Sorbonne Paris North University

Galilée Institute

SIVO PROJECT REPPORT

Title :

Wireless Capsule Endoscopy Image Analysis

Proposed by:

Pr BEGHDADI Azeddine

Implemented by :

M. SEKKAI Samy

2022/2023

Table of Contents

List of Tables ii

List of Figure iii

Introduction 1

Objectives 1

Part 1: Image classifier 1

1. Definition 1
2. Benefits 1
3. Existing works 1
4. Architectures of the networks 2
   - 4.1 VGG16 2
   - 4.2 VGG19 3
   - 4.3 Xception (Extreme Inception) 4
5. Dataset 4
   - 5.1 Selection of Classes 5
6. Evaluation Metrics: 5
   - 6.1 Accuracy 5
   - 6.2 Precision 6
   - 6.3 Recall 6
   - 6.4 Importance of the metrics 6
7. **7.** Technical Environment 6
8. Results 7
   - 8.1 VGG16 7
   - 8.2 VGG19 9
   - 8.3 Xception 11
   - 8.4 Results Comparison 12
   - 8.5 Possible improvement of the model's performance 14

Part 2: Medical image visualisation and manipulation 15

1. Application features 15
   - 1.1 Zoom 15
   - 1.2 Rotation 15
   - 1.3 Flip 15
   - 1.4 Gray scale 15
   - 1.5 Brightness/Contrast 15
   - 1.6 Image slideshow 15
   - 1.7 Edge detection 15

2. Implementation 16
2.1 Technical environment 16
2.2 Implementation results 17
2.2.1 Image Classification 17
2.2.2 Image Slideshow 18
2.2.3 Single Image Processing and visualisation 19
2.3 Installation of the application 21
Prerequisites 21
General Conclusion 24
Summary 24
Limitations 24
Perspectives 24
References 25

List of Tables

Table 1: Kvasir Dataset(Hyper,V2, capsule) 5
Table 2: Final Dataset 5
Table 3: Performance Metrics of the three models for training, validation and test 13
Table 4: Precision and Recall values for each class of the Xception model for test set 14

List of Figure

Figure 1: Performances of different architectures trained on K-vasir dataset[4] 2
Figure 2: VGG16 architecture with extra layers 3
Figure 3: VGG19 architecture with extra layers 3
Figure 4: Xception architecture with extra layers 4
Figure 5: VGG16 Loss function evolution for training and validation 7
Figure 6: VGG16 performance evolution for training... 7
Figure 7: VGG16 performance evolution for validation 8
Figure 8: VGG16 Normalized Confusion Matrix for test Set 8
Figure 9: VGG19 Loss function evolution for training and validation 9
Figure 10: VGG19 performance evolution for training.. 9
Figure 11: VGG19 performance evolution for validation 10
Figure 12: VGG19 Normalized Confusion Matrix for test Set 10
Figure 13: Xception Loss function evolution for training and validation 11
Figure 14: Xception performance evolution for training.. 11
Figure 15: Xception performance evolution for validation 12
Figure 16: Xception Normalized Confusion Matrix for test Set 12
Figure 17: Xception original confusion Matrix 13
Figure 18: Emgu installation in Visual Studio 16
Figure 19: Main window of the image visualisation Application 17
Figure 20: Folder Explorer to select images for classification.. 17
Figure 21: Progression of the classification process 18
Figure 22: SlideShow Window... 18
Figure 23: Image zoom *2 19
Figure 24: Zoom*2, rotation (60°), and Gray scale 19
Figure 25: Sobel Filter on a zoomed Image.. 20
Figure 26: Lapacien filter on a zoomed image 20
Figure 27: Canny filter on a zoomed image.. 21
Figure 28: System variables window 22
Figure 29: Creating a python system variable. 22

Introduction

Capsule endoscopy is an alternative to colonoscopy. A capsule endoscope is a camera of the size and shape of a large pill that is swallowed to visualize the Gastro Intestinal (GI) tract. Capsule video endoscopy is an appealing alternative to traditional diagnostic techniques, since it allows inspection of the GI tract without discomfort to the patient or need for sedation, thus preventing the risks of conventional endoscopy. The reliability of the automatic anomaly detection is dependent on making high quality video recordings of the gastrointestinal tract available for post processing analysis.

