accuracy and loss performance of dice and binary cross-entropy test, see Table 7.

Table 7. Binary cross entropy loss function vs Dice loss function

	Constant learning rate of 0.001		Learning rate scheduler	
LOSS FUNCTION	**Binary Cross-Entropy**	**DICE**	**Binary Cross-Entropy**	**DICE**
Training loss	0.03618	0.00628	0.10421	0.1467
Training accuracy	0.98544	0.99626	0.95877	0.91973
Validation loss	0.27884	0.19220	0.31598	0.21353
Validation accuracy	0.91051	0.90177	0.88883	0.88229

The graphs obtained of the training and validation performance values during each epoch trained are shown in the figures below. For the first test, Figure 22 and Figure 23, the binary cross-entropy test curve is more instable and with more variability than the dice test curve, where it is observed that get to stabilize before in the final performance value and with less peaks on the last epochs.

When comparing the metric plots of the second loss test, the one using a learning rate scheduler, see Figure 24 and Figure 25, the training and validation curves for the dice test get to stabilize before and with less noise on the process. With this graphical representation it's clearly seen that the dice loss function for evaluating the loss performance of the model is much suitable for this project.

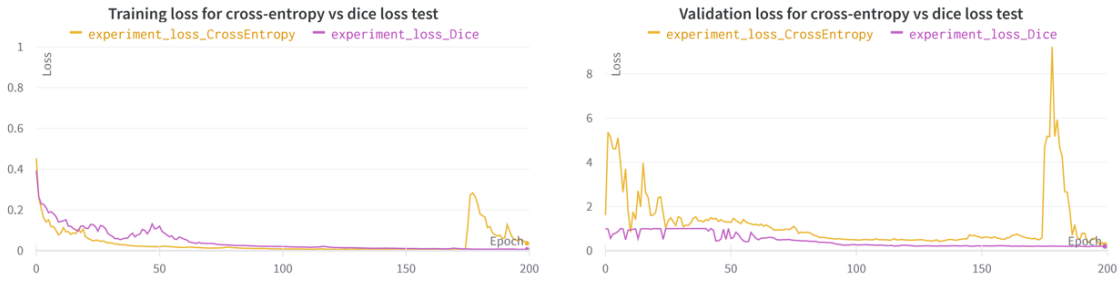


Figure 22. Training and validation loss performance of the loss test comparing the binary cross entropy and dice loss functions, with a constant learning rate of 0.001, tested during 200 epochs and a batch size 16.

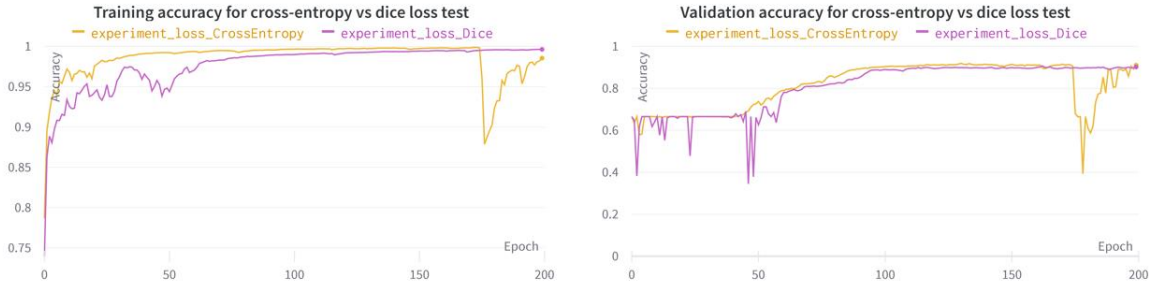


Figure 23. Training and validation accuracy performance of the loss test comparing the binary cross entropy and dice loss functions, with a constant learning rate of 0.001, tested during 200 epochs and a batch size 16.

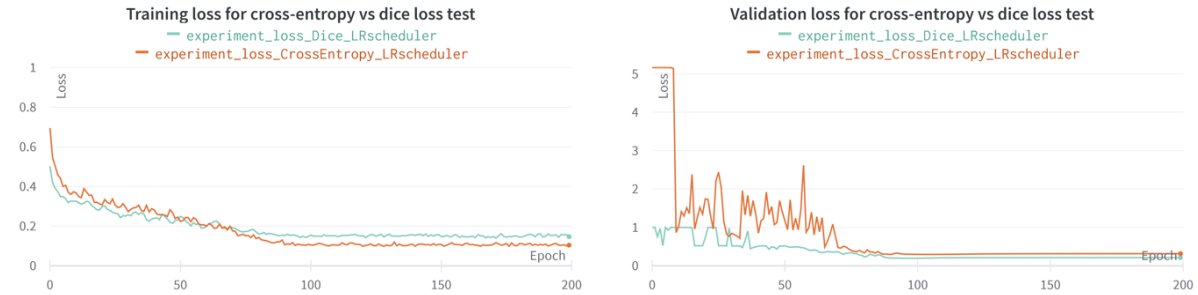


Figure 24. Training and validation loss performance of the loss test comparing the binary cross entropy and dice loss functions, with learning rate scheduler, tested during 200 epochs and a batch size 16.

Figure 25. Training and validation accuracy performance of the loss test comparing the binary cross entropy and dice loss functions, with learning rate scheduler, tested during 200 epochs and a batch size 16.

The best overall results are the ones where the dice loss metric is used; therefore, from now on the following tests will be done using this loss metric. In Figure 26 an output prediction example is presented of the dice test. When comparing it with the binary cross-entropy test, using a ResNet50 test performed in the previous section (Figure 21), is observed that the model starts segmenting with more precision the boundaries of the smoke plume but it should be improved in order to obtain more accurate results.

Figure 26. Prediction from test dataset for the dice loss function test applying constant learning rate of 0.001 and batch size of 16.

8.3. Learning Rate tests

Once the loss metric is defined, several tests were conducted with the basic UNet architecture to evaluate the impact of different learning rate values. Table 8 provides a summary of the results of each test, including metrics such as loss, accuracy, precision, recall, DICE coefficient, and IOU (Intersection over Union).

First of all, the model was implemented without a pretrained encoder, but after several tests and modifications was not possible to achieve a proper accuracy to have good output results of the predictions, after some research, the encoder was replaced by a ResNet50 autoencoder with pretrained weights. The configuration parameters of experiment_base3, with a learning rate of 0.0001 and a pretrained encoder, is the test that achieved the best overall performance results. The experiments named _base5 and _base6 were tests developed using a learning rate scheduler, which is used to decrease the learning rate every 1000 steps of the training process. It was expected to obtain a better performance of the model using this modification but as the quantitative results in Table 8 show that they did not improve the model's results, so a stable learning rate of 0.0001 will be used from now on.

Table 8. Base experiments of the UNet model with a ResNet50 autoencoder, training for 100 epochs and a batch size of 16

Nom	lr	Loss	Accuracy	Precision	Recall	DICE	IOU
_base1	0.01	0.43508	0.54911	0.42482	0.98093	0.56492	0.33227
_base2	0.001	0.26519	0.88519	0.86217	0.78197	0.73481	0.65028
_base3	**0.0001**	**0.17078**	**0.92231**	**0.89913**	**0.8649**	**0.82922**	**0.70323**
_base4	0.00001	0.22505	0.90426	0.87102	0.83806	0.77495	0.61306
_base5	scheduler 0.0001	0.185	0.92047	0.86078	0.90949	0.815	0.63406
_base6	scheduler 0.001	0.01903	0.8952	0.8728	0.8502	0.7843	0.6794

Figure 27, Figure 28 and Figure 29 show the results of the loss, accuracy and IOU. Figure 30 shows an example of some predictions made by the experiment with best performance (with a learning rate of

0.0001), to be able to visually evaluate the results. Additionally, in Figure 30 an example of a predicted image in the test dataset is show.

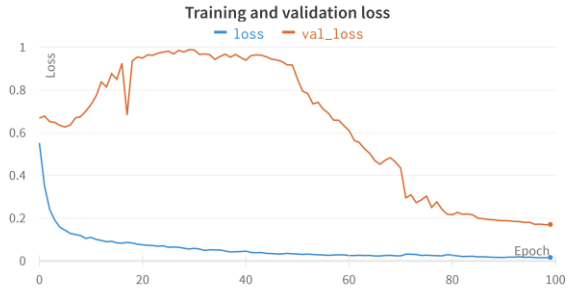


Figure 27. Training and validation loss using dice loss function, of experiment_base3 (trained for 100 epochs with a LR=0.0001)

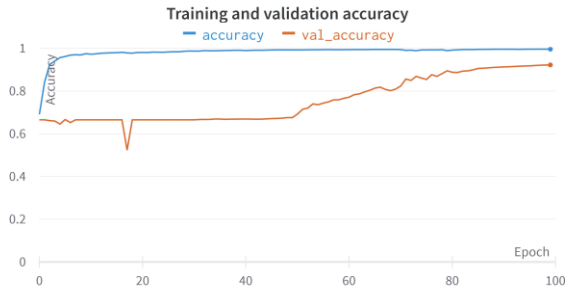


Figure 28. Training and validation accuracy using dice loss function, of experiment_base3 (trained for 100 epochs with a LR=0.0001)

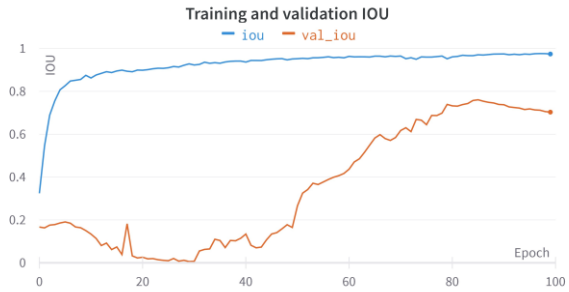


Figure 29. Training and validation IOU using dice loss function, of experiment_base3 (trained for 100 epochs with a LR=0.0001)

Figure 30. Smoke prediction of test_base3 (trained for 100 epochs with a LR=0.0001 and a batch size of 16)

8.4. Augmentation tests

As shown in the previous section, although the model is achieving a 92% accuracy, the output predictions are not satisfactory enough and the IOU still has a low percentage (70%). Therefore, several additional modifications are performed to obtain better results. The firsts actions that were made to reduce overfitting in the model was to use more images for training. This requires providing more data, therefore the data augmentation function, *Albumentations,* was used.

When loading the data and making the augmentation postprocessing of the images, they keep saved in order, so when training, in the same batch there are the same image but transformed (by applying the different augmentations methods explained in the section above), as the results of the previous tests shows an overfitted model the dataset have been improved to reduce this. One characteristic of a good dataset is that it has to have variability in it to give randomness to the smoke dataset, so the original images and the augmented images have been shuffled, this shuffle action is done only for the training dataset, after the augmentation is done, so each epoch have minibatches with different training images. The validation dataset is not included in the shuffle function, in this way it's possible to visualize the same image in the same epoch for the different tests performed, used to compare visually how the model is learning.

Table 9. Data augmentation experiments of the UNet model with a ResNet50 autoencoder, training for 100 epochs, a learning rate of 0.0001 and a batch size of 16

Test	Aug	Loss	Accuracy	Precision	Recall	DICE	IOU
_aug1	x1	0.16991	0.9167	0.87075	0.88204	0.83009	0.66147
_aug2	x2	0.17105	0.92061	0.89488	0.86433	0.82895	0.68667
_aug3	x3	0.14258	0.93632	0.89126	0.92226	0.85742	0.68727
_aug4	**x4**	**0.1334**	**0.93798**	**0.89273**	**0.92595**	**0.8666**	**0.69676**

Figure 31, Figure 32 and Figure 33 show the progression during the training epochs of the validation metrics loss, accuracy and IOU, respectively. The results provided have generally improved the model performance with respect the previous tests done only with the basic model. It can be observed that as more augmented data is used for the training process, the metrics stabilize before, the output metric results are quite similar between the data augmentation tests, but the experiment_aug4 has slightly better performance, which is the one with more augmentations applied, so also the one with more variability in the dataset for training the model.

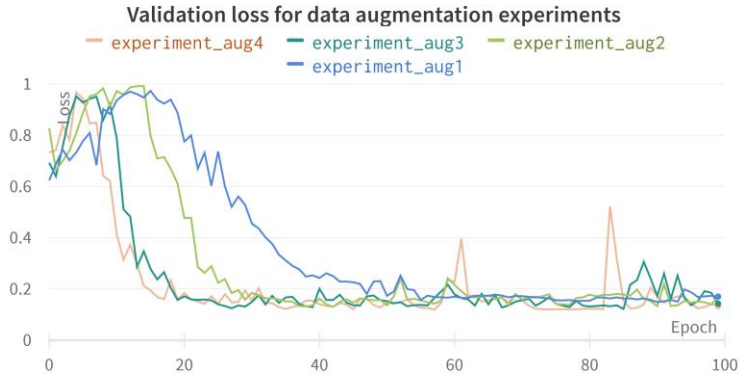


Figure 31. Validation loss using dice loss function, of data augmentation experiments (trained for 100 epochs with a LR=0.0001)

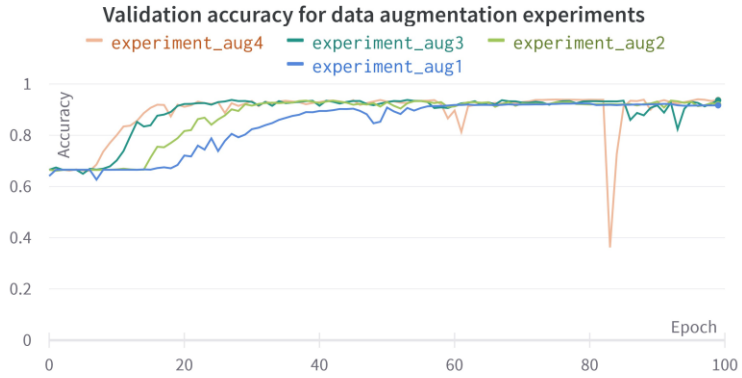


Figure 32. Validation accuracy using dice loss function, of data augmentation experiments (trained for 100 epochs with a .LR=0.0001)

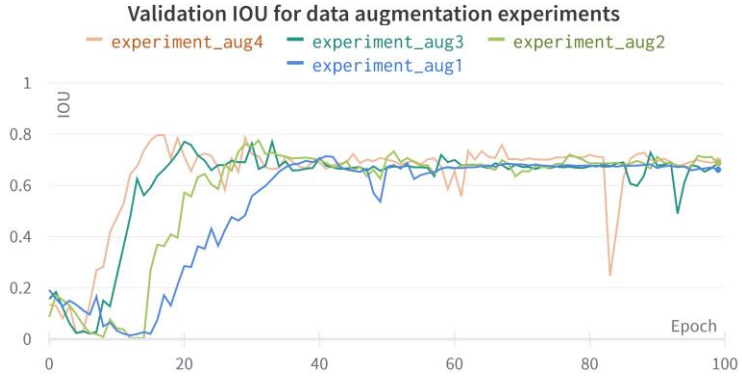


Figure 33. Validation IOU using dice loss function, of data augmentation experiments (trained for 100 epochs with a LR=0.0001)

The same test images predicted for the four different augmentation tests explained are shown below (Figure 34, Figure 35, Figure 36 and Figure 37) were a slightly improvement of the segmentation is observed when increasing the number of augmentations. It can be seen that the smoke plumes are detected but it still mismatches some objects, so more modifications must be done in order to avoid these mismatching segmented pixels.

