**Automated Polyps Detection**

Deep Learning 2023-25268-T1

06-06-2024

[**Introduction 2**](#_1eahp8ywiu1c)

[**Pre-trained models 3**](#_3v1y5ypmmu9k)

[VGG16 3](#_ylzhy60cyue)

[Inception-V3 4](#_fzxi30fci630)

[ResNet50 5](#_pxn35o43mr63)

[**Code Implementation 7**](#_rws0c369z5gm)

[**Results Visualization 8**](#_m1e8an9g5icx)

[VGG16 8](#_sa55261fp7pa)

[Inception-V3 9](#_82pi8erksfxh)

[ResNet50 10](#_uddca0np7ug0)

[**PROPOSALS 11**](#_idocpey3z20u)

[Data Augmentation (DA) 11](#_33seiy6gp9jb)

[Results DA 11](#_ji7wo9vf01gj)

[VGG16 11](#_db88dqopf2th)

[InceptionV3 12](#_aus7dfon7h2y)

[ResNet50 13](#_an67bsho9mln)

[Fine-tuning (FT) 14](#_e8v4l7v4v5fm)

[Results FT 14](#_aiyykwanb16c)

[VGG16 14](#_9g1p5om5rpc0)

[InceptionV3 15](#_brz8kx9v2y0d)

[ResNet50 15](#_ydovhnj2n0dz)

[**CVC-ClinicDB 16**](#_hoa5wqhigf8x)

[**Results new Dataset 16**](#_6fk3v0py5tax)

[InceptionV3 16](#_5jcy0i3m499)

[**Conclusion 17**](#_5aij4yfco0kw)

[**Bibliography 18**](#_k6zex6e7ozyd)

# **Introduction**

Colorectal cancer (CRC) is one of the most critical health issues globally, classified as one of the most common cancers and the second leading cause of cancer-related deaths. Its incidence rates are steadily increasing each year. CRC originates from adenomatous lesions or polyps in the colon, which must be identified and removed to prevent the disease. The main method for detecting these polyps is colonoscopy; however, due to the complexity of colon anatomy and potential human error, approximately 25% of colorectal polyps are missed during colonoscopy. This high error rate highlights the need for a more reliable secondary observer.

Artificial Intelligence (AI) offers a promising solution to this problem. Deep learning techniques can enhance the accuracy of polyp detection in colonoscopy by serving as an additional observer, thus reducing the likelihood of missed polyps and aiding in the prevention of colorectal cancer. This project focuses on developing a computational method for the automatic detection of colorectal polyps using deep learning techniques.

For this purpose, different datasets have been used. One is called Kvasir-SEG, and the other was extracted from a GitHub repository intended to detect polyps using the CVC-ClinicDB dataset. The Kvasir-SEG dataset includes 1000 polyp images and their corresponding masks, while the CVC-ClinicDB dataset, which has been preprocessed, separates the images into two folders: polyps and non-polyps, used to train the model. This latter dataset is an open-access collection of 612 images with a resolution of 384×288 from 31 colonoscopy sequences. Leveraging both datasets, which include images of gastrointestinal polyps and their corresponding masks, pre-trained models like VGG16, InceptionV3, and ResNet50 have been adapted to accurately identify polyps in the images.

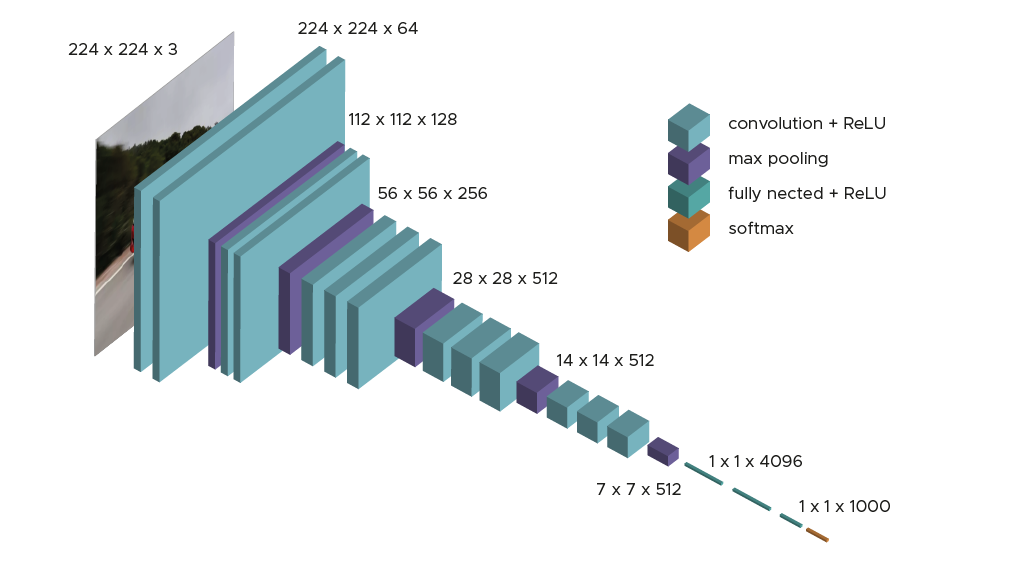
The primary objective of the project is to create an automatic polyp detection system that can assist in real-time during colonoscopies. By using deep learning models, we aim to improve accuracy and precision, thus reducing the incidence of cancers caused by overlooked polyps. Both datasets provide a solid foundation for training these models, ensuring they can generalize well across different types of polyps and variations in colonoscopy procedures.

# 

# **Pre-trained models**

## VGG16

VGG16 is a convolutional neural network proposed by K. Simonyan and A. Zisserman, from Oxford university. This model achieved a precision of 92.7% in ImageNet, that is one of the bigger punctuations achieved. This represents an improvement over previous models by proposing smaller convolution cores (3x3) in the convolution layers of what had been done previously. Regarding the architecture the model consists of 16 layers with weights distributed as follows:



1. **Convolutional Layers:** The model begins with a series of convolutional layers. There are 13 convolutional layers divided into 6 blocks; each block contains 2 or 3 convolutional layers with small filters (of 3x3 pixels) that help capture fine details in the images while preserving spatial resolution.
2. **Activation Function:** After each convolutional layer, a Rectified Linear Unit (ReLU) activation function is applied. ReLU introduces non-linearity to the model, allowing it to learn complex patterns.
3. **Pooling Layers:** After the blocks of convolutional layers a max-pooling layer with a 2x2 filter and a stride of 2 is applied. This helps in reducing the spatial dimensions of the feature maps, thus down-sampling the image and reducing the computational load.
4. **Fully Connected Layer:** The convolutional and pooling layers are followed by three fully connected layers. Each dense layer is also followed by a ReLU activation function.
5. **Softmax Layer:** The final layer is a softmax classifier that outputs a probability distribution over the 1000 classes for classification tasks.

VGG16 offers several advantages such as the straightforward architecture that is easy to understand and implement, the high performance in image classification benchmarks, and effective transferability due to its pre-trained weights on large datasets. However, this model has some limitations such as being computationally intense and needing high memory usage.

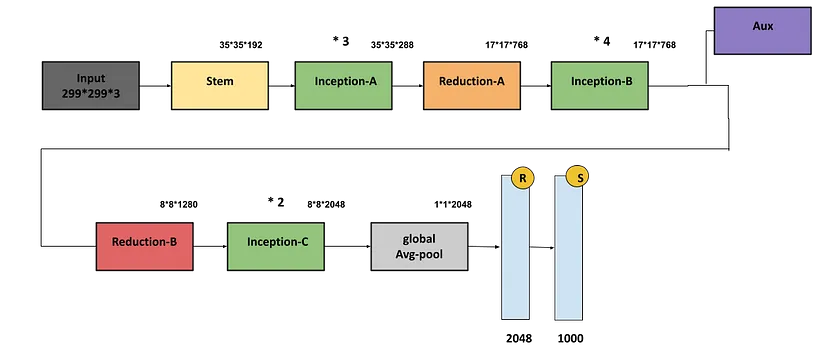
Among this project, VGG16 is being adapted and fine-tuned on both datasets to detect polyps in colonoscopy images. By leveraging its deep architecture and pre-trained weights, VGG16 should be able to effectively learn to identify the polyps.

## 

## Inception-V3

Inception-V3 is an image recognition model that in many cases is able to achieve an accuracy of around 78.1% in the ImageNet dataset. Is a Convolutional Neural Network Network architecture introduced by Google as part of the Inception family of networks.

The model consists of symmetric and asymmetric compilation blocks including convolution, average reduction, maximum reduction, concatenations, withdrawals and fully connected layers. The Inception V3 model consists of a series of convolutional layers and specialized inception modules.