Objectives

In this project we will focus on three aspects; namely the development of a graphical interface that allows the visualization of images in an intelligent way to facilitate the exploration of anomalies. The second is the development of a tool to detect images containing anomalies (we will limit ourselves to a few).

Part 1: Image classifier

1. Definition

Capsule endoscopy also known as wireless or video capsule endoscopy/interoscopy is a diagnostic procedure that involves swallowing a small capsule resistant to stomach enzymes to visualize the entire gastrointestinal tract [1]. The capsule is about the size of vitamin tablet and contains a battery, a transmitter and LED light source and colour video camera. Sensors placed on the patient abdomens receive signals transmitted by the capsule. A wireless recorder places on the waist receives and record the data sent by the sensors. The data can be visualised as images or videos on the monitor of the physicians. Capsule endoscopy helps visualise section of the small intestine that cannot easily be seen or reaches with traditional endoscopic procedures. Capsule endoscopy is recommended for the diagnosis of Gi tract conditions such as Crohn’s disease, Ulcerative colitis, Tumors, Polyps, Ulcers, Unexplained GI bleeding, etc.

2. Benefits

Some of the benefits of capsule endoscopy over standard endoscopy procedures include:

- No need for anaesthesia or sedation
- Non-invasive and painless
- No need for a hospital stay
- Helps with early and accurate diagnosis of GI problems.

3. Existing works

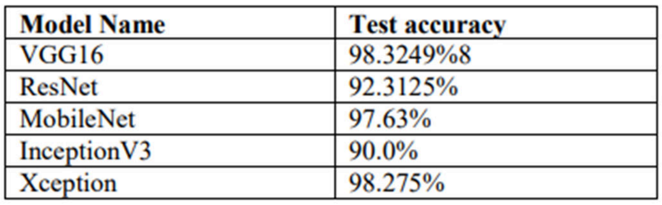
WCE procedure produces a huge amount of images (50000 ~60000)[2][3]. Visualizing this amount of data by the physicians to generate a diagnosis is a tedious and time-consuming process; add to this the possibility of missing an information during the screening process leading to wrong diagnosis.

To support the physician in this task, several computers aided diagnosis works have been proposed. The most common are, anomaly detection, anomaly segmentation, and anomaly image classification [3].

- Anomaly detection: is about detecting the presence of anomaly as well as region and class,
- Anomaly segmentation: divides the image into multiple partitions by grouping similar pixels.

- Image Classification: classify whether the image is normal or not based on the learned features.

In this project we focus on the Image classification approach. Machine learning and Deep learning models are used for image classification. Recent methods are based on deep learning architecture given their ability to capture and extract high level semantic information. In a survey study made in [3] on medical image analysis, studies on classification were organized by objective, feature extraction approach, network architecture and dataset used. Most of the listed studies were based on CNN in terms of architecture and features. They used custom datasets which makes it difficult to reproduce the performance of their models. Plus, studies focus mostly on detecting a specific abnormality such as bleeding or polyps. A recent study [4] developed a deep learning model for anomaly classification of the Gastrointestinal Tract. The authors trained state of the art architectures with K-vasir dataset. Performances of the models are shown in figure 1



Model Name	Test accuracy
VGG16	98.3249%8
ResNet	92.3125%
MobileNet	97.63%
InceptionV3	90.0%
Xception	98.275%

Figure 1: Performances of different architectures trained on K-vasir dataset[4]

Figure 1 shows that Xception and VGG models have higher accuracy than the other architectures. Thus, Xception and VGG16 architecture are selected for our project. We add to them VGG19 architecture, since it is more recent than VGG16.