Figure 34. Smoke predictions with data augmentation of test_aug1 (trained for 100 epochs with a LR=0.0001, applying 1 augmentation)

Figure 35. Smoke predictions with data augmentation of test_aug2 (trained for 100 epochs with a LR=0.0001, applying 2 augmentations)

Figure 36. Smoke predictions with data augmentation of test_aug3 (trained for 100 epochs with a LR=0.0001, applying 3 augmentations)

Figure 37. Smoke predictions with data augmentation of test_aug4 (trained for 100 epochs with a LR=0.0001, applying 4 augmentations)

8.5. Dropout tests

The data simplification method is used to reduce overfitting by decreasing the complexity of the model to make it simple enough that it does not overfit. Some techniques to carry out this method are to create decision trees, reduce the parameters of the CNN or add drop out layers to the CNN, this last option is the first one tested. First a drop out layer has been added in several tests after every convolutional layer, which did not provide the expected improved results. For this reason, finally the dropout layers are implemented after the activation of the contraction path.

Dropout is a regularization technique that randomly sets a fraction of input units to 0 at each update during training, which helps prevent overfitting. A comparison of different dropout values, for a model with a batch size of 16, a learning rate of 0,0001 and trained during 100 epochs, to have a quick overview of this modification performance, is shown in Table 10. The first test (_drop1) had a dropout of the 10% after the convolution layer called u8, and a second test (_drop2) two dropout layers are set, one after layer u8 and another after the last layer called u9. Considering the slightly better results obtained in the first test, it is stablished to only use one dropout layer, before the last layer.

Next step is to stablish which is the best percentage of dropout. We tested a range of values: 10 %, 20%, 50% and 75%. The dropout percentage number is the fraction of neurons that are randomly inhibited at every single training step. The best results were obtained using a a layer with 20% of dropout (see Figure 38, Figure 39 and Figure 40).

Table 10. Dropout experiments of the UNet model with a ResNet50 autoencoder, training for 100 epochs, a learning rate of 0.0001, a batch size of 16 and four data augmentations

Test	Dropout	Loss	Accuracy	Precision	Recall	DICE	IOU
_drop1	0.1 (after u8)	0.1551	0.92085	0.87264	0.894	0.8449	0.72571
_drop2	0.1 (after u8 & u9)	0.15583	0.91565	0.8219	0.95491	0.84417	0.62833
_drop3	**0.2**	**0.14078**	**0.92938**	**0.88298**	**0.90955**	**0.85922**	**0.70346**
_drop4	0.5	0.16612	0.92472	0.87975	0.89781	0.83388	0.66895
_drop5	0.75	0.19094	0.91771	0.90235	0.84564	0.80906	0.6874

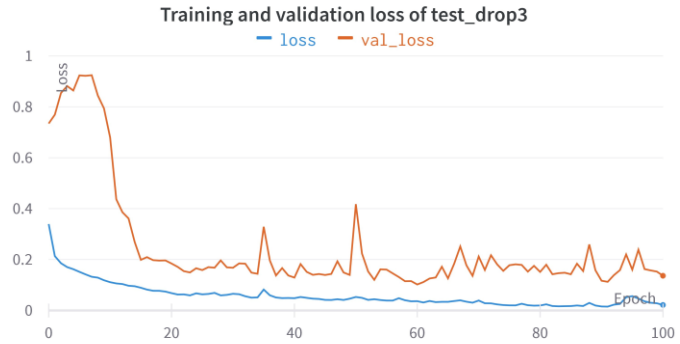


Figure 38. Training and validation loss using dice loss function, of dropout test (drop3) with 20% of dropout (trained for 100 epochs with a LR=0.0001 and 4 augmentations)

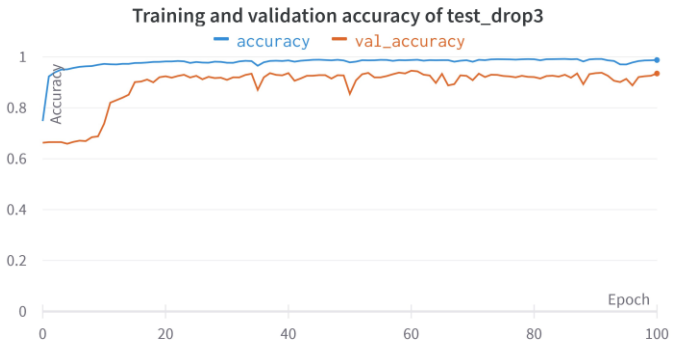


Figure 39. Training and validation accuracy using dice loss function, of dropout test (drop3) with 20% of dropout (trained for 100 epochs with a LR=0.0001 and 4 augmentations)

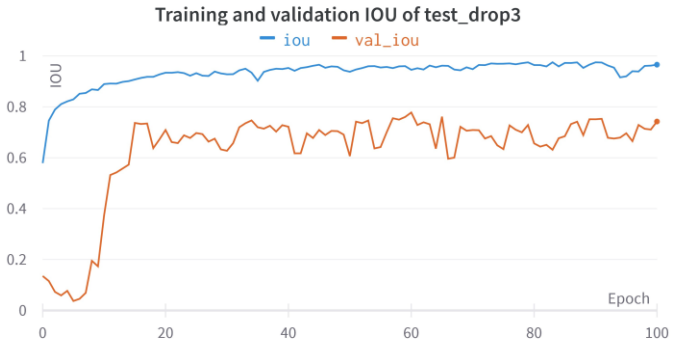


Figure 40. Training and validation IOU using dice loss function, of dropout test (drop3) with 20% of dropout (trained for 100 epochs with a LR=0.0001 and 4 augmentations)

Additionally, a visual evaluation is done for this dropout modification test, where the best predictions output are the ones for the modification of adding a 20% dropout layer (test _drop3). Actually it segments the smoke plume quite well, but a posterior test performing hyperparameter search will be done, in order to acquire a model that has less segmentation error, it is important to obtain segmented smoke plumes without 'holes' in the mask predicted, as it's observed that they appear in Figure 41.

Figure 41. Smoke predictions for dropout test of 20% (trained for 100 epochs with a LR=0.0001 with 4 data augmentations)

8.6. Hyperparameters search tests

In order to optimize the performance of the UNet model for smoke plume image segmentation, a series of hyperparameter search tests is performed. Hyperparameter sweeps involve training and evaluating a model with different combinations of hyperparameter values. The aim of these tests is to identify the best combination of hyperparameters that would provide optimal results of the UNet model. *Wandb* platform permits to automate the hyperparameter search process by creating a sweep function.

Three different tests were performed to obtain the best hyperparameter configuration, tested all during 150 epochs, configuration parameters of each test are summarized in Table 11. All these tests shared the same batch size, dropout, and learning rate range values. Where the range of search for the batch size is [16, 8, 4], the dropout rate values are [0, 0.1, 0.3, 0.5] and the range values of learning rate parameter are [0.01, 0.001, 0.0001, 0.00001]. Using the same parameters permits ensuring consistency in the search process. However, the main difference among the tests is the choice of encoder architecture and the use of transfer learning. Transfer learning in deep learning refers to the process of using of a previously trained neural network on a source task to enhance the performance on a new one.

A first test in order to check the performance of using transfer learning or not is done. Afterwards another test to compare the performance of two different backbone structures, ResNet50 and MobileNetV2 is also done.

Table 11. Configuration parameters for each hyperparameter sweep test performed

Test name	Transfer learning	Backbone (encoder)	Batch size	Dropout	Learning rate
Transfer learning	Yes	ResNet50	[16, 8, 4]	[0, 0.1, 0.3, 0.5]	[0.01, 0.001, 0.0001, 0.00001]
Non-transfer learning	No	ResNet50	[16, 8, 4]	[0, 0.1, 0.3, 0.5]	[0.01, 0.001, 0.0001, 0.00001]
Backbone	No	MobileNetV2	[16, 8, 4]	[0, 0.1, 0.3, 0.5]	[0.01, 0.001, 0.0001, 0.00001]

8.6.1. Transfer learning test

This first test used as a backbone a ResNet50 encoder with transfer learning. Transfer learning involves using the knowledge learned by a pre-trained model on a large dataset and applying it to a different but related task. The maximum validation accuracy achieved during the hyperparameter search was 85.55%. This value is obtained with a configuration consisting of a batch size of 8, dropout rate of 0.5, and a learning rate of 0.00001. Figure 42 illustrates the results of the 10 best parameters configurations found in the hyperparameter search for transfer learning tests. Each line is one of the tests performed, were the different hyperparameters of each configuration are shown in the vertical lines, the batch size is represented in the first vertical line where the values can be 4, 8 or 16. The dropout percentage implemented is represented in the second vertical line, this configuration values are 0, 0.10, 0.30 or

0.50. In the third vertical line the learning rate is shown where it can be 0.01, 0.001, 0.0001 or 0.00001. Finally, the last vertical line, shown in a colormap, the validation accuracy of each model configuration is represented, in order to visualize the performance evaluation of the models configurations for this test.

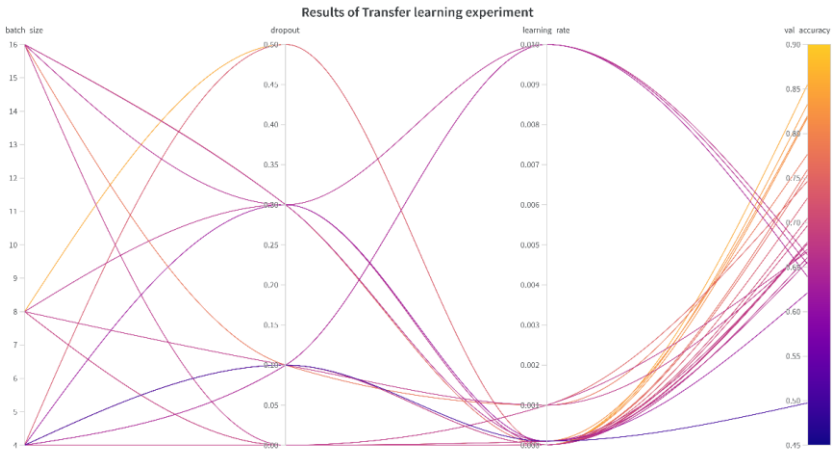


Figure 42. Hyperparameters search results for Transfer learning experiment

8.6.2. Non-transfer learning test

The second test (non-transfer learning) used a ResNet50 encoder without transfer learning. This test aimed to evaluate the performance of the UNet model when the encoder was trained from scratch, without the assistance of pre-trained weights. The comparison of this first two tests allowed to determine the benefits of using transfer learning in the specific application for this project of smoke plume image segmentation.

During the hyperparameter search of sweep3, the maximum validation accuracy achieved was 93.50%. This optimal performance achieved with a specific configuration of a batch size of 4, dropout rate of 0, and a learning rate of 0.00001, Figure 43 displays the results of the hyperparameter search for non-transfer learning tests.

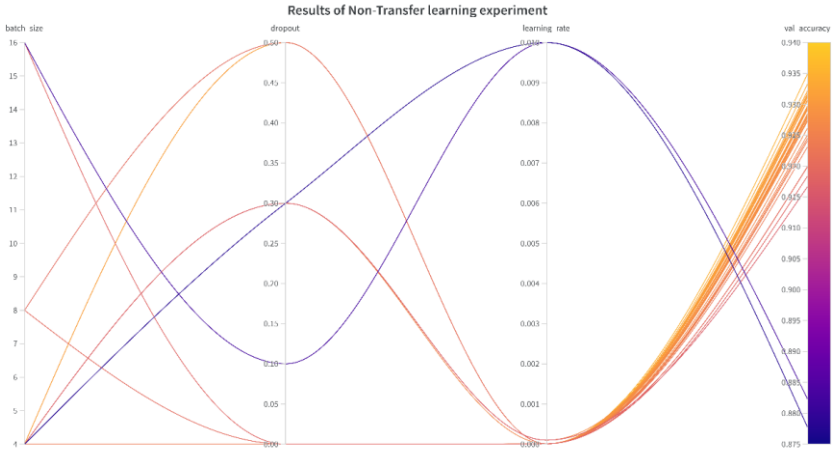


Figure 43. Hyperparameters search results for non-transfer learning experiment

When taking a first view to the transfer learning experiment metrics, it becomes evident that the validation metrics show a significant instability, the standard deviation of across the different tests is higher than non-transfer learning. The loss values remain relatively high (around 0.20) and fails to reach and fails to reach the same range of values as non-transfer learning tests. Additionally, the other metrics, such as accuracy also do not achieve desirable levels of performance. This metrics are shown in Figure 44, Figure 45, Figure 46 and Figure 47. The remaining metrics values are shown in the Annex A: Image segmentation results material. These observations indicate that despite achieving a reasonably high accuracy, the transfer learning experiment faces challenges in terms of stability and overall performance. The unstable validation metrics suggest that the model may be struggling to generalize well to unseen data.