Its architecture is built on the idea of using multiple convolutional filters of different sizes in parallel that are concatenated into a single output. This fancy approach allows the network to capture features at various scales at the same time. The architecture includes many modules that factorize bigger convolutions into smaller convolutions (7x7 is factored into 3x3), this idea reduces the number of parameters and the computational load while maintaining the ability to learn complex features. Inception V3 also includes techniques like the batch normalization, label smoothing and some auxiliary classifiers that help to improve the training of the model.

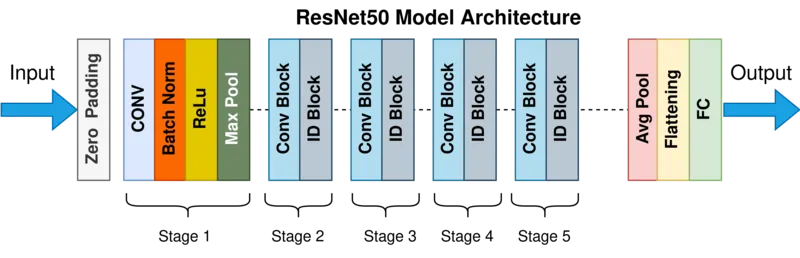
Taking a deeper view to the architecture can be observed that the model begins with a series of convolutions and pooling layers whose main function is to reduce the spatial dimensions of a given image. Following, the core of the network consists of many Inception modules (Inception-A, Reduction-A, Inception-B, Reduction-B, Inception-C) that further process the feature maps. All these modules are organized into blocks, with reductions in spatial dimensions between blocks that are used to manage the computational load effectively. The final layers of the network architecture include average pooling, dropout, a fully-connected layer and a softmax classifier.

This model has some advantages such as the ability to achieve high accuracy on large-scale image classification benchmarks with relatively model computational resources.. The main reason why it is particularly effective is because of transfer learning, as its pre-trained weights can be adapted to many different tasks. However, it is noteworthy that the complexity of the network can also be a limitation; requiring careful tuning of hyperparameters and significant computational power for training.

Finally, as a conclusion, it can clearly be stated that Inception V3 represents a balanced trade-off between computational efficiency and model complexity, making it a versatile choice for various computer vision applications.

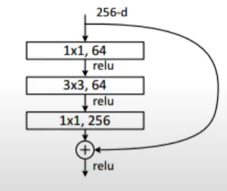
## ResNet50

ResNet50 is a deep CNN architecture, a variant of the popular ResNet architecture together with “50” that stands for the number of layers, developed in 2015 by Microsoft Research. It is a powerful image classification model that is trained on large datasets and it is able to achieve optimal results. This model introduces the revolutionary concept of the residual connections to be able to address the challenges of training deep networks.



The architecture of the model can be divided into four main parts that are convolutional layers, identity blocks and convolutional blocks and fully connected layers.

1. **Convolutional Layers:** These are the initial layers followed by batch normalization and ReLU activation. The main goal of these layers is to extract the basic features, such as edges, textures, shapes…, from the input images. Convolutional layers are also followed by max-pooling layers that reduce the spatial dimensions of the feature maps and preserve the most important features.
2. **Residual Blocks:** These are the core of the model, and can be splitted into two types (following the structure of the picture above):
   1. **Identity blocks:** that perform a series of convolutions on the input, check if the result fits with the original input image and then add the improvements to the input to get the output. This operation, kwon as skip connection, allows the network to learn residual functions. This helps to address the vanishing gradient problem enabling the training of much deeper networks.
   2. **Convolutional Blocks:** These blocks are similar to identity blocks but these ones have an additional 1x1 convolutional layer to adjust the dimensions of the input before it is added to the output. This allows the block to change the number of filtering making sure the dimension matches.



1. **Fully Connected:** After passing through the residual blocks, the network includes fully connected layers that aggregate the high-level features learned by the convolutional layers. The final fully connected layer is followed by a softmax activation function.

ResNet50 is a model that offers high performance achieving good results on image classification benchmarks. Its deep learning capabilities are enhanced by residual connections, which enable the network to learn complex representations without training difficulties. Additionally, its pre-trained weights make it highly effective for transfer learning across various tasks and datasets. However, ResNet50 requires significant computational resources, including powerful GPUs and substantial memory, making it challenging for real-time processing or deployment on devices with limited resources. Furthermore, its complex architecture necessitates careful tuning of hyperparameters.

# **Code Implementation**

In this section will be explained the code implementation of the previously explained models, as well as some issues found during the development and a proposal of solutions. All models have the same implementation structure.

First of all, dataset loading is done using a function and the data split for training and test subsets.

The model is adapted to our data through transfer learning, the pre-trained model is charged except for the last fully connected layers, which are the ones modified to train the dataset. The implemented layers are *Con2D* and *UpSampling2D*. Convolutional layers learn to detect patterns in the images whereas the up sampling layers increase the size of the image after each set of convolutional layers. The las *Conv2D* layer produces the final output image using a single channel and a sigmoid activation function, to be able to get values between 0 and 1, this represents the probability of each pixel being part of the object was wanted to segment. It was also ensured that the output image of the model has the same size as the input image.

Once inputs and outputs are defined, the model is completed using an Adam optimizer and a cross entropy loss. To train the model the function fit is used together with the data loaded and defining the epochs to be 50 and the batch size to 8.