In the non-transfer learning experiment metrics, it can be observed that the stability increased significantly compared with the transfer learning experiment, because most of the test performances fall in a very narrow range of values. A good behaviour is seen in both, training and validation metrics, obtained from the model training process without transfer learning on the ResNet50 encoder. The training loss value of the best hyperparameter configuration is of 0.0267, decreasing an 85 % with respect to the previous test that is using transfer learning. The validation loss is of 0.1306, as shown in Figure 44 and Figure 45 respectively. Moreover, the validation metrics of accuracy is also stabilized, achieving a training accuracy around 99 % (Figure 46), while the validation accuracy reaches a 93.50 % (Figure 47).

Although the use of transfer learning is generally recommended for image segmentation problems, the results found after performing this hyperparameter experiments indicate that training the ResNet50 encoder from scratch (non-transfer learning experiment), without relying on transfer learning, obtained significant improvements in stability and performance than when training the model applying transfer learning. The decrease in loss values and the enhanced accuracy values demonstrate the effectiveness of training the model without using transfer learning in this particular application for smoke plume image segmentation.

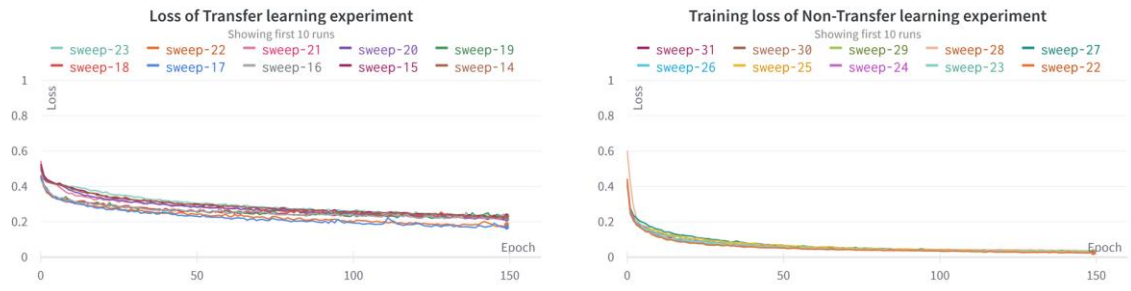


Figure 44. Training loss of the 10 bests parameters configurations performed on transfer learning tests, using a ResNet50 encoder. Left: with transfer learning. Right: without transfer learning.

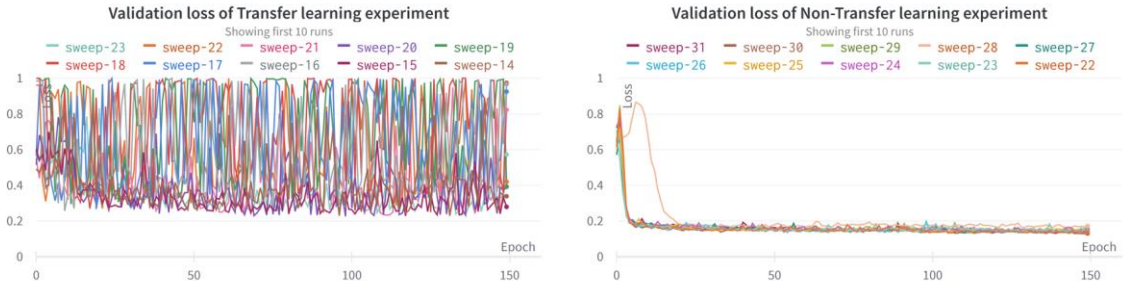


Figure 45. Validation loss of the 10 bests parameters configurations performed on transfer learning tests, using a ResNet50 encoder. Left: with transfer learning. Right: without transfer learning.

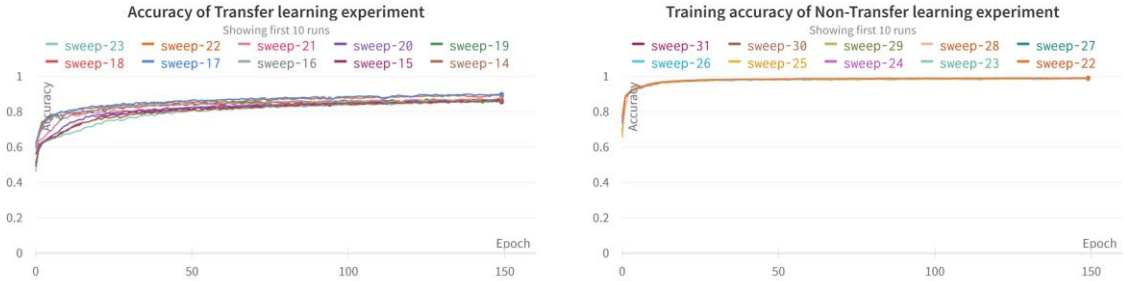


Figure 46. Training accuracy of the 10 bests parameters configurations performed on transfer learning tests, using a ResNet50 encoder. Left: with transfer learning. Right: without transfer learning.

Figure 47. Validation accuracy of the 10 bests parameters configurations performed on transfer learning tests, using a ResNet50 encoder. Left: with transfer learning. Right: without transfer learning.

8.6.3. Backbone test

Lastly, a third test was performed (backbone experiment), were the backbone architecture MobileNetV2 is employed as the encoder of the UNet, without transfer learning. The purpose of this test is to investigate the viability of using a simpler encoder structure like MobileNetV2, which offers computational efficiency while maintaining competitive performance. By comparing its performance with non-transfer learning test, that uses ResNet50 encoder, we can assess which trade-off between complexity and accuracy is better for the smoke plume image segmentation task.

Throughout the hyperparameter search conducted in this experiment, the highest validation accuracy achieved is 93.45%. This performance was accomplished by utilizing a configuration parameter

consisting of a batch size of 16, a dropout rate of 0, and a learning rate of 0.0001. The results obtained from the hyperparameter search for backbone tests are visualized in Figure 48.

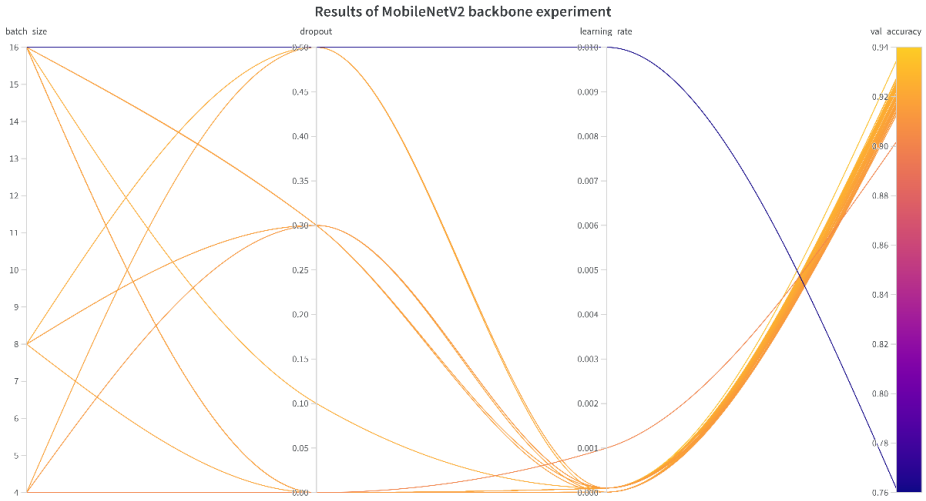


Figure 48. Hyperparameters search results for MobileNetV2 backbone experiment

The overview of the metrics obtained from this test, performed with MobileNetV2 encoder and without transfer learning, indicates that a stable model is obtained, show a stable progression of the metrics seen in the figures below of the loss and accuracy in both training and validation process during the epochs that the model has been trained. Comparing the overall results of hyperparameters search when using ResNet50 backbone (Figure 43) and MobileNetV2 backbone (Figure 48), both without transfer learning, we can conclude that the MobileNetV2 backbone has more consistency between all the results, so is more reliable than using ResNet50.

The maximum training loss achieved for MobileNetV2 test was of 0.0354, while the maximum training accuracy reached 98.33%, these metric plots are shown in Figure 49 and Figure 51.

In terms of validation, the maximum validation loss obtained was 0.1488. The model achieved a maximum validation accuracy of 93.45%, being similar to the non-transfer learning test using a ResNet50, indicating its ability to correctly classify smoke plume regions. The validation metrics plots are shown in Figure 50 and Figure 52.

Comparing the results of tests of ResNet50 and MobileNetV2 backbone (both without transfer learning), it is evident that the maximum validation accuracy achieved in both experiments is similar, with MobileNetV2 being slightly lower at 93.45% compared to ResNet50 that has a 93.50%. The maximum validation loss is also quite similar with a 0.1306 for ResNet50 being a little bit better than the 0.1488 of MobileNetV2 test.

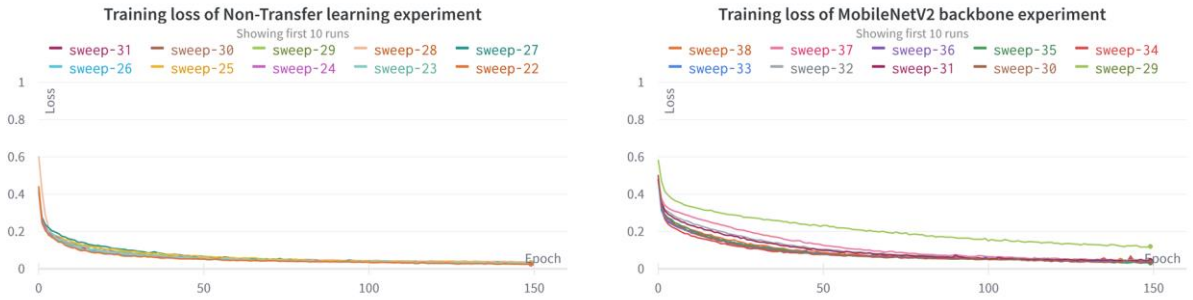


Figure 49. Training loss of the 10 bests parameters configurations performed on non-transfer learning tests, Left: ResNet50 backbone. Right: MobileNetV2 backbone.

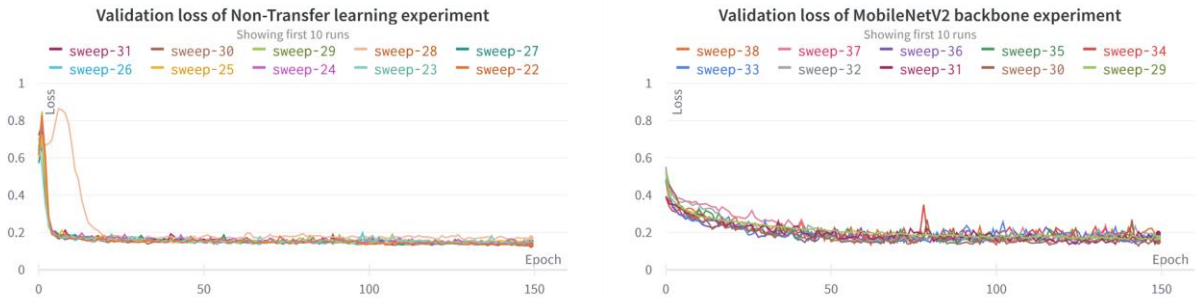


Figure 50. Validation loss of the 10 bests parameters configurations performed on non-transfer learning tests, Left: ResNet50 backbone. Right: MobileNetV2 backbone.

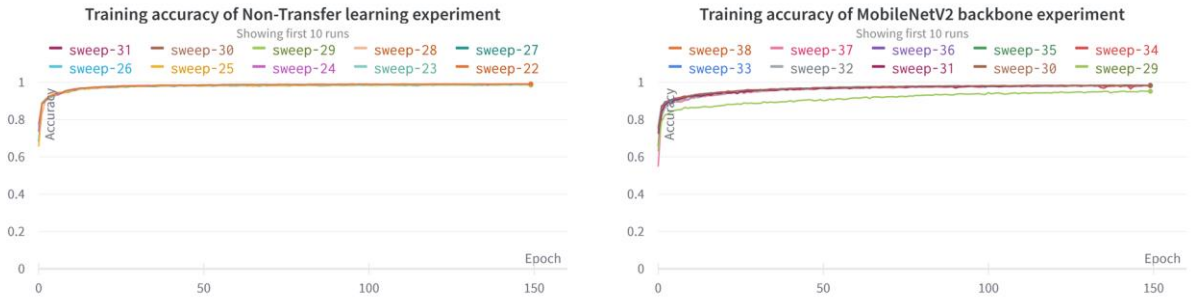


Figure 51. Training accuracy of the 10 bests parameters configurations performed on non-transfer learning tests, Left: ResNet50 backbone. Right: MobileNetV2 backbone.

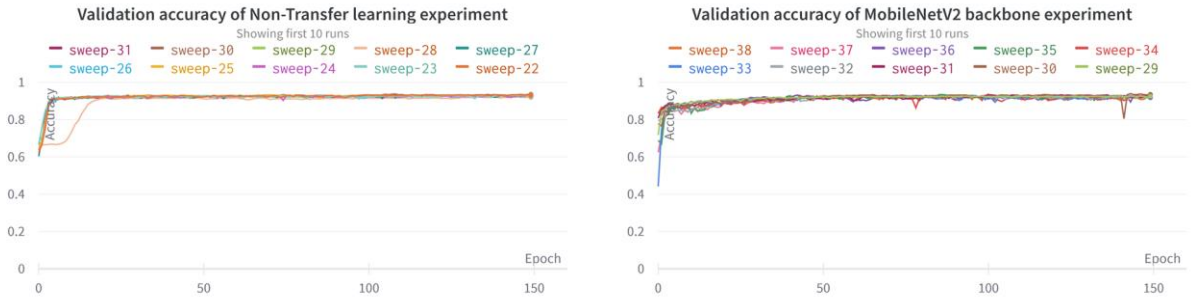


Figure 52. Validation accuracy of the 10 bests parameters configurations performed on non-transfer learning tests, Left: ResNet50 backbone. Right: MobileNetV2 backbone.

In conclusion, the hyperparameter search tests conducted in this study aimed to optimize the performance of the UNet model for smoke plume image segmentation. Three tests were performed to identify the best hyperparameter configuration for the task.

The transfer learning experiment employed transfer learning with a ResNet50 encoder. Although achieving a validation accuracy of 85.55%, the experiment exhibited instability in validation metrics, with relatively high loss values. Therefore, the use of transfer learning for the application of this project has been discarded.

In the non-transfer learning experiment, the ResNet50 encoder was trained from scratch without transfer learning. This test achieved a maximum validation accuracy of 93.50% and demonstrated significant improvements in stability and performance compared to transfer learning. The model trained without transfer learning showed a decreased loss values, enhanced all the other metrics, indicating its effectiveness for smoke plume image segmentation.

The backbone experiment was performed with a MobileNetV2 encoder without transfer learning, exploring a trade-off between complexity and accuracy. The experiment achieved a maximum validation accuracy of 93.45% and demonstrated being a stable model. While not matching the performance of non-transfer learning on ResNet50, MobileNetV2 indicated the viability of using a simpler encoder structure for computational efficiency while maintaining competitive performance.

Table 12. Results summary of hyperparameters search sweep tests

Test	Transfer learning	Backbone (encoder)	Max val loss	Max val accuracy
Transfer learning	Yes	ResNet50	25.67 %	85.55 %
Non transfer learning	No	ResNet50	13.06 %	93.50 %
Backbone	No	MobileNetV2	14.88 %	93.45 %

This comparison highlights the trade-off between complexity and accuracy when using different encoders. While MobileNetV2 offers computational efficiency, it falls slightly behind ResNet50 in terms of the quantitative values obtained from the tests, when checking qualitatively the predictions from the validation dataset obtained for the test with best performance (see Figure 53), the predicted smoke plume is similar where accurate prediction results are obtained. Also a prediction of the test dataset is shown in Figure 54, where a good prediction of the smoke plume is obtained.

Figure 53. Predicted output images of the validation dataset, using MobileNetV2 backbone without Transfer learning, lr=0.001, batch size=16 and 0% of dropout. First raw shows the ground truth, second row shows the prediction.

Figure 54. Prediction of test data image, using MobileNetV2 backbone without transfer learning, lr=0.001, batch_size=16 and 0% of dropout.

The selection of the encoder architecture should be based on the specific requirements of the smoke plume image segmentation task, considering factors such as computational resources and the desired balance between accuracy and efficiency. As the accuracy and output performance of both ResNet50 and MobileNetV2 encoders without applying transfer learning are similar, both could be used as the backbone for this project. The final backbone chosen is the MobileNetV2 due to its stability and lower computational cost.

If the computer used has a dedicated GPU hardware ResNet50 could be also a suitable choice. However, if this application is intended to be used in a mobile environment (ex. helicopters) the lower hardware needs of MobileNetV2 is a plus. As a guide in our experiments training the ResNet50 cost around 300 W, against 100 W for MobileNetV2, this numbers will also scale in terms of prediction. The computational cost of the tests are documented in the Annex A: Image segmentation results files, where a graph of the Watt consumption for each epoch of the model training process is plotted using the Wandb platform.

8.7. Final UNet model

This project focused on image segmentation of smoke plumes from wildfires using deep learning, where a UNet model architecture is implemented. Several tests and modifications were performed to optimize and improve the performance of the model. . Initially, the original UNet model was compared with a UNet model using a ResNet50 backbone as the encoder, where the test of the model using a backbone showed better performance in comparison of the original architecture.

Different loss functions were compared, including binary cross-entropy and dice loss. The dice loss function showed better results in terms of loss values and stability during training. It was chosen as the loss metric implemented for the evaluation of smoke plume segmentation task.

Tests were also conducted to evaluate the impact of different learning rate values. The experiment with a learning rate of 0.0001 and a pretrained ResNet50 encoder achieved the best overall performance. Learning rate schedulers were also tested but did not improve the results significantly.

Data augmentation techniques were applied to address overfitting and improve the model's performance, by increasing the variability in the training dataset. Augmentation experiments showed improvements in accuracy, precision, recall, dice coefficient, and IOU metrics as more augmented data was used for training.

A series of hyperparameter search tests were conducted to optimize the performance of the UNet model. The tests aimed to identify the best combination of hyperparameters for the task. Three tests were performed to compare the performance with different configuration combinations. Transfer learning was found to be less effective for this application, as it resulted in instability and lower performance compared to training from scratch. The experiments demonstrated the benefits of non-transfer learning, with the ResNet50 and MobileNetV2 encoders achieving high validation accuracies of 93.50 % and 93.45 %, respectively.

The choice of encoder architecture depends on specific requirements such as computational resources and the desired balance between accuracy and efficiency. Both ResNet50 and MobileNetV2 can be used as the backbone for the project.

The final UNet model architecture choice is the one where MobileNetV2 backbone is used without transfer learning applied. Trained during 200 epochs with the hyperparameters configuration of batch size 16, a dropout rate of 0 %, and a learning rate of 0.0001. This final decision is made due to its stability and computational efficiency, making it a reliable choice for smoke plume image segmentation task, a final validation accuracy of the 93.45 % is obtained, were the predicted smoke plume obtained are the ones in Figure 55.

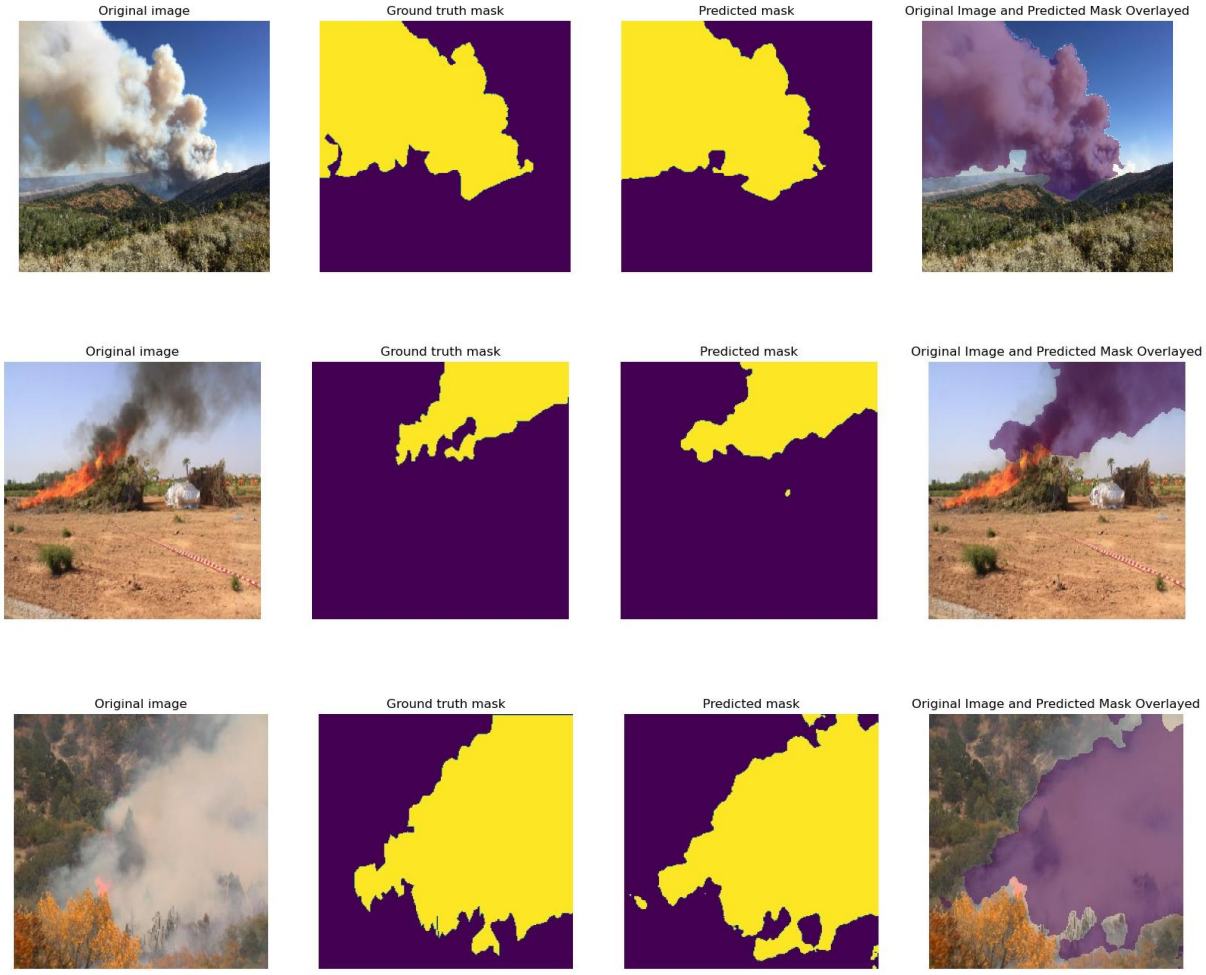


Figure 55. Predictions on test dataset of final UNet model chose (MobileNetV2 backbone with a learning rate of 0.0001, no dropout and a batch size of 16)

9. Application to video segmentation

Video segmentation involves the process of segmenting smoke plumes from video footage using a UNet architecture. In this section, we describe the video dataset used, the implementation of the video segmentation algorithm, and present the results obtained.

9.1. Video dataset

The video dataset consists of smoke plume scenes videos files in .mp4 format. From each video, its frames are extracted and saved for further processing.

9.2. Video segmentation implementation

The video segmentation algorithm is based on the UNet architecture from section 8.6.3, a UNet model with a batch size of 16, a 0 % of dropout and trained with 0.001 learning rate, this model has been proved the desired effect for image segmentation tasks. To implement the segmentation in videos, the first step is to obtain the frames per rate of each video, and the frames of each video are extracted and saved for their posterior segmentation using the UNet model created, when the segmentation is done, the reconstruction of the video with the proper configuration of its frames per rate is done to obtain the output video with the segmented mask of the smoke plume.

9.3. Video segmentation results

In this section, we presented the implementation and evaluation of a video segmentation algorithm for smoke plume detection. The algorithm, based on the UNet architecture, demonstrated promising results in accurately segmenting smoke plumes from video footage, see example from Figure 56 to Figure 60. The test shown in this section is of the video DJI_0012-012 file, it can be found the details in Annex A: Image segmentation results.

The evaluation metrics and visual comparisons indicated the effectiveness of the algorithm in detecting and segmenting smoke plumes, with high accuracy, precision and recall. The algorithm showcased robust performance across various scenarios, but certain limitations were also identified, suggesting opportunities for future improvements.

Overall, the video segmentation algorithm presented in this study contributes to the field of smoke plume analysis and can be utilized in applications such as wildfire monitoring, air quality assessment, and fire incident management, for the development of this project and further research, this video segmentation technique is used to continue developing the 3D reconstruction of smoke plumes.