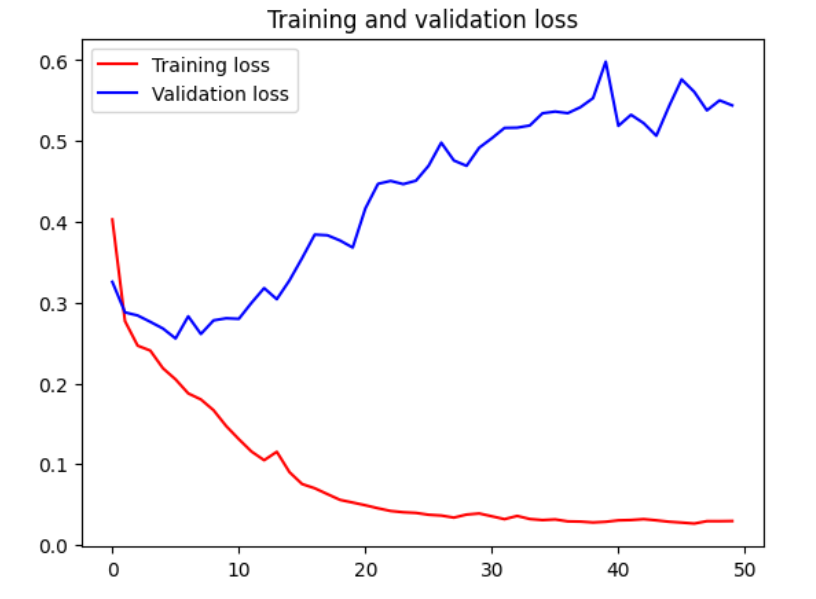
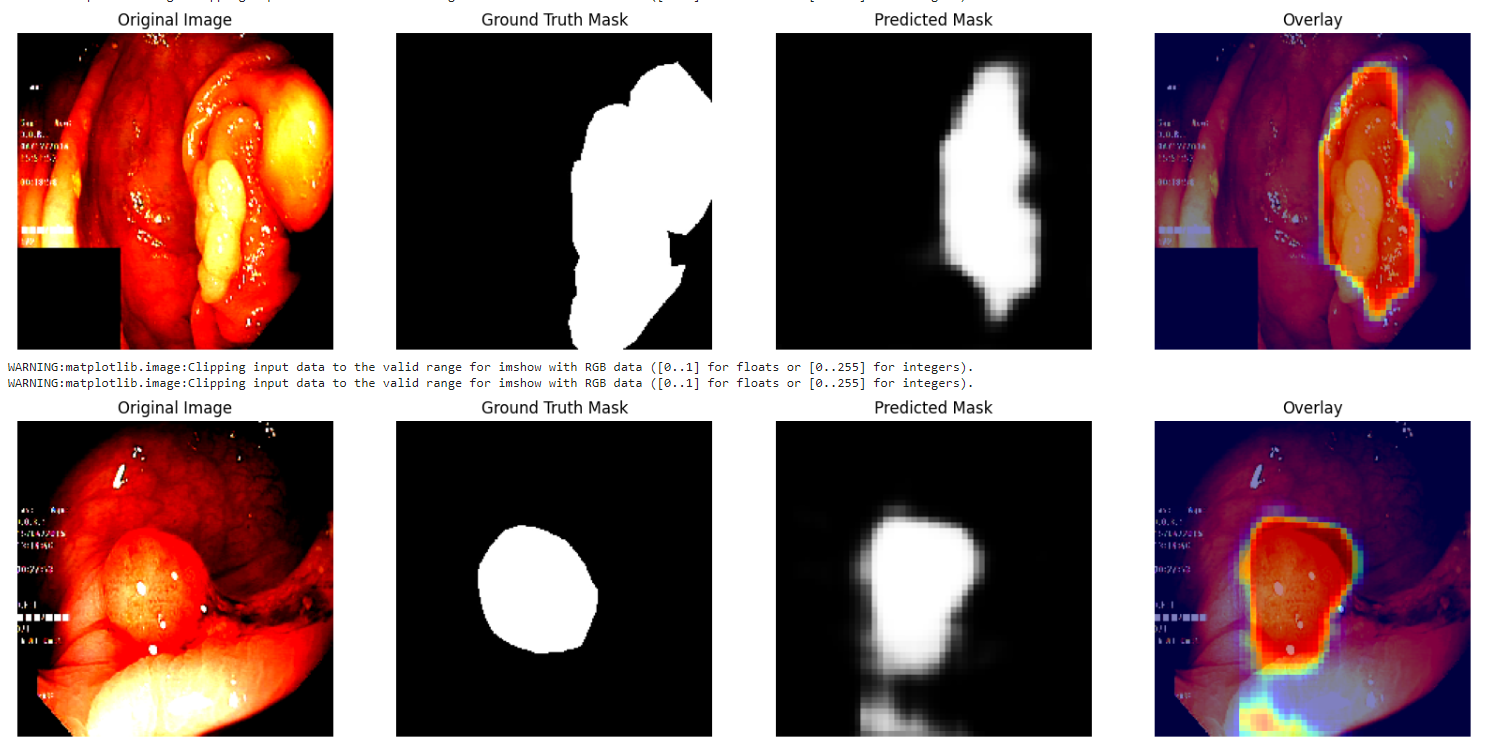
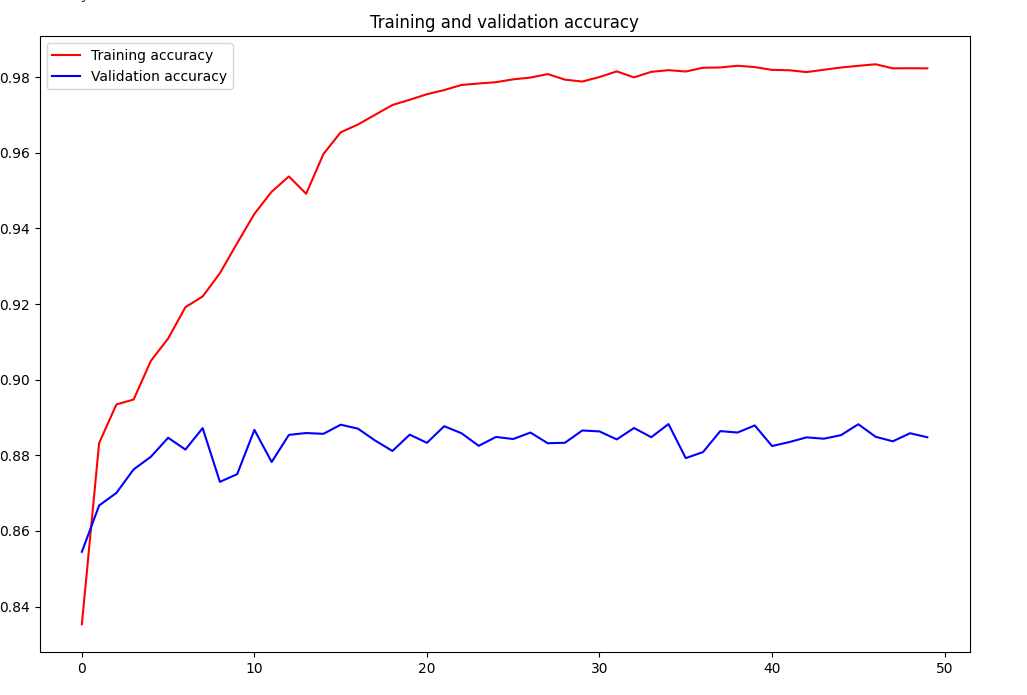
Finally, having the results of the model execution, to visualize them, some auxiliary functions have been created: visualize\_results(), plot\_history().

visualize\_results() makes a plot of 4 images for each image in the dataset: the original image, its respective mask, the predicted mask and the mask behind the original image. plot\_history() creates two plots to do a comparison between train and test data subsets: one showing the accuracy and, the other one, the loss.

# **Results Visualization**

## VGG16

* Accuracy
* Loss



## 

## 

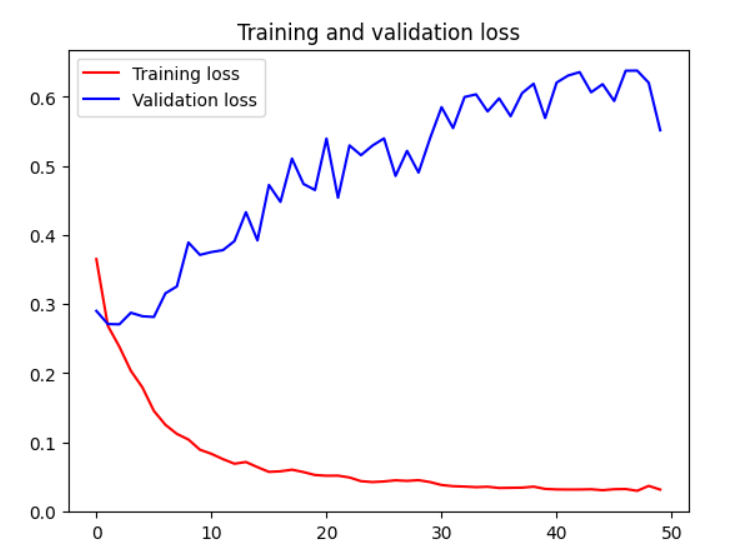
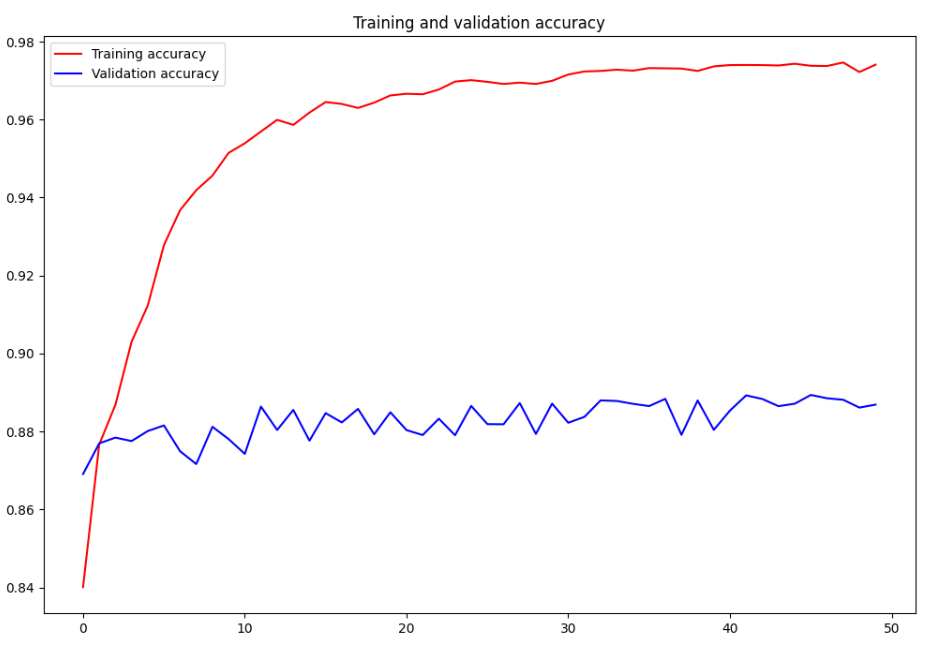
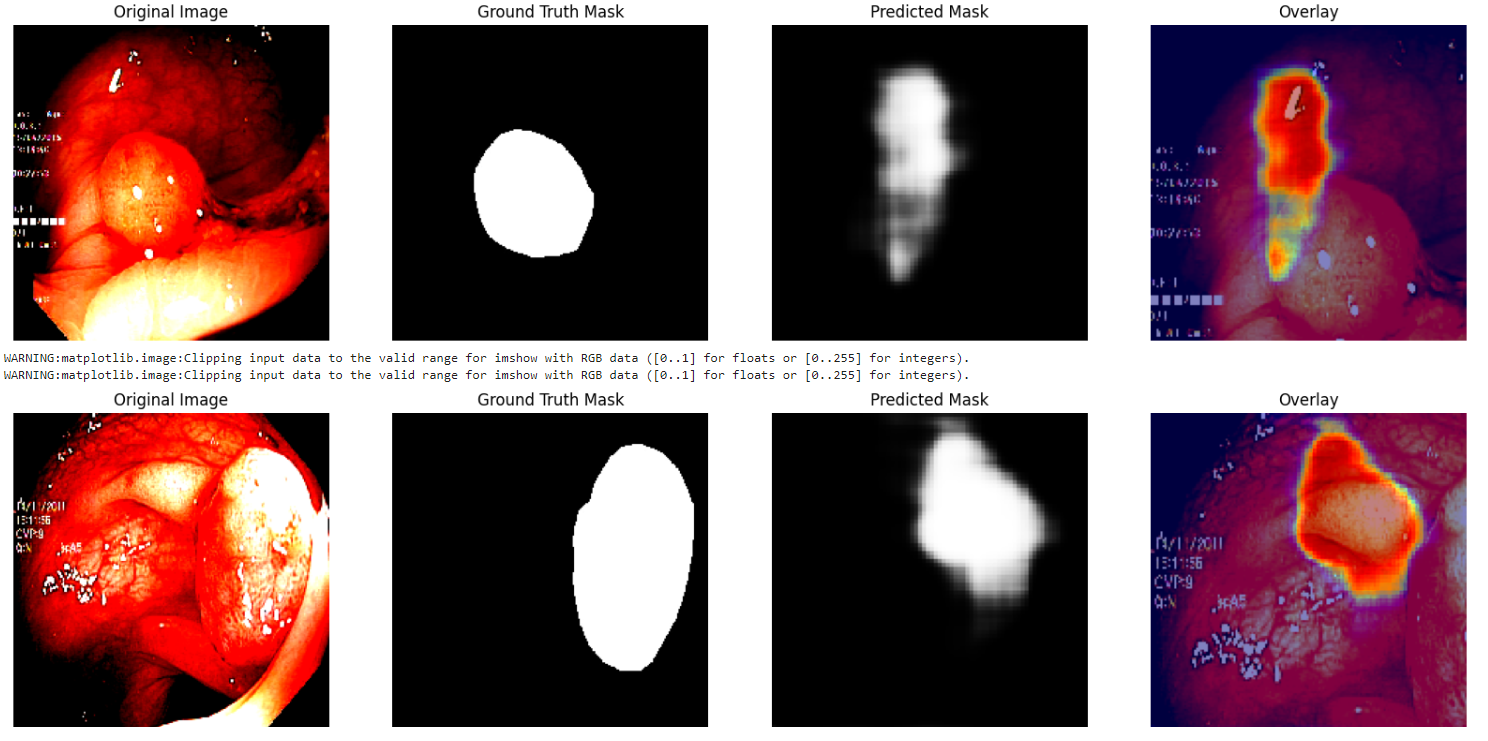
## 