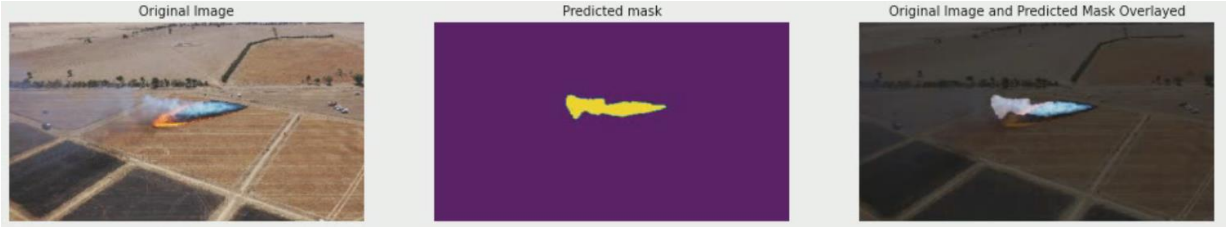


Figure 56. Smoke segmentation frame t=190s from DJI_0012-012.mp4 video

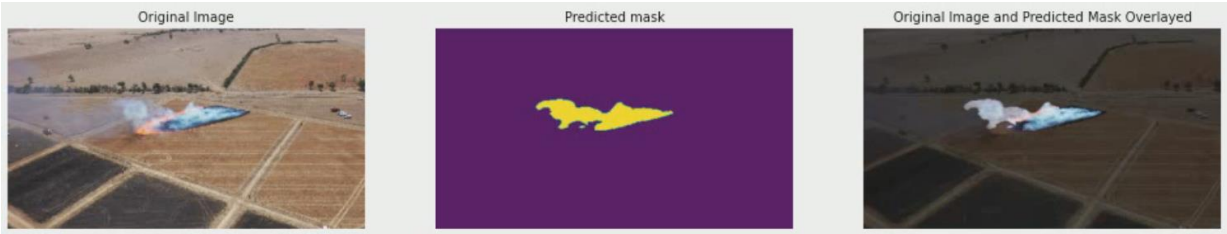


Figure 57. Smoke segmentation frame t=199s from DJI_0012-012.mp4 video

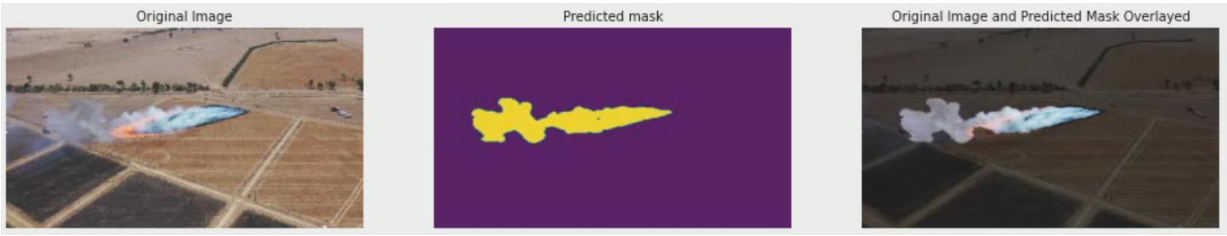


Figure 58. Smoke segmentation frame t=208s from DJI_0012-012.mp4 video

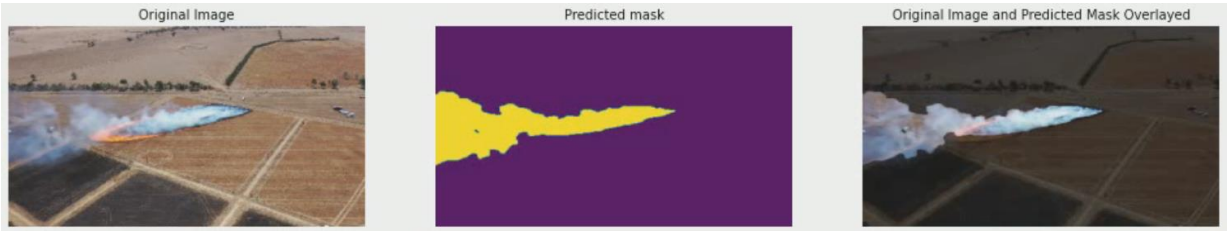


Figure 59. Smoke segmentation frame t=216s from DJI_0012-012.mp4 video

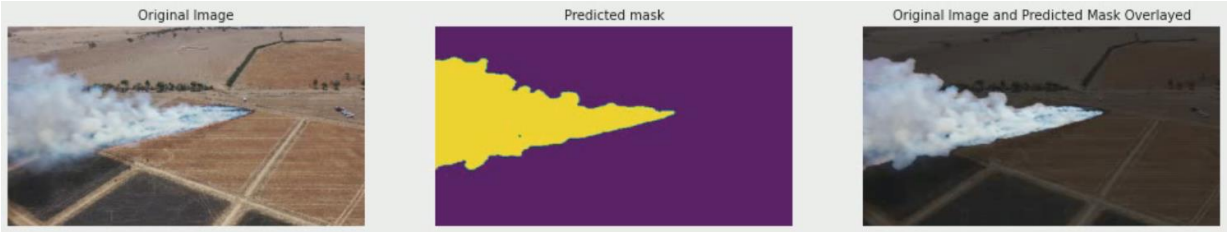


Figure 60. Smoke segmentation frame t=248s from DJI_0012-012.mp4 video

10. 3D reconstruction

This section focuses on the representation of 3D data and the methods used for 3D reconstruction. As the scope of this project is not specifically the 3D representation of smoke plumes, this section is developed extensively in the Annex B: 3D reconstruction. The main challenges in computer vision include creating realistic 3D models from 2D images, the reconstruction of smoke plumes is a complex research field, due to the dynamic nature of smoke and its blurred constraints.

A first research a discussion on the different types of 3D representations is developed, including multi-view images, volumetric or voxel grid, point cloud, and polygonal mesh representations. Each representation type has its advantages and applications in various fields such as object recognition, robotics, virtual reality, and entertainment industries.

Next, a section where the different algorithms used for 3D reconstruction are explored, which are categorized based on the information they extract from the environment. Photogrammetry, LiDAR, and Neural Radiance Fields (NeRF) are three main methods discussed.

- Photogrammetry is a technique that extracts 3D information from 2D images by reconstructing the 3D geometry of an object or scene. The algorithms used in photogrammetry include Structure from Motion (SfM), Multi-View Stereo (MVS), Iterative Closest Point (ICP), and Bundle Adjustment (BA).
- LiDAR, on the other hand, uses light to measure distances and creates a 3D point cloud representation of an object or scene. LiDAR algorithms such as ICP, Moving Least Squares (MLS), and Poisson Surface Reconstruction are used for 3D reconstruction.
- NeRF is a newer technique that represents a scene as a continuous 3D function using neural networks. It creates highly detailed and photorealistic 3D models by learning how objects would look from different angles. The steps involved in NeRF implementation include data collection, feature extraction, neural network training, and rendering.
- An improved version of NeRF called Instant NeRF, developed by NVIDIA, is also mentioned. Instant NeRF allows real-time 3D representation and reduces the training process time significantly. It incorporates multiresolution hash encoding, implicit hash collision resolution, online adaptivity, and an optimized neural network architecture.

The annex concludes by discussing different software options for 3D reconstruction, and three test performed were first a static object test is performed comparing the photogrammetry and Instant NeRF methods, a second test on the laboratory setup is done and a final test of a controlled smoke scenario test in a laboratory is done, this last test output obtained were no successful results. The research on this field should be considered as part of the future work.

Overall, this section provides an overview of the challenges and methods involved in 3D data representation and reconstruction, highlighting the importance of choosing the appropriate algorithms and software for specific applications.

11. Environmental impact analysis

The development of a wildfire smoke plume image segmentation model has the potential to reduce the impact of wildfires by accurately identifying the location and extent of the smoke plume. However, the development process itself could have environmental impacts, including increased energy use for data processing and model training, also has potential e-waste from discarding old or outdated computing equipment, after the use of it for the project, we have to take into account that this equipment is reused after this project, not discarded directly.

The greenhouse gas (GHG) emissions associated with the electricity consumption are calculated for the coding process, using CPU and the training process of the models, using GPU, the calculations are based on the ISO 14064 (Scipioni et al., 2012). The carbon footprint is calculated by multiplying the electricity consumption (in KWh) by the emission factor (in grams of CO_2eq per KWh). The electricity consumption is then converted to tons metric (tCO_2). The emission factor used is taken from the Catalonia's Government public data (*Factor d'emissió de l'energia elèctrica*, 2023). It has been considered that for the CPU consumption approximately 100 h have been allocated for the coding tasks with a mean consumption of 200 KW. Regarding the GPU, the electricity consumption is calculated for the tests exposed in this report, being a total of 18.77 tCO_2 with a mean consumption of 290 KW. A total of 23.95 tons of CO_2 is the carbon footprint released for this project.

$$\boldsymbol{Emision\ factor} = \mathbf{259}\ \frac{\boldsymbol{g\ CO2eq}}{\boldsymbol{KWh}} \quad \text{(Eq. 11.1)}$$

$$\boldsymbol{Electricity\ consumption} = \boldsymbol{KWh} * \boldsymbol{emission\ factor} \quad \text{(Eq. 11.2)}$$

$$\boldsymbol{Carbon\ footprint} = \frac{\mathbf{KWh}\ *\ \mathbf{emission\ factor}}{\mathbf{1000000}}\ [\boldsymbol{tCO2}] \quad \text{(Eq. 11.3)}$$

$$Carbon\ footprint\ of\ GPU\ (training) = \frac{290\ \text{KW} * 250\ \text{h}\ *\ 259\ \frac{g\ CO2eq}{KWh}}{1000000\ g} = 18.77\ tCO2$$

$$Carbon\ footprint\ of\ CPU\ (coding) = \frac{200\ \text{KW} * 100\ \text{h}\ *\ 259\ \frac{g\ CO2eq}{KWh}}{1000000\ g} = 5.18\ tCO2$$

$$Total\ Carbon\ footprint = 18.77\ tCO2 + 5.18\ \ tCO2 =\ 23.95\ tCO2$$

To mitigate these potential impacts, the development process should prioritize energy efficiency and sustainability. This could include using energy-efficient hardware and optimizing code for more efficient processing. Additionally, old or outdated computing equipment can be reused or recycled to reduce e-waste. It is important to assess and mitigate the potential environmental impact of the development of this project to promote sustainable development practices in technology.

12. Economic analysis

The economic analysis of the costs of this project is divided into three blocks, first the calculation of the hardware used is done, the only expense taken into account for this block is the GPU used for the models training process, it's the NVIDIA model RTX A2000 12GB, which has a cost of 1200€ and a desktop computer of 2000€ approximately.

Table 13. Hardware budget

Hardware	Concept	Price/Unit (€/unit)	Units	Price (€)
GPU - NVIDIA RTX 3090 24GB	Graphics Processing Unit	1200	1	1200
Desktop Computer	I7 32GB	~ 2000	1	2000
TOTAL				**3200 €**

The second block for calculating the total cost of the project is the software expenses, regarding the segmentation tasks, all the software's used are open-source, being the license completely free. When talking about the 3D reconstruction tasks, all the tested softwares were either open source or licence free. This includes NeRF instant NGP (Instant Neural Graphics Primitives), Blender and MeshRoom. For the office tasks, the Microsoft office 365 package is used, for reporting the process and results, having a cost of 69€ per year the basic pack.

Table 14. Software budget

Software	Concept	Price/Unit (€/unit)	Units	Price (€)
Python	Programming language	0	1	0
LabelMe	Image annotation tool	0	1	0
Labelbox	Image annotation tool	0	1	0
Wandb	Evaluation tool	0	1	0
Microsoft office	Office software tools	69	1	69
TOTAL				**69 €**

In order to calculate the cost of the labor to carry out this thesis, a first calculation of the real labor cost considering a junior engineer as the designer and developer of the project, supervision tasks of the project made by a senior engineer must be added also in the labor budget.

Table 15. Labor project cost: firs row show the real cost and second one the theoretical cost as if it was developed by a junior engineer

Labor	Concept	Price/Unit (€/unit)	Units	Price (€)
Junior engineer budget	Design and development	30	750	22500
Senior engineer budget	Management tasks	100	40	4000
TOTAL				**26500 €**

The final economic cost of the development of this project has been of 29769 € in total.

Table 16. Total budget cost of the project.

Concept	Price (€)
Hardware	3.200
Sofware	69
Labor	26500
Total	**29.769,00 €**

13. Conclusions

Initially the traditional implementation of smoke plume segmentation techniques was explored, specifically focusing on threshold segmentation and clustering segmentation. For threshold segmentation, both manual thresholding and Otsu's algorithm were tested. Manual thresholding involved visualizing the histogram of the grayscale image and manually selecting a threshold value to separate the smoke plume from the background. Although this method is straightforward, it is susceptible to interpretation and human errors. On the other hand, Otsu's algorithm offered an automatic approach to determine the optimal threshold value based on maximum inter-class variance. The evaluation metrics for manual thresholding showed a true positive rate (TPR) of 90.82%, false positive rate (FPR) of 11.87%, and Dice similarity coefficient (DSC) of 89.14%. Otsu's algorithm achieved a TPR of 88.45%, FPR of 9.31%, and DSC of 89.08%.

Another traditional technique tested is clustering segmentation where K-means algorithm was applied. This technique involved specifying the desired number of clusters, initializing cluster centroids, assigning observations to the nearest cluster based on Euclidean distance, and iteratively updating the centroids until convergence. The evaluation metrics for K-means segmentation demonstrated a TPR of 88.16%, FPR of 9.02%, and DSC of 89.05%.

The section on UNet model development for smoke plume image segmentation involved a series of tests and modifications to improve the model's performance. While the model achieved an accuracy of 93.45% on MobileNetV2 and successfully detected smoke plumes, there were still mismatches and room for improvement. Further modifications and simplifications of the model will be explored to enhance the segmentation results. The main point to improve accuracy was using the MobileNetV2 backbone, apply data augmentation extensively and adjust dropout and learning rate parameters.

After obtaining an accurate model for segmenting smoke images, the focus was on video segmentation for smoke plume detection. The implementation done of the video segmentation algorithm is based on the UNet architecture, which has proven effective for image segmentation tasks. The video segmentation results demonstrate the successful implementation and evaluation of the algorithm for smoke plume detection. The UNet-based algorithm shows promising performance in accurately segmenting smoke plumes from videos.

While the algorithm has shown effectiveness, it is important to acknowledge its limitations and identify opportunities for future improvements. Further research and development can focus on leveraging this video segmentation technique to advance the 3D reconstruction of smoke plumes. Smoke plume 3D reconstruction is a challenging problem due to the complex and dynamic nature of smoke. Extensive research of the different 3D representation and reconstruction methods has been performed. The implementation of 3D reconstruction requires careful consideration of data acquisition protocols. It is crucial to ensure consistent camera models, simultaneous image capture, and appropriate camera settings.

Regarding 3D reconstruction section a comprehensive overview of the methods, algorithms, and software used for 3D reconstruction of smoke plumes. The knowledge gained from this research will contribute to the development of an effective and accurate reconstruction of segmented smoke plume system.

14. Future work

There are different pipelines for continuing the development of a tool for smoke plume segmentation and reconstruction. Regarding the image segmentation process, additional tests could be done such as:

- Backbones: as we show the backbone decision is a crucial point, we should invest more time investigating different architectures for the encoder segmentation model, this involves experimenting with various pre-trained models such as VGG, Xception or Inception to assess their impact on segmentation performance.
- Hyper-parameter search: doing an extensive hyper-parameter search can help optimize the performance of the segmentation model.
- Comparison with other architectures: Compare the performance of the developed model with other CNN architectures such as DeepLab or Mask R-CNN. This comparative analysis can provide insights into the strengths and weaknesses of different approaches for smoke plume segmentation.

When talking about the video segmentation section, a future improvement that can be done is to extend the scope of the video segmentation techniques, where recurrent neural networks (RNNs) should be studied and applied, this techniques consists on segmenting with a temporal coherence, where the previous frame of the one being segmented is also taken int account, there are several techniques that can be applied such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU). For small datasets, this standard RNNs can be utilized, while for larger datasets, more advanced architectures such as Transformers can be explored, which are the ones used in industry nowadays.

As the scope of this project is focused on the segmentation part, previous research has been done in 3D reconstruction methods, this research can be used to continue with the 3D reconstruction of segmented smoke plumes. The future work guidelines to be done, is first perform tests applying 3D reconstruction to synthetic smoke plumes, this could be created by using Blender software, or any other 3d modelling software. A second step consists in testing on laboratory smokes, this can lead to a final application in real case wildfire smoke.

References

Albumentations: Fast and flexible image augmentations. (n.d.). Retrieved 1 March 2023, from https://albumentations.ai/

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*, 1–74.

Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(5), Article 5.

Benchamardimath, B., & Hegadi, R. (2014). A Survey on Traditional and Graph Theoretical Techniques for Image Segmentation. *International Journal of Computer Applications*, 38–46.

Beyond the pixel plane: Sensing and learning in 3D. (n.d.). Retrieved 11 May 2023, from https://thegradient.pub/beyond-the-pixel-plane-sensing-and-learning-in-3d/

Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access*, *6*, 64270–64277.

Biewald, L. (2020). Experiment tracking with weights and biases, software available from wandb. Com (2020). *URL Https://Www. Wandb. Com*.

Centre d'Estudis del Risc Tecnològic. CERTEC — UPC. Universitat Politècnica de Catalunya. (2023, June 4). https://certec.upc.edu/ca

Chollet, F. & others. (2018). Keras: The Python Deep Learning library. *Astrophysics Source Code Library*, ascl:1806.022.

Condorelli, F., Rinaudo, F., Salvadore, F., & Tagliaventi, S. (2021). A comparison between 3D reconstruction using nerf neural networks and mvs algorithms on cultural heritage images. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *43*, 565–570.

de Balestrini, E. F., & Guerra, F. (2011). New instruments for survey: On line software for 3D reconstruction from images. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, *38*, 545–552.

Dr.A.Usha Ruby. (2020). Binary cross entropy with deep learning technique for Image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(4), Article 4. https://doi.org/10.30534/ijatcse/2020/175942020

Egels, Y., & Kasser, M. (2001). *Digital Photogrammetry* (0 ed.). CRC Press. https://doi.org/10.4324/9780203305959

Factor d'emissió de l'energia elèctrica: El mix elèctric. (2023, June 4). Canvi climàtic. http://canviclimatic.gencat.cat/ca/actua/factors_demissio_associats_a_lenergia/index.html

Gonzalez, R. C., Woods, R. E., & Masters, B. R. (2009). Digital Image Processing, Third Edition. *Journal of Biomedical Optics*, *14*(2), Article 2. https://doi.org/10.1117/1.3115362

How to make annotation painless. (2023, January 10). https://kili-technology.com/data-labeling/how-to-make-annotation-painless

Huang, L.-K., & Wang, M.-J. J. (1995). Image thresholding by minimizing the measures of fuzziness. *Pattern Recognition*, *28*(1), Article 1.

Instant Neural Graphics Primitives. (2023). [Cuda]. NVIDIA Research Projects. https://github.com/NVlabs/instant-ngp (Original work published 2022)

Keras: Deep Learning for humans. (n.d.). Retrieved 3 March 2023, from https://keras.io/

Kien, D. T. (2005). A review of 3D reconstruction from video sequences. *University of Amsterdam ISIS Technical Report Series*.

Koutsoudis, A., Vidmar, B., Ioannakis, G., Arnaoutoglou, F., Pavlidis, G., & Chamzas, C. (2014). Multi-image 3D reconstruction data evaluation. *Journal of Cultural Heritage*, *15*(1), Article 1.

Kumar, A. (2021). *Different Types of CNNArchitectures Explained: Examples*. Retrieved from Vitalflux. Com: Https://Vitalflux. Com/Different-Types-of-Cnn-Architectures-Exp Lained-Examples/Weapon Detection in Surveillance Videos.

Labelbox | The Leading AI Platform for Building Intelligent Applications. (2023, Ferbruary 10). https://labelbox.com/

Maeda, K. (2012). Performance evaluation of object serialization libraries in XML, JSON and binary formats. *2012 Second International Conference on Digital Information and Communication Technology and It's Applications (DICTAP)*, 177–182.

Metcalfe, J. (2017). Learning from Errors. *Annual Review of Psychology*, *68*(1), Article 1. https://doi.org/10.1146/annurev-psych-010416-044022

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer Vision – ECCV 2020* (Vol. 12346, pp. 405–421). Springer International Publishing. https://doi.org/10.1007/978-3-030-58452-8_24

Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, *41*(4), Article 4. https://doi.org/10.1145/3528223.3530127

Mwiti, D. (2022, February 21). *Image Segmentation: Architectures, Losses, Datasets, and Frameworks*. Neptune.Ai. https://neptune.ai/blog/image-segmentation

Nex, F., & Rinaudo, F. (2011). LiDAR or Photogrammetry? Integration is the answer. *Italian Journal of Remote Sensing*, 107–121. https://doi.org/10.5721/ItJRS20114328

Pandey, R., Castillo, C., & Purohit, H. (2019). Modeling human annotation errors to design bias-aware systems for social stream processing. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 374–377. https://doi.org/10.1145/3341161.3342931

Pandey, R., Purohit, H., Castillo, C., & Shalin, V. L. (2022). Modeling and mitigating human annotation errors to design efficient stream processing systems with human-in-the-loop machine learning. *International Journal of Human-Computer Studies*, *160*, 102772.

Rahman, M. A., & Wang, Y. (2016). Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, & T. Isenberg (Eds.), *Advances in Visual Computing* (Vol. 10072, pp. 234–244). Springer International Publishing. https://doi.org/10.1007/978-3-319-50835-1_22

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, 234–241.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

Scipioni, A., Manzardo, A., Mazzi, A., & Mastrobuono, M. (2012). Monitoring the carbon footprint of products: A methodological proposal. *Journal of Cleaner Production*, *36*, 94–101. https://doi.org/10.1016/j.jclepro.2012.04.021

Senthilkumaran, N., & Vaithegi, S. (2016). Image segmentation by using thresholding techniques for medical images. *Computer Science & Engineering: An International Journal*, *6*(1), Article 1.

Sezgin, M., & Sankur, B. lent. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, *13*(1), Article 1.

Stathopoulou, E. K., Welponer, M., & Remondino, F. (2019). Open-source image-based 3D reconstruction pipelines: Review, comparison and evaluation. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-2/W17*, 331–338.

Torralba, A., Russell, B. C., & Yuen, J. (2010). LabelMe: Online Image Annotation and Applications. *Proceedings of the IEEE*, *98*(8), Article 8. https://doi.org/10.1109/JPROC.2010.2050290

Tseng, T., Stent, A., & Maida, D. (2020). Best practices for managing data annotation projects. *ArXiv Preprint ArXiv:2009.11654*.

VanRossum, G., & Drake, F. L. (2010). *The python language reference*. Python Software Foundation Amsterdam, Netherlands.

W&B Docs | Weights & Biases Documentation. (n.d.). Retrieved 11 June 2023, from https://docs.wandb.ai/?_gl=1*1kmbayt*_ga*MTE3MjU1NzEyOS4xNjg1MTkxNTA3*_ga_JH1SJHJQXJ*MTY4NjQ4OTY1My4zNC4xLjE2ODY0ODk2NTcuNTUuMC4w

Welcome to Python.org. (n.d.). Retrieved 1 February 2023, from https://www.python.org/

Wu, S., Zhong, S., & Liu, Y. (2018). Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, *77*(9), 10437–10453. https://doi.org/10.1007/s11042-017-4440-4

Zach, C. (2014). Robust Bundle Adjustment Revisited. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (Vol. 8693, pp. 772–787). Springer International Publishing. https://doi.org/10.1007/978-3-319-10602-1_50

Zhao, R., Qian, B., Zhang, X., Li, Y., Wei, R., Liu, Y., & Pan, Y. (2020). Rethinking Dice Loss for Medical Image Segmentation. *2020 IEEE International Conference on Data Mining (ICDM)*, 851–860. https://doi.org/10.1109/ICDM50108.2020.00094

Zheng, Y., Rao, J., & Wu, L. (2010). Edge detection methods in digital image processing. *2010 5th International Conference on Computer Science & Education*, 471–473.

Annex A: Image segmentation results

All the code scripts, evaluation metrics, computational costs, output segmented images and output segmented videos can be found in the following GitHub project's folder: https://github.com/albabaldrich/smoke_segmentation_unet.git

The folder structure for the U-Net algorithm application is the following:

- smoke_dataset
  - train (175 images - afterward divided into 90% train and 10% test)
    - images
    - masks
  - test (15 images)
    - images
    - masks
- video_dataset
- experiments
  - experiment_name.json
  - video_segmentation.json
- code
  - main.py
  - utils.py
  - dataset.py
  - unet_model.py
  - train.py
  - predict.py
  - prediction_output.py
  - evaluation.py
  - hyper_search.py
  - video_prediction.py
  - video_segmentation.py
- output
  - model
  - confusion matrix
  - prediction
  - video_output
  - results

Annex B: 3D reconstruction

Representation of 3D data

The main difficulties in computer vision are to create a realistic 3D model from a set of 2D images, in this section a study of smoke plume 3D reconstruction is performed, which is a challenging problem because of the complex, dynamic nature of smoke and its blurred constraints.

There are different three-dimensional representation types of an object or scene, Figure 61, they are defined as the way in which the 3D data is represented, the four main representations are (Kien, 2005):

- **Multi-view images 3D representation.** This representation of three dimensions space uses N elemental images, taken from different viewpoints, to reconstruct the object or scene. Multi-view 3D representation is used in computer vision for object recognition, localization, and tracking tasks. It is also used in robotics, virtual reality, and entertainment industries. Multi-view 3D representation can be obtained from photogrammetry or LiDAR methods. This 3D reconstruction type can capture detailed information about the object or scene from different points of view, which gives an accurate and complete 3D representation, which permits having a more robust and accurate model.
- **Volumetric or voxel grid 3D representation**. Volumetric 3D representation is a method of representing objects or scenes in three dimensions by dividing the space into a set of small volume elements, or voxels. In this method the properties (colour, opacity, physical properties, etc.) of the object or scene to represent are defined within each voxel. Volumetric 3D representation can be obtained from photogrammetry, LiDAR or NeRF methods. The main advantage of this method is that it handles complex shapes and structures, mostly used in biological tissues or natural environments, but it is expensive to store.
- **Point cloud 3D representation**. This 3D representation consists of obtaining a collection of individual points, each one with its specific location in the 3D space, its colour and reflectance, this group of individual points collected is called point cloud. Point cloud can be obtained by photogrammetry, 3D scanning, LiDAR or NeRF methodology. The advantage of applying point cloud representation is that it handles complex shapes and structures without losing precision and accuracy.
- **Polygonal mesh 3D representation - Photogrammetry/LiDAR**. Polygonal mesh can represent only hard surfaces (not suitable for medical imaging applications, for instance).

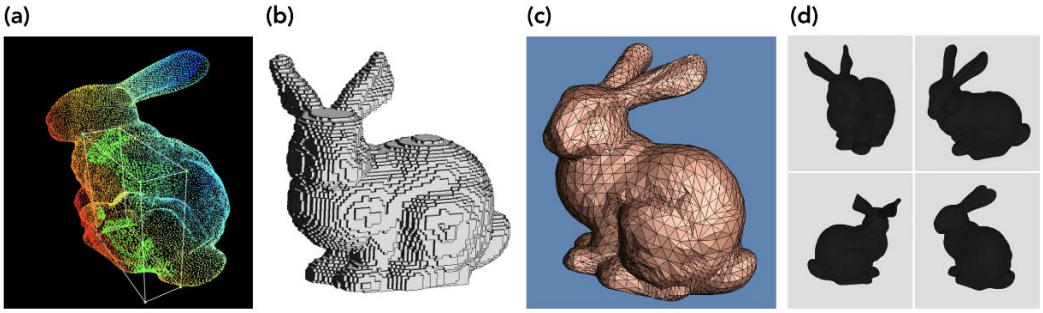


Figure 61. Representation of 3D reconstruction data. (a) Point cloud, (b) Voxel grid, (c) Triangle mesh, (d) Multi-view representation (*Beyond the Pixel Plane: Sensing and Learning in 3D*, n.d.)

3D reconstruction methods

There are several algorithms that can be implemented to obtain 3D information directly from the environment or object, the algorithm used in this project will be chosen depending on the requirements and constraints of segmenting wildfire smoke plumes. The algorithms used to obtain 3D reconstruction can be divided depending on how they obtain information about the environment, such as photogrammetry, Lidar or NeRF.

Photogrammetry

Photogrammetry is the science and technology of extracting 3D information from 2D images (Egels & Kasser, 2001). It involves the process of measurement of the objects and their environments through the use of imaging techniques. The aim of this method is to generate a 3D model with meshes and textures and it's stored in a way that traditional 3D tools can use it, using photographs of an object or scene taken from different angles to reconstruct its 3D geometry. This process consists of first identifying the corresponding points in the images, applying one of the algorithms for this method to compute the 3D coordinates of the points based on the positions and angles of the photos taken. Photogrammetry is applied in many fields such as archaeology, architecture, or film production in order to perform 3D animation, VR or AR applications, where an accurate 3D model of real-world objects or scenes are required. Photogrammetry uses photographs to reconstruct the 3D geometry of an object or scene.

The algorithms for 3D reconstruction that can be applied with photogrammetry technology are the following:

- *Structure from Motion (SfM)*: is the process of estimating the 3D structure of a scene or object from a set of 2D images. The principle of SfM is to recover the camera poses and the 3D positions of the scene points simultaneously. This is achieved by establishing correspondences between the 2D image features across different images and using geometric constraints to estimate the camera motion and scene geometry.

- *Multi-View Stereo (MVS)*: this method aims to reconstruct the 3D shape of an object by fusing multiple images taken of the object from different points of view.
- *Iterative Closest Point (ICP):* this algorithm consists on aligning two 3D point clouds by minimizing the distance between the corresponding points, used to refine the 3D model by aligning it with the observed images. In the case of applying ICP with photogrammetry data, it is needed to generate point clouds using this technology even though depending of the final application LiDAR or NeRF are other options.
- *Bundle Adjustment (BA)*: is an optimization technique that refines the camera parameters and 3D point locations to minimize the reprojection error. It is usually applied in conjunction with SfM or MVS to refine the 3D model obtained with one of this two algorithms explained previously. LiDAR and NeRF algorithms do not require to apply BA because their algorithm outputs a precise and accurate representation of the 3D object or scene.