## 

## 

## Inception-V3

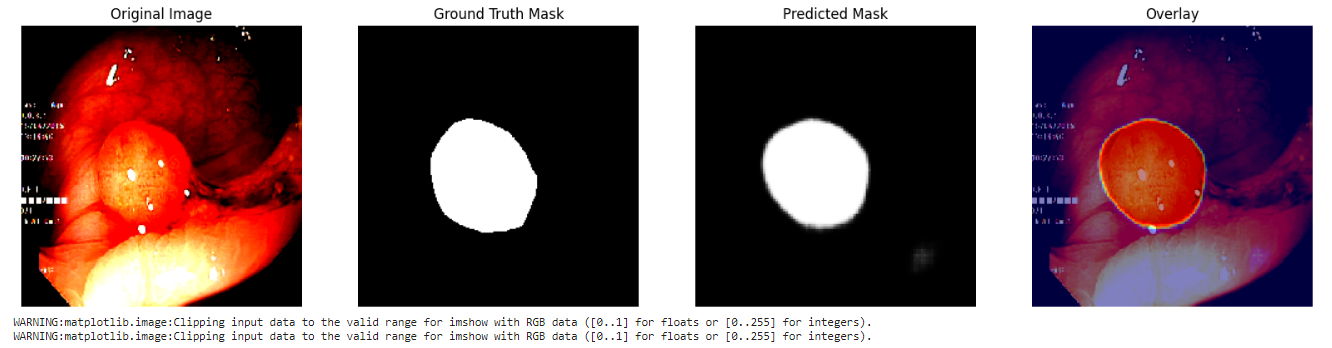
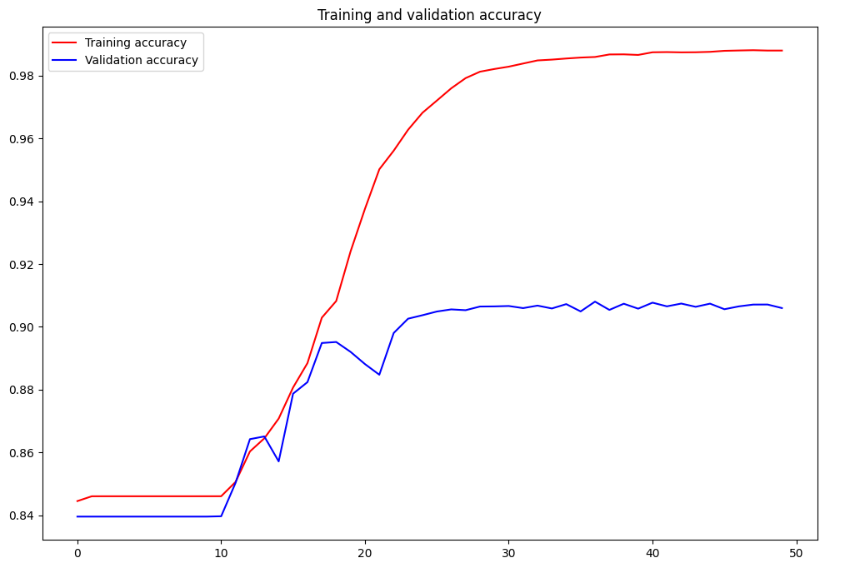
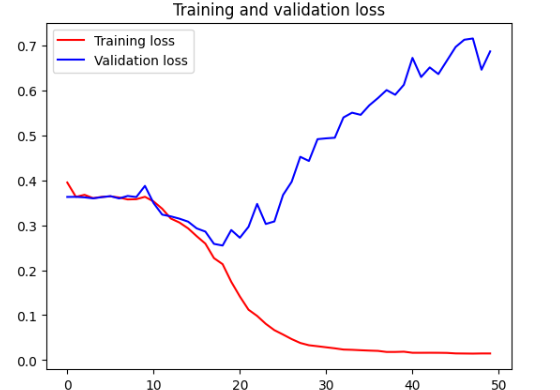
* Accuracy
* Loss

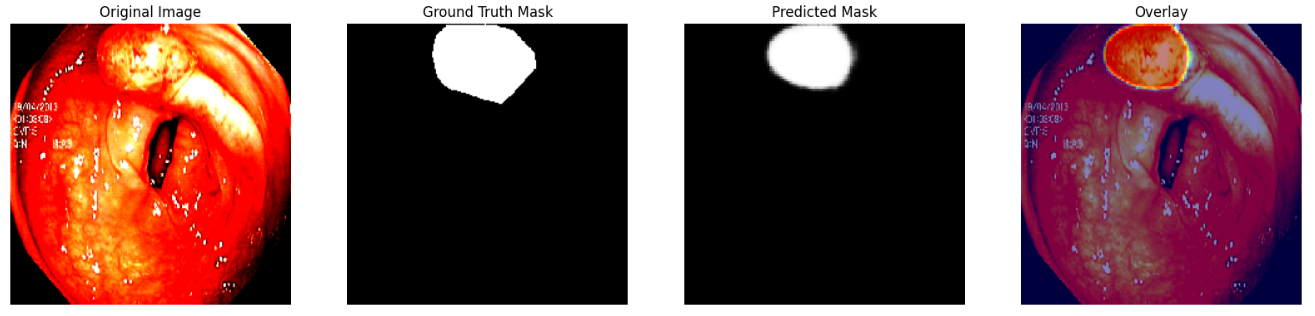


## 

## ResNet50

* Accuracy
* Loss





Results obtained from all the models are not the best ones. The tendency of the validation line in the loss is to go in the opposite direction that the training one does, probably representing overfitting. And for this reason the image visualization shows an almost perfect estimation.