Light Detection and Ranging (LiDAR)

LiDAR uses light to measure distances and create a 3D point cloud representation of an object or scene (Nex & Rinaudo, 2011). This methodology uses a laser emitting light pulses that bounce off surfaces and return to the sensor, to be detected, this sensor measures the time execution of this process that permits calculating the distance to each point.

3D reconstruction LiDAR algorithms are used when precise 3D measurements of the environment are required for the 3D reconstruction, such as autonomous vehicles.

LiDAR can be applied to obtain filtering, segmentation, classification, reconstruction or registration algorithms, this project is focused on the 3D reconstruction ones, some of the LiDAR algorithms used for 3D reconstruction are:

- *Iterative Closest Point (ICP)*: this algorithm explained previously, usually it is applied in a combination of different 3D reconstruction algorithms.
- *Moving Least Squares (MLS)*: it is a mesh-based algorithm.
- *Poisson Surface Reconstruction*: it is also a mesh-based algorithm where a Poisson equation is used to generate a smooth 3D model from the LiDAR point clouds obtained.

Neural Radiance Fields (NeRF)

In 2020 a new technique developed by researchers in the University of Berkeley, Google Research and University of California (Mildenhall et al., 2020) changed the paradigm of 3D reconstruction, modifying the conventional way to handle this data in machine learning, generating a scene as a continuous 3D function, called neural radiance fields (NeRF). Instead of a traditional 3D model as photogrammetry,

NeRF uses machine learning to create the radiance field, with this radiance fields it can be rendered new viewpoints of an object or scene from new angles, the radiance field learns and can know how the object would look like from any angle in order to generate a highly detailed photorealistic 3D model from a set of 2D images, representing a scene as a continuous 3D function, which is called neural radiance field (NeRF).

The potentiality of NeRF for 3D reconstruction is that, differently from classical photogrammetry, it is able to 3D reconstruct objects that present features that could lead to a failure of the photogrammetric process, such as thin objects (trees, leaves), or reflecting (metal) objects, etc. The network can represent detailed scene geometry with complex occlusions, without any background isolation or masking (Condorelli et al., 2021). It's mostly used in computer graphics and virtual reality applications, where highly detailed, photorealistic 3D models are needed.

NeRF models the radiance field and density of a scene from a set of input images within the weights of a neural network and can render high-resolution photorealistic novel views of real objects and scenes from RGB images captured in natural settings (Mildenhall et al., 2020). The first step when working with NeRF is to collect the data properly, a dataset of 2D images from different viewpoints of the object or scene desired to reconstruct is needed. It can be from videos of the same scene from different viewpoints, it's important to obtain these videos synchronized in time. Once the dataset is properly built, feature extraction process must be done, with a five-dimension input of the camera respect the scene, corresponding to the spatial location in three dimensions and the viewing direction the features are obtained and after an output of four dimensions is acquired. The input variables are used to train a fully connected multilayer perceptron neural network with 9 layers and 256 channels, the output obtained from the training consists of the RGB values representing the emitted colours and the volume density that determines the opacity of the radiance field. Finally, the rendering step is done, different rendering techniques (volume rendering and rendering loss) are applied to transform the output value into an image.

The (probability) volume density indicates how much radiance (or luminance) is accumulated by a ray passing through (x, y, z) and is a measure of the "effect" this point has on the overall scene. Intuitively, the probability volume density provides the likelihood that the predicted colour value should be considered.

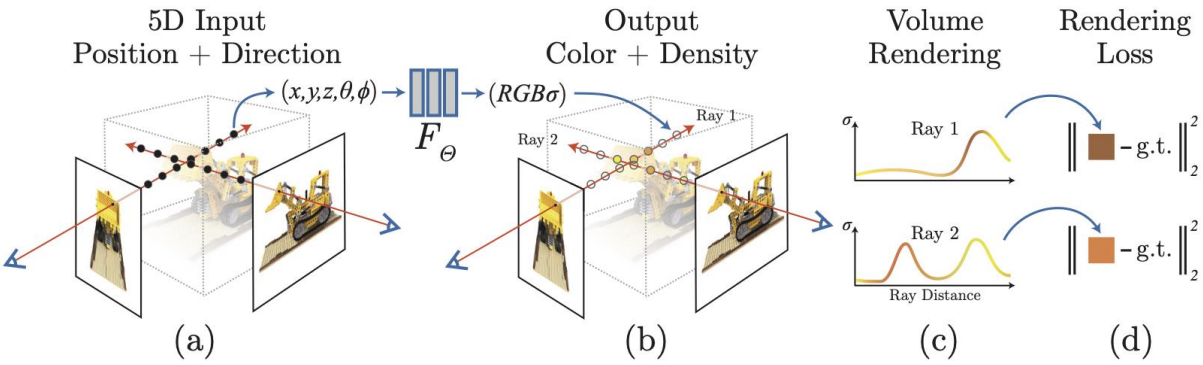


Figure 62. Procedure of neural radiance field scene representation and differentiable rendering.

NeRF implementation steps

The steps to be followed to reconstruct a 3D scene with NeRF are the following, also explained visually in Figure 62:

1) Data collection - Collect a set of 2D images of an object or scene from different viewpoints.
2) Feature extraction - synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (Figure 62.a)
3) Neural Network training (MLP) - feeding those locations into a multi-layer perceptron to produce a colour and volume density (Figure 62.b). The neural network architecture is shown in Figure 63.
4) Rendering - finally use volume rendering techniques to composite these values into an image. This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images. (Figure 62.c and Figure 62.d)

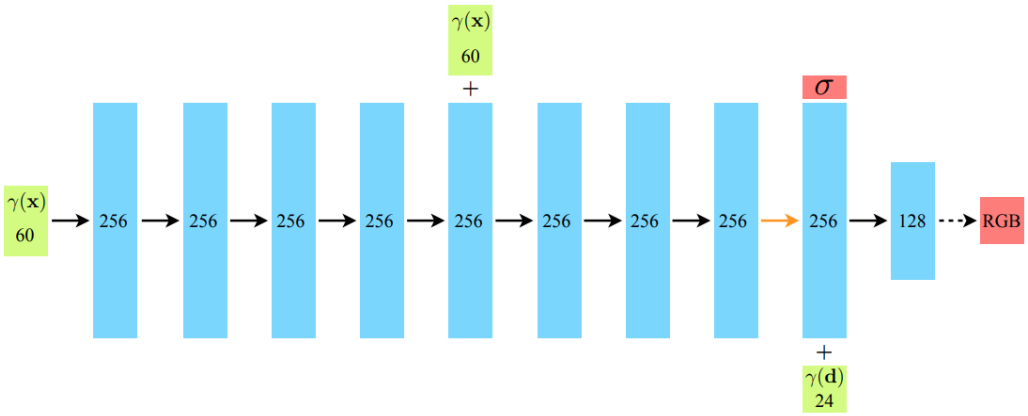


Figure 63. Fully-connected neural network architecture of NeRF

NVIDIA, todays main supplier of IA hardware and software, has developed a new technology called *Instant NeRF* which allows solving some of the problems of the original NeRF (Müller et al., 2022).

NVIDIA InstantNeRF is one of the most well-known and used improvement algorithms derived from NeRF, with the original NeRF architecture the training process takes several hours, with this new application it allows reducing the training process from hours to seconds. Instant NeRF is a real-time

3D representation technique, it achieves this through a novel neural network architecture that can handle the computations of neural radiance fields in real-time and optimized for a single GPU (*Instant Neural Graphics Primitives*, 2022/2023). Instant NeRF improvements are:

- Multiresolution hash encoding – it decreased the processing time (without quality loss), no Deep learning only 2 hidden layers
- Implicit Hash collision resolution
- Online adaptivity
- Neural Network
- Almost all the convergence happens in the first few seconds.

Instant NeRF workflow is the following:

1. Data Collection: Collect a set of 2D images of an object or scene from different viewpoints.
2. Feature Extraction: Extract features from the images to create a point cloud or depth map of the object or scene.
3. Neural Network Training: Train a neural network to predict the 3D geometry and appearance of the scene in real-time using the extracted features.
4. Reconstruction: Use the neural network to create a 3D model of the scene in real-time as new images are captured.
5. Rendering: Render the 3D model in real-time as the user interacts with it.

While NERF and Instant NERF are both powerful techniques for 3D representation, the possibility of implementing real-time 3D reconstruction by using Instant NERF and its ability to handle dynamic scenes makes it more suitable for the application of this thesis, the smoke plume 3D reconstruction (Condorelli et al., 2021).

Summary of 3D reconstruction softwares

The main 3D reconstruction software for applying 3D reconstruction have been compared in diverse papers (Schöning & Heidemann, 2015; de Balestrini & Guerra, 2011; Koutsoudis et al., 2014; Stathopoulou et al., 2019) the information has been summarized in Table 17.

Colamap and Meshroom are the most used softwares in academics for 3D reconstruction, they both can be also used for a commercial output and permit to create a mesh of the 3D object and have a free license, the main advantage of choosing Meshroom as the software used in this project is that there are available a lot of documentation and tutorials about it, it can be accelerated with GPU for a faster performance and user-friendly, Since I have the basic knowledge in this field this is the best software option considered. Agisoft is also a well-known software for this kind of application, but as it has a fee for the licence, from a starting point of 10€ per month having the less memory option, this option was discarded.

Table 17. 3D Softwares comparison for 3D reconstruction

Software	GPU supported	Price	Advantages	Disadvantages
COLMAP	Yes	Free	- Fast and efficient - Mesh output - Handles large amount of data	- Powerful PC required - No texture - No user-friendly
Meshroom	Yes	Free	- Fully documented - GPU accelerated - Highly automated - User-friendly - Lot of exporting options	- No large amount of data - Only photogrammetry inputs supported
Autodesk 123D catch	Yes	Free	- User-friendly - Available to mobile devices - Lot of exporting options - Cloud based (no powerful hardware)	- Struggle with complex geometry - Limited model scale - Internet connection needed
Agisoft Photoscan	Yes	Fee-based	- Intuitive - Extremely accurate - Handles large amount of data - Advanced features (for professional users)	- Expensive - Not open source - Powerful PC required
Visual SFM	Yes	Free for academic use	- Handles large amount of data - User-friendly	- Limited documentation - No actively maintained - Non-commercial - No user-friendly

Implementation of smoke plume 3D reconstruction

Initial considerations

A crucial point in 3D reconstruction is how the protocol for data acquisition. If this is important in most of the project involving images is a common cause of failure in 3D problems if the following points are not considered (*Instant Neural Graphics Primitives*, 2022/2023):

- Images must be taken with the same camera model.
- Images must be taken at the same time.
- When taking the images, cameras should be with the same imaging conditions, as much as possible.
- Rich variety of viewpoints of the smoke image to reconstruct is needed (the higher the number of images, the better will be the quality of the reconstructed images).
- Have a good coverage of the scene with the dataset: not contain mislabelled image data without being blurry (motion blur and defocus blur are both problematic).

3D reconstruction tests

We tested the basic reconstruction algorithm (InstantNeRF) on three different setups:

A. On static objects where many objects can be acquired without any change on the scene
B. In the Flames Lab in CERTEC institute where burning experiments are held. In this scenario only room reconstruction was performed and
C. In a wild scenario where one of the videos in the dataset was used extracting the frames to get different points of view.

The main difference between first two and the scenarios is that in the third one it would appear moving context. The number of points of views can be considered comparable. Also, the difference between the different images in the wild scenario is so small that we don't expect the method to work in a consistent way.

Once the software and technique for 3D reconstruction are compared and choose the best option available for this project, the first step is to obtain the images of the object or scene to reconstruct, the quality of this images is an important characteristic. The main goal is to have sharp images without motion blur and without depth blur.

A. Static object test

A first implementation of 3D reconstruction of a static object is done to qualify the reconstruction quality the traditional photogrammetry technique using Meshroom software versus the NERF architecture-based technique using Instant NERF API created by NVIDIA. To start implementing the 3D reconstruction algorithm, first a custom dataset must be created. In this case a dataset of 45 images taken around a school bag, being a basic object with some small details, the images were taken into account the guidelines on section Initial considerations.

Meshroom applies photogrammetry pipelines in its software, photogrammetry is the science and technology of extracting 3D information from 2D images. It involves the process of measurement of the objects and their environments through the use of imaging techniques. Meshroom uses a nodal system which consists on following pipeline steps in Figure 64, considering each step a node with it is own parameters.

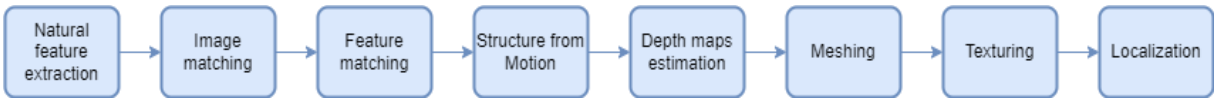


Figure 64. 3D reconstruction pipeline using Meshroom

Meshroom software is useful only with images, when taking a video of the object or scene to reconstruct, it has to be cut in various frames to obtain the single images, in this process, the orientation points of the camera are lost, so the software cannot identify them, in the other hand if

the dataset is taken directly by obtaining images of the object or scene, Meshroom software identifies the camera position of each image and is able to reconstruct it following the pipeline.