For this reason, some possible solutions for this problem have been proposed: data augmentation, fine-tuning and using different datasets.

# **PROPOSALS**

## Data Augmentation (DA)

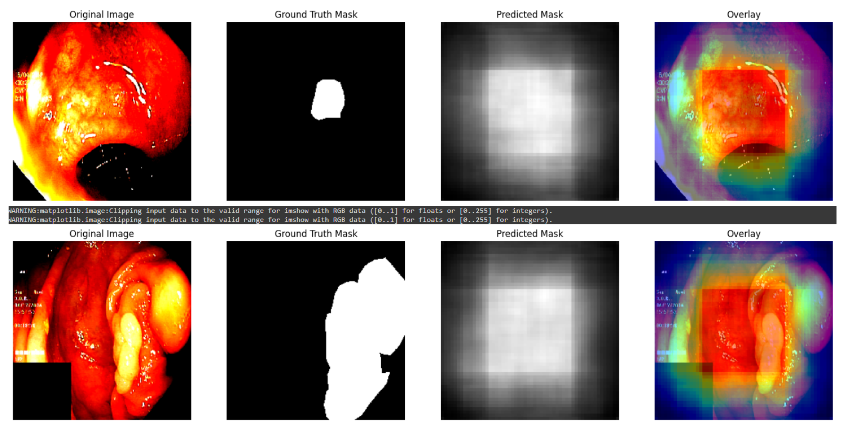
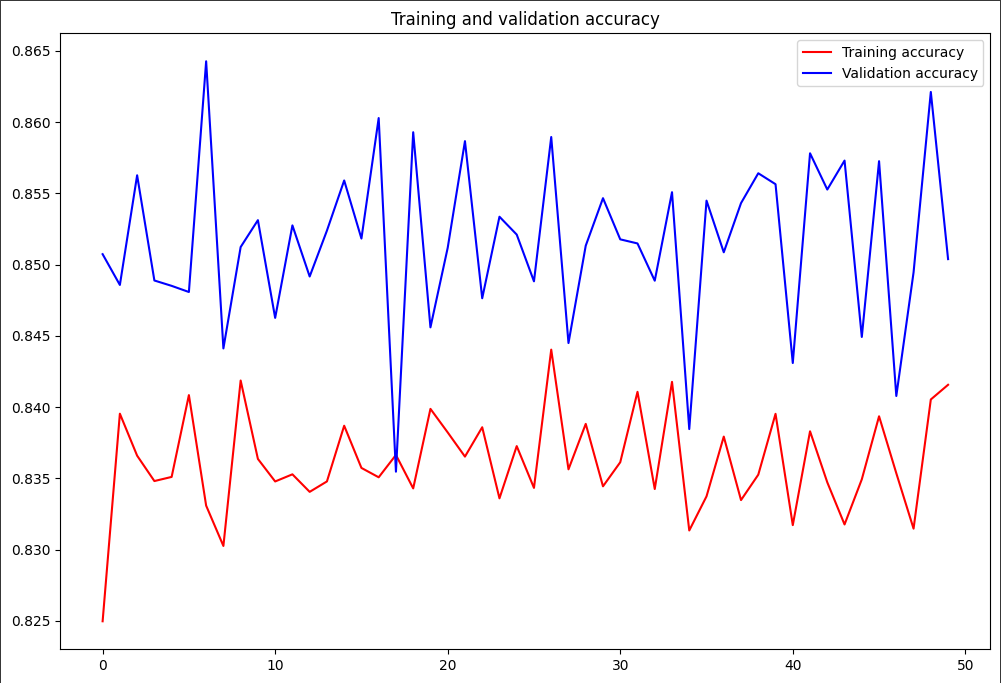
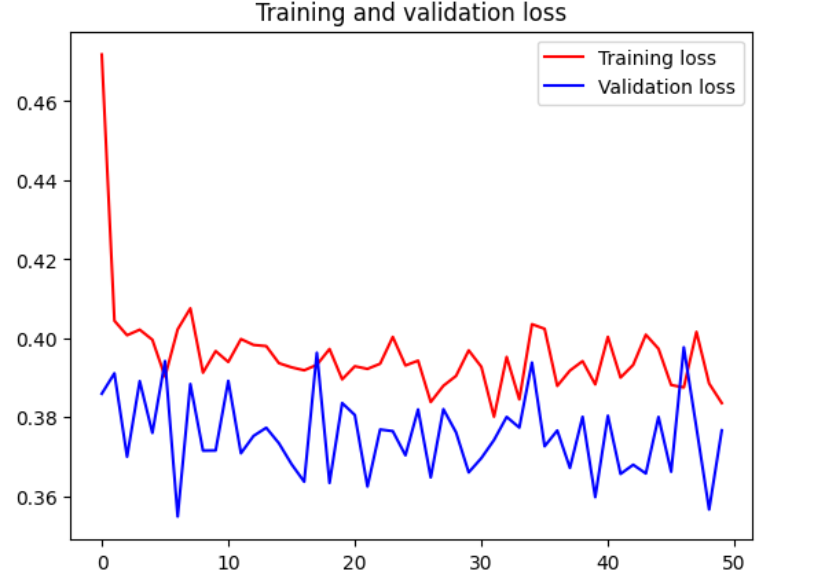
Data augmentation is a technique used to increase the diversity of a dataset by applying various transformations to the existing data. This helps improve the performance and robustness of machine learning models, particularly when the original dataset is limited in size. Common data augmentation techniques include:

* Rotation: Rotating the image by a certain angle.
* Width and Height Shifts: Shifting the image horizontally or vertically.
* Shear: Applying a shear transformation to the image.
* Zoom: Zooming in or out on the image.
* Horizontal Flip: Flipping the image horizontally.
* Fill Mode: Filling in new pixels created by transformations.

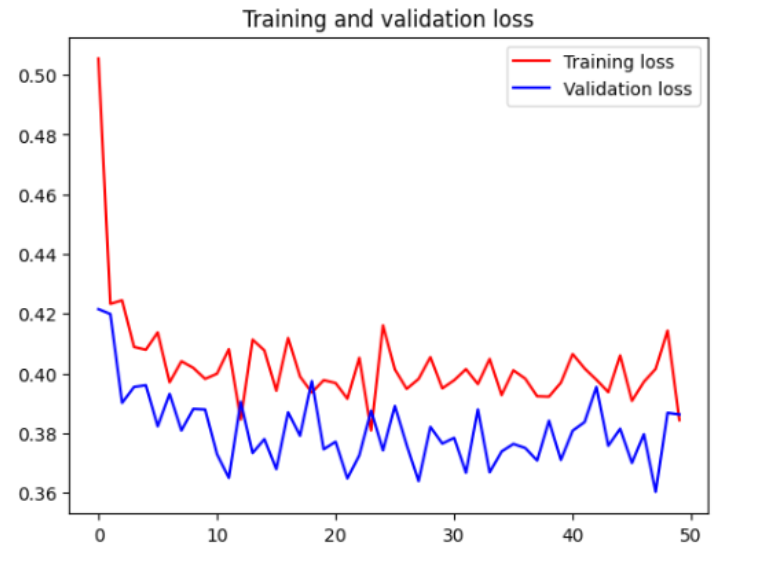
Data augmentation helps the model generalize better by exposing it to a wider variety of variations in the training data.

### Results DA

#### VGG16

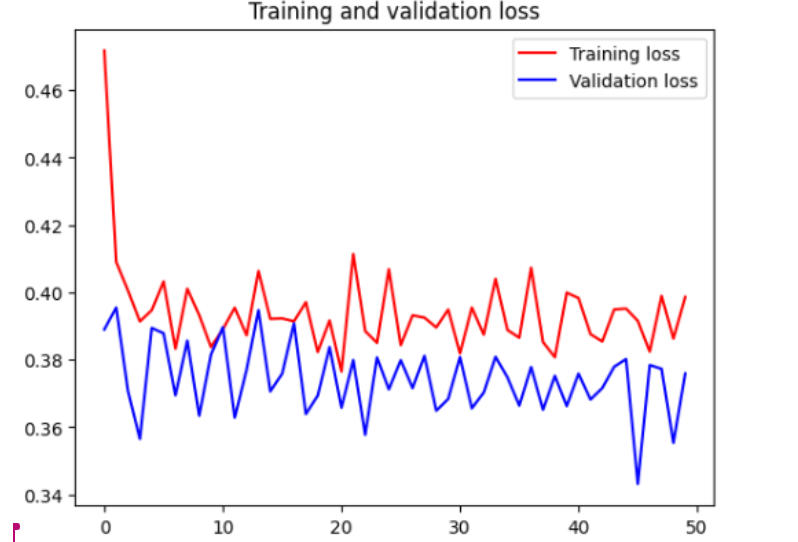
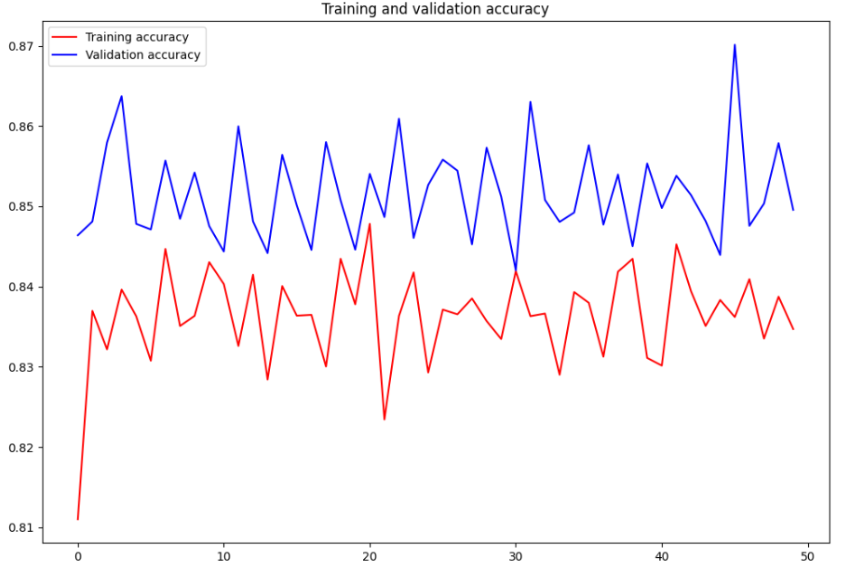
* Accuracy
* Loss

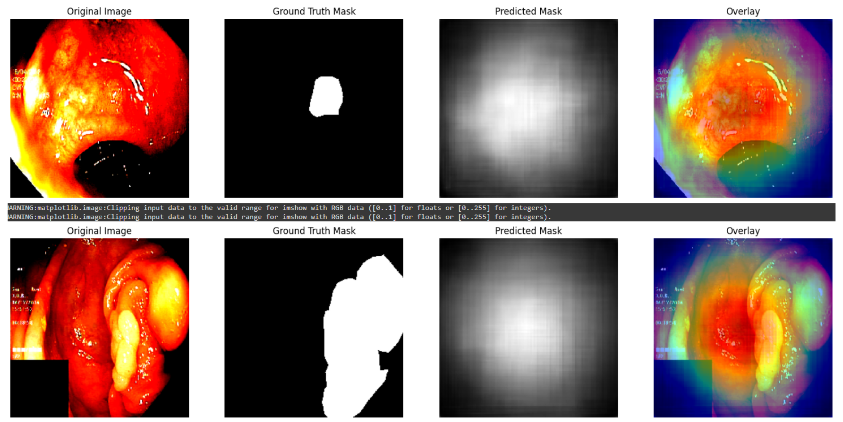
#### InceptionV3

* Accuracy
* Loss

#### 

#### ResNet50

* Accuracy
* Loss



## 

## Fine-tuning (FT)

Fine-tuning is a process used in transfer learning where a pre-trained neural network model is further trained on a new, often smaller, dataset to adapt it to a specific task.

* **Freeze Initial Layers:** Initially, freeze the weights of the early layers of the model. These layers typically capture general features (e.g., edges, textures) that are useful across various tasks.

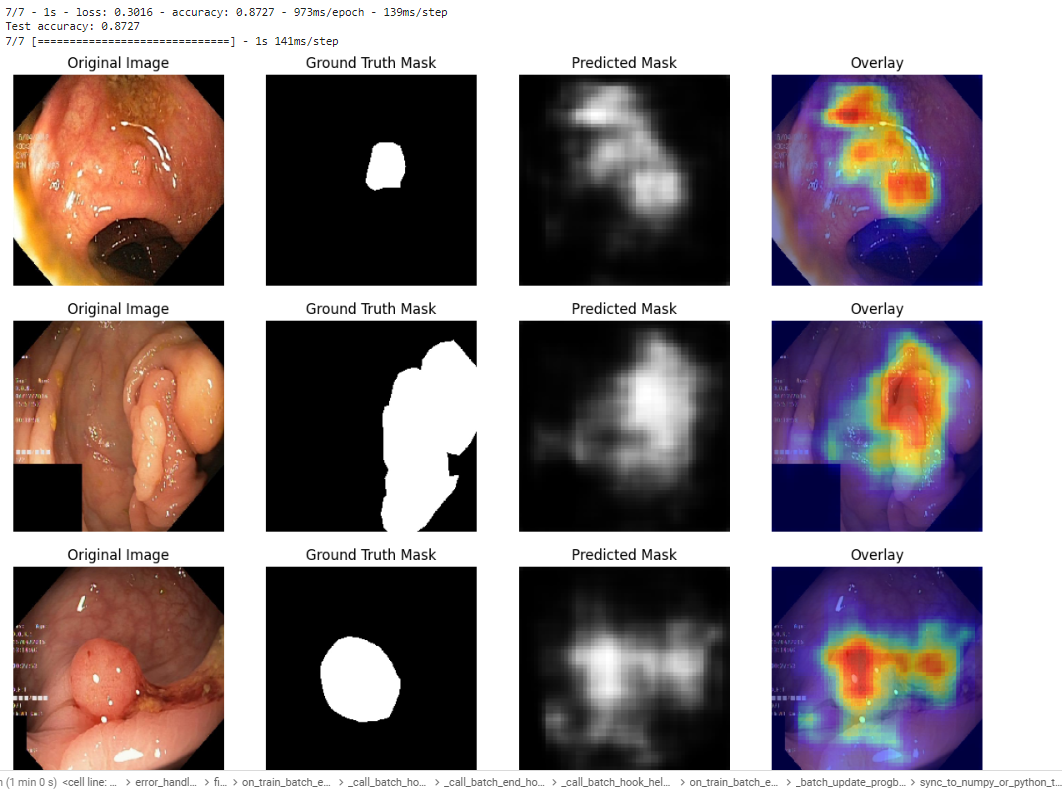
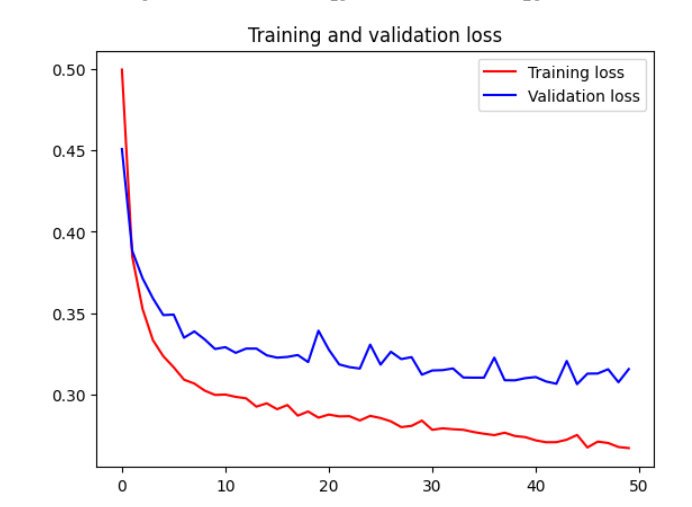
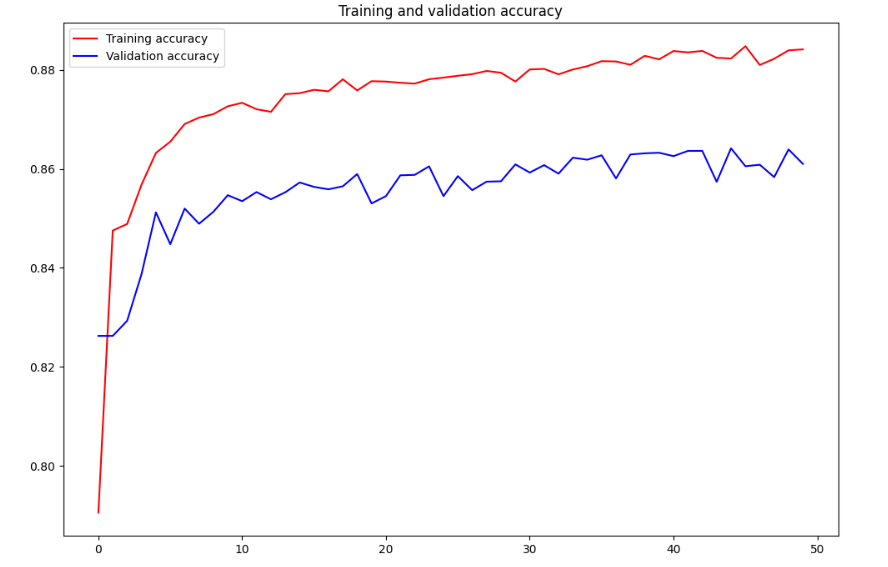
Freezing is done to retain the learned features and prevent them from being altered during subsequent training.