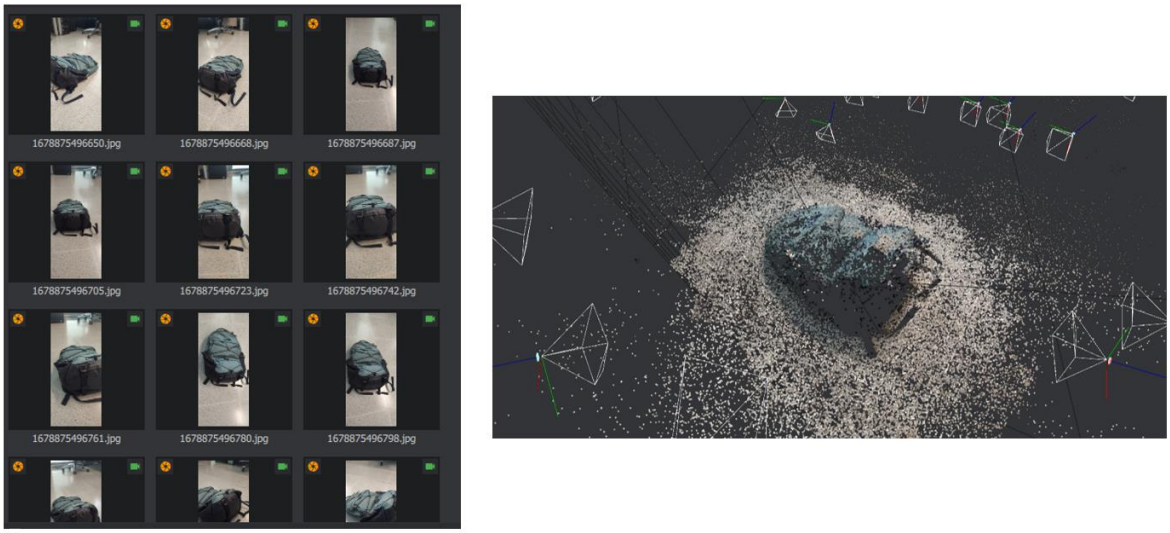


Figure 65. 3D static object reconstruction using Meshroom. Left: original input images. Right: 3D reconstruction

A posterior implementation of NeRF algorithm is done; it requires a photogrammetry reconstruction in order to obtain the camera positions before implementing the NeRF reconstruction methodology. The dataset is created following traditional photogrammetry pipeline based on SfM/MVS. The NERF pipeline accepts the input to be photos or videos (but video get blurry), when having a video as an input a pre-processing of it has to be done, in order to extract the frames from the video.

Structure from Motion (SfM) photogrammetry technique is used to determine the camera positions and viewing directions (x, y, z, Θ, Φ) before NeRF training is implemented by running the *colmap2nerf.py* script, this SfM application is done using COLMAP software following the best practices for photogrammetry explained in the previous section, to extract the necessary input camera data (feature matching computed as shown below).

```
python scripts/colmap2nerf.py --colmap_matcher exhaustive --run_colmap --aabb_scale 16 --images data/image_folder
```

The command above will run the *colmap2nerf.py* script which runs FFmpeg and COLMAP and outputs the conversion step in the required format to transforms.json file, where the camera positions of each image from the dataset created are detected using COLMAP. Performing the global bundle adjustment, technique used in photogrammetry that consists on refining the camera positions and the 3D structure of a scene based on a multi-view dataset (Zach, 2014).

The file structure, were the dataset and transforms file are stored, must be changed as following:

Figure 66. Data structure example for NERF 3D reconstruction. Left: file structure before generating transforms.json. Right: File structure after generating transforms.json (*Instant Neural Graphics Primitives*, 2022/2023).

To start the training process *instant-ngp.exe* must be executed with the path of the transforms file folder. According to the NVIDIA Instant NeRF documentation (*Instant Neural Graphics Primitives*, 2022/2023), the training process lasts about two minutes, the result obtained within two minutes is as good as it was trained during more time (there are not significant improvements after 2 mins). After the training process RGB values representing the emitted colours and the volume density (α) are obtained, this data is used to reconstruct in 3D the scene.

Figure 67. 3D reconstruction pipeline using Instant NERF

To proceed with the rendering, some more dependencies must be installed in the conda virtual environment:

pip install tqdm scipy pillow opencv-python

conda install -c conda-forge ffmpeg

The camera positions in order to make an output video of the obtained scene representation must be placed, afterwards rendered to have a visual output. Finally, this scene is converted into a 3D object (.obj) and this model file can be exported to the desired 3D mesh editing software, such ar Blender or MeshLab. In Figure 68 an example of the 3D static object reconstruction is shown, where the 3D reconstruction using the instant Nerf method is performed and its mesh visualized using Blender software.

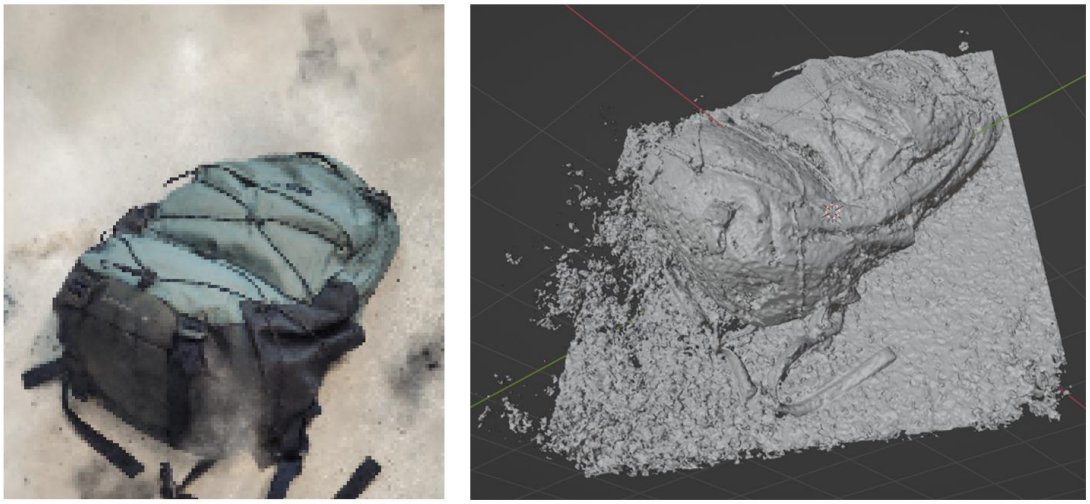


Figure 68. 3D static object reconstruction using Instant NERF. Left: 3D reconstruction. Right: Blender object of the 3D reconstruction

As its showed in this section, the 3D reconstruction applying Instant NERF algorithm is qualitative better than applying only photogrammetry technique with the Meshroom software. They both have been trained with the same dataset. From now on Instant Nerf architecture is the one used for the tests performed.

B. Flames lab test

The second test performed has the aim to reconstruct the scene where the controlled smoke tests will be performed in the future, the FlamesLAB, a laboratory of CERTEC in the UPC to develop tests on controlled fires and smoke plumes, using smoke booms. Some image of the dataset used to perform this test are shown in Figure 69.

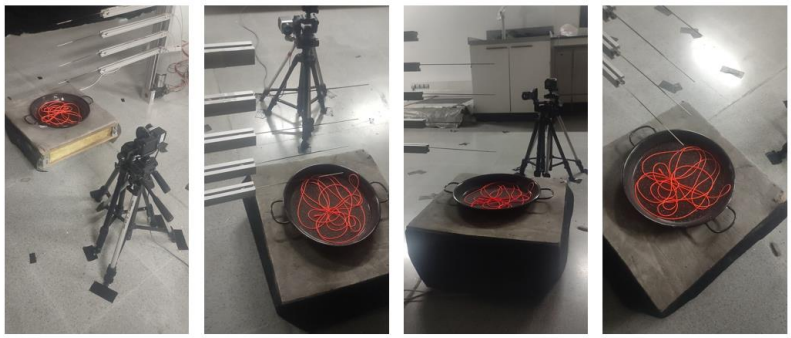


Figure 69. Image examples of dataset for flames lab test

The aim of this section is to extract as maximum as possible the details of the FlamesLAB set. A first test with 163 image frames extracted from a video taken around the set is done, where in Figure 70 are showed the output results.

Figure 70. 3D reconstruction using Instant Nerf with a dataset of 163 images.

Results shown in Figure 71, were in this test the thermocouples from the set can be appreciated in the 3D reconstruction, being this the most precise detail of the scene. The results from this test can be improved by incrementing the dataset images.

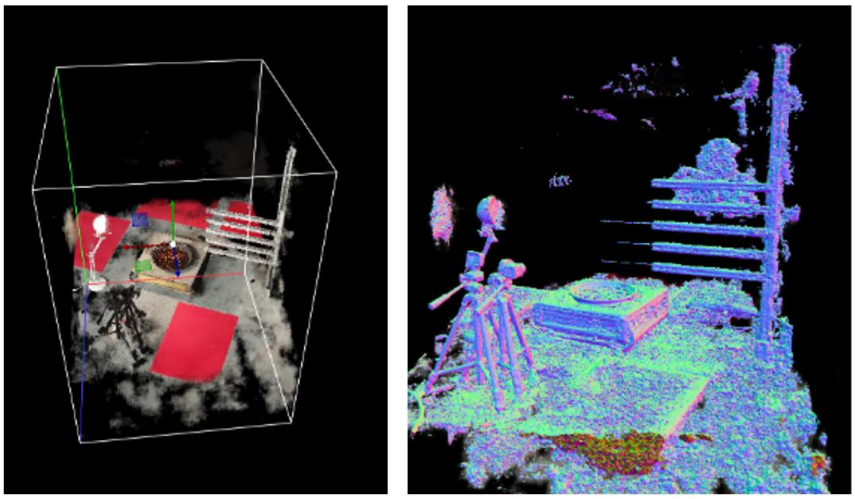


Figure 71. 3D reconstruction using Instant Nerf with a dataset of 615 images. Left: 3D reconstruction. Right: mesh of the 3D reconstruction.

C. Smoke scenario test

Another test was done using a dataset of 6 different viewpoints of a smoke plume test performed by the US Forest Service at Firelab in Missoula. The set-up layout is shown in Figure 72, with an example image frame of each viewpoint shown in Figure 73. The main problem of this test is that there are not enough viewpoints of the object to reconstruct (smoke), where any of the cameras have a superposed viewpoint between them, so the results obtained were not satisfactory, as shown in the Right side of Figure 72 shows the actual environment of this test.

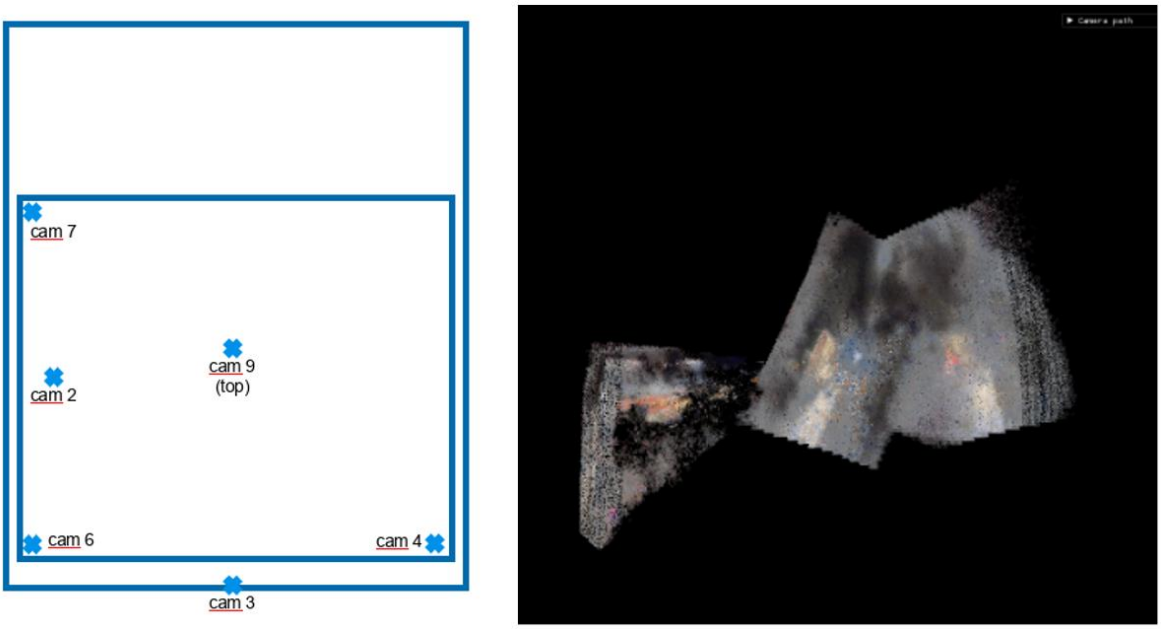


Figure 72. Smoke scenario test. Left: Cameras layout of 3D reconstruction test data acquisition. Right: output reconstruction using Instant NeRF.

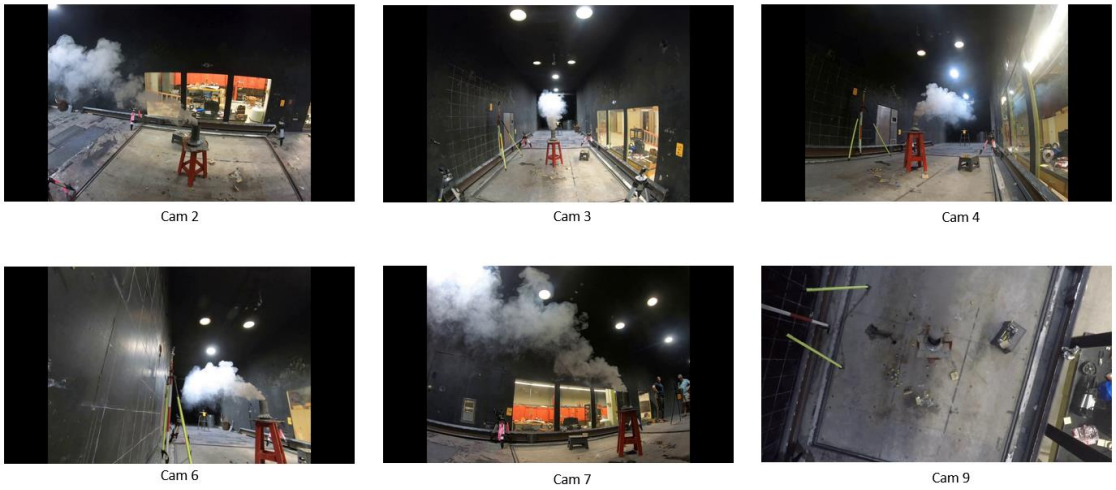


Figure 73. Examples of each viewpoint of the 3D reconstruction test