* **Replace or Add New Layers:** Add new layers on top of the pre-trained model to adapt it to the new task. This could involve adding fully connected layers, convolutional layers, and, for segmentation tasks, upsampling or transposed convolutional layers.
* **Train the New Layers:** Train the model with the new layers while keeping the pre-trained layers frozen. This helps in adapting the model to the new dataset without altering the previously learned features.
* **Unfreeze Some Layers and Fine-Tune:** Unfreeze some of the later layers of the pre-trained model. These layers are more specialized and can be fine-tuned to better fit the specific characteristics of the new dataset.

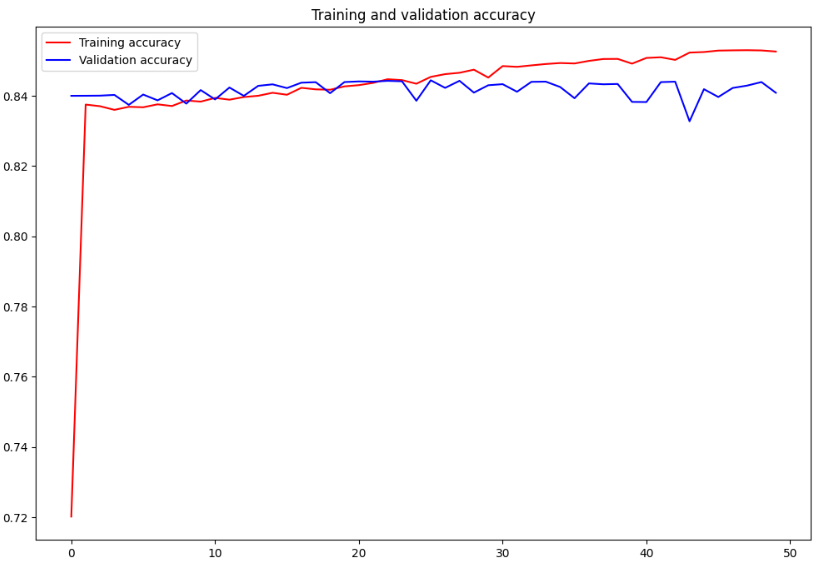
### Results FT

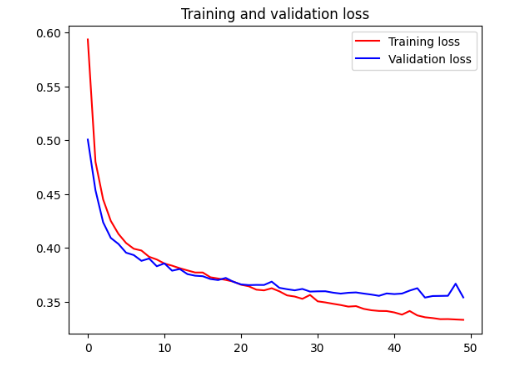
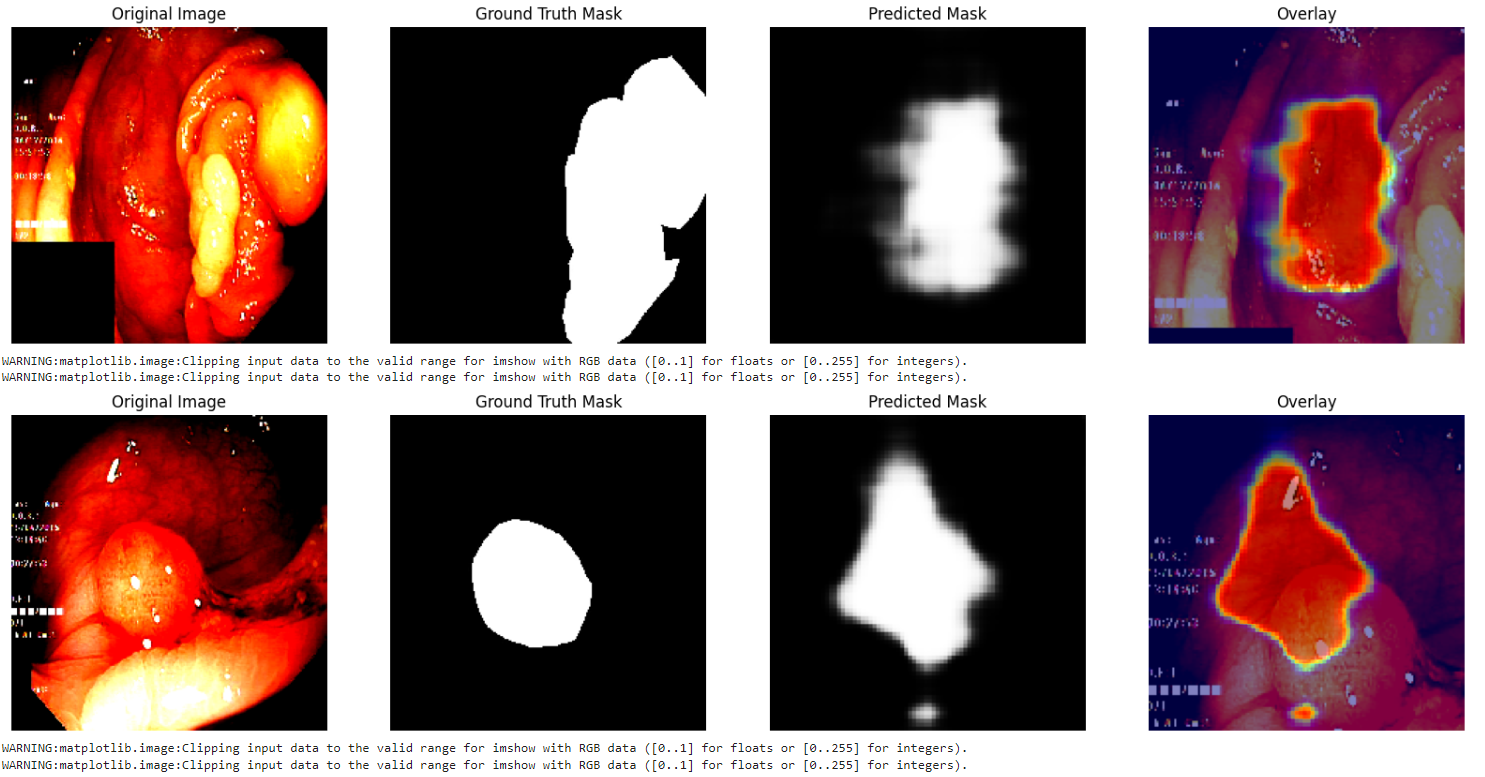
#### VGG16

* Accuracy
* Loss



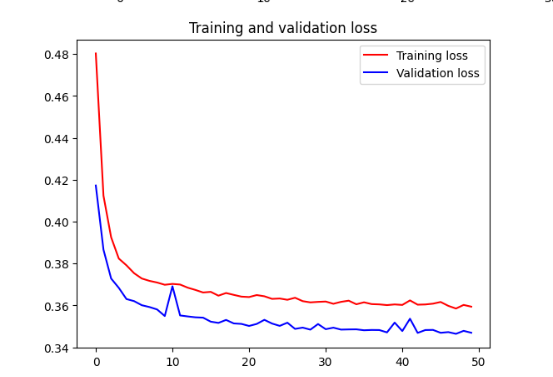
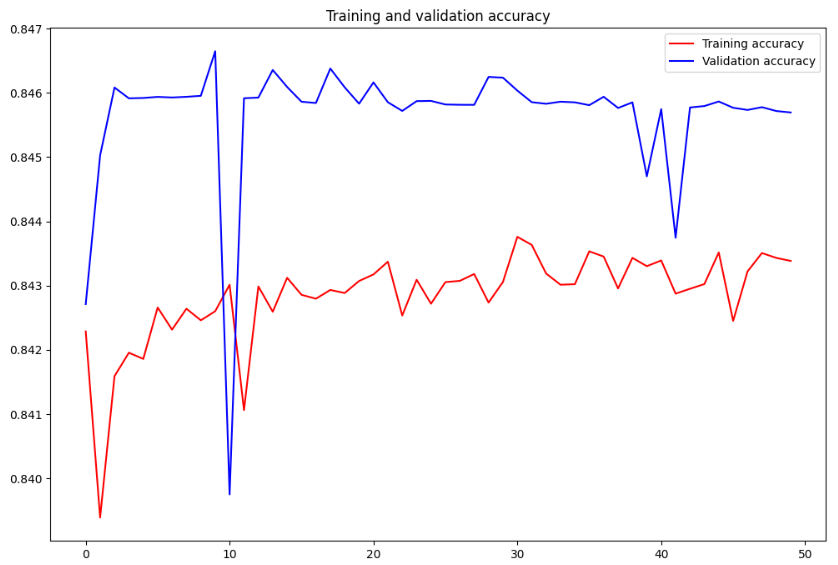
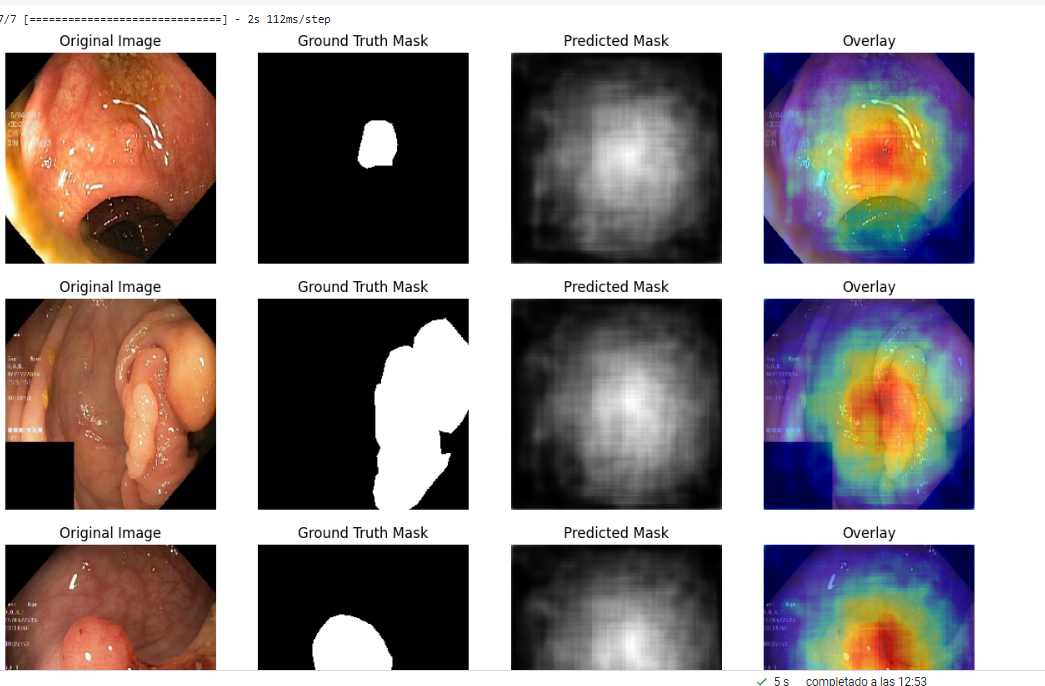
#### InceptionV3

* Accuracy
* Loss



#### ResNet50

* Accuracy
* Loss



# CVC-ClinicDB

As another proposal for improving result a different dataset, CVC-ClinicDB, has been tried, taken from a GitHub repository where it was already preprocessed [3].

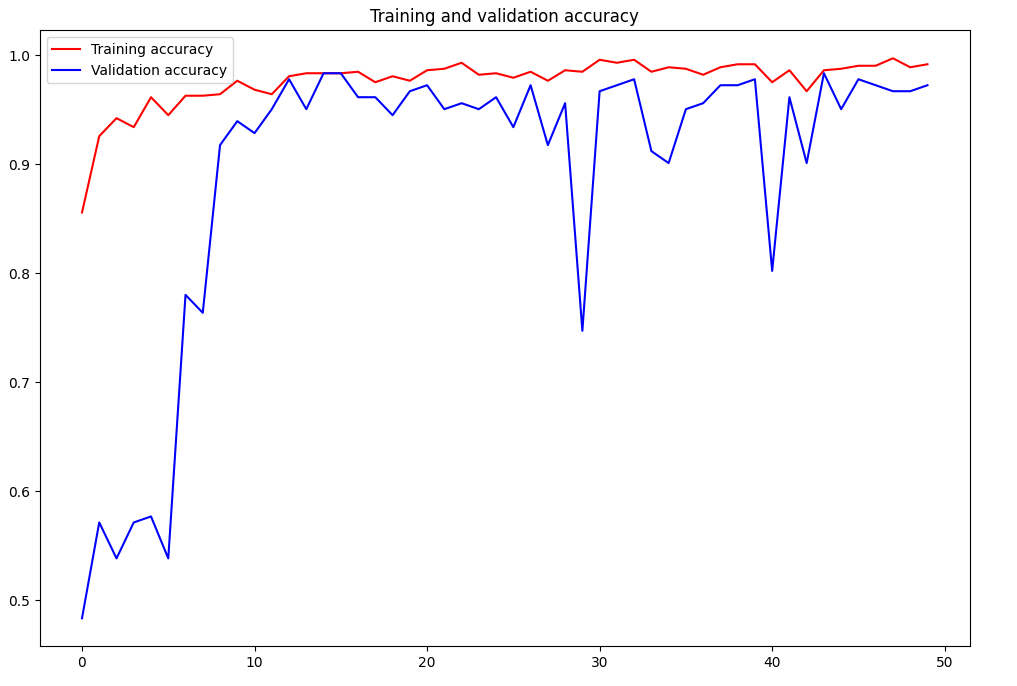
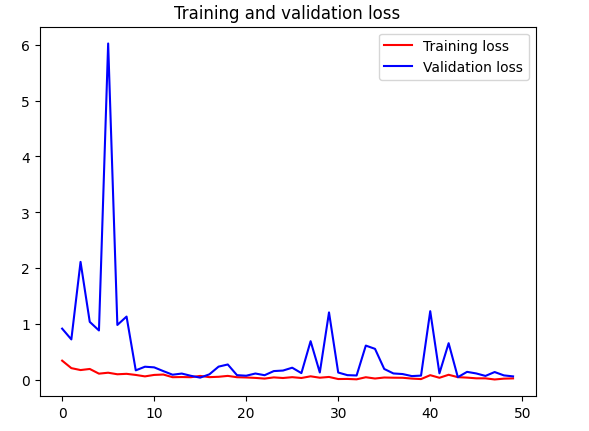
The preprocessing of the dataset involves creating two sets of images: one containing polyps and another containing non-polyps. This process ensures that the model can distinguish between images with and without polyps effectively.  
For that end, the black margins from both original and ground truth images are removed firts. Then, identify and crop the polyp region from the original image using the ground truth mask. And to extract non-polyp sections, areas around the polyp are selected.

# Results new Dataset

Note: Only InceptionV3 model result will be added into the report

#### InceptionV3

* Accuracy
* Loss



# 

# 

# **Conclusion**

Our project underscored the importance of addressing overfitting in deep learning models, especially in the context of medical image analysis. While our initial attempts with pre-trained models and standard techniques did not achieve the desired results, the insights gained will guide future efforts in model selection, data handling, and regularization strategies. The journey through this project highlighted the iterative nature of machine learning and the need for continuous experimentation and adaptation to achieve optimal results.

# Bibliography

[1] World Health Organization: WHO & World Health Organization: WHO. (2023, July 11). *Colorectal cancer*. <https://www.who.int/news-room/fact-sheets/detail/colorectal-cancer>

[2] *Papers with Code - CVC-ClinicDB Dataset*. (n.d.). <https://paperswithcode.com/dataset/cvc-clinicdb>

[3] *muntisa/Colonoscopy-polyps-detection-with-CNNs: Colonoscopy polyps detection with CNNs*. (n.d.). GitHub. <https://github.com/muntisa/Colonoscopy-polyps-detection-with-CNNs/tree/master>

Daniel. (2023, 30 octubre). *VGG: ¿Qué es este modelo? ¡Daniel te lo cuenta todo!* Formación En Ciencia de Datos | DataScientest.com. <https://datascientest.com/es/vgg-que-es-este-modelo-daniel-te-lo-cuenta-todo>

Brital, A. (2022, 4 enero). Inception V3 CNN Architecture explained .

- Anas BRITAL - Medium. Medium. <https://medium.com/@AnasBrital98/inception-v3-cnn-architecture-explained-691cfb7bba08>

Kundu, N. (2023, 23 enero). Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation. Medium.

<https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f>