4. Architectures of the networks

4.1 VGG16

An algorithm for object classification and detection. One of the popular algorithms for image classification, it is easy to use with transfer learning [5], its architecture is shown in figure 2

- The number 16 in VGG16 refers to the number of layers with weights (i.e learnable parameters layers). In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers, but it has only sixteen weight.
- VGG16 input is a image tensor of size of (224, 224) with 3 RGB channels
- VGG16 has convolution layers of 3x3 filter with stride 1 and always uses the same padding and maxpooling layer of 2x2 filter of stride 2; instead of having many hyper-parameters.
- The convolution and max pool layers are consistently arranged throughout the architecture.
- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.

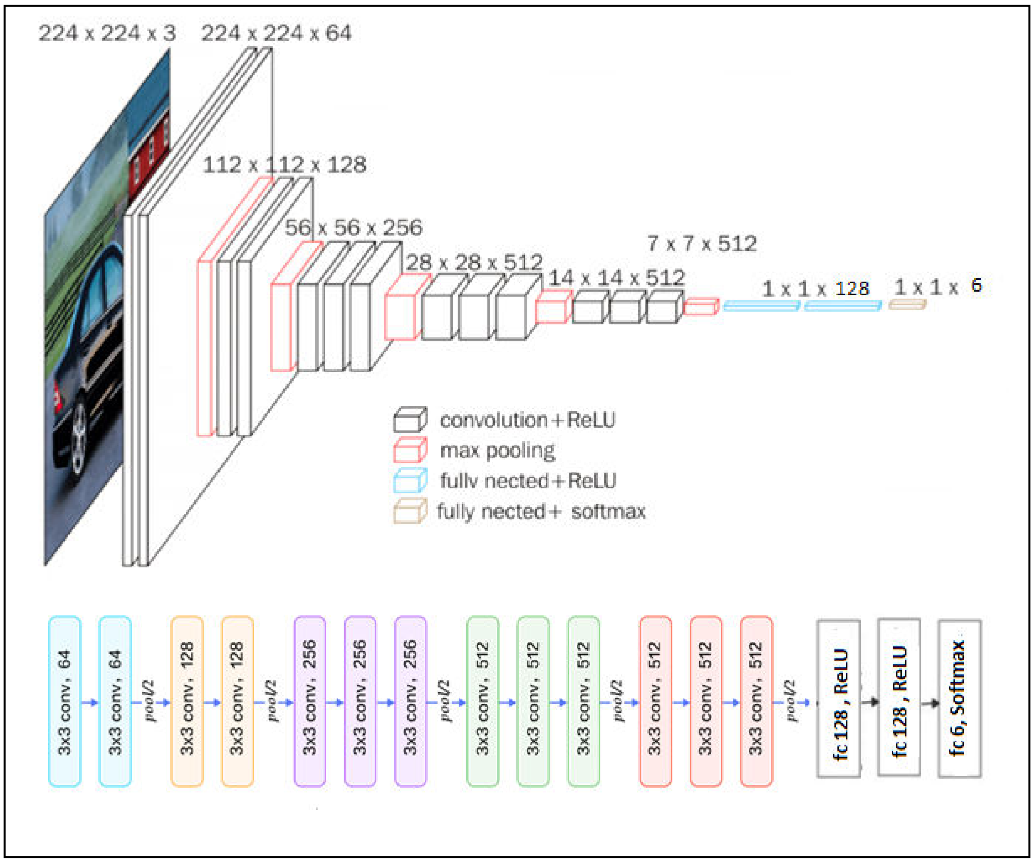


Figure 2: VGG16 architecture with extra layers

For this project, we added to the base model three layers which are:

- Fully connected (128) with ReLu activation function,
- Fully connected (128) with ReLu activation function
- Fully connected (6) with softwmax activation function

4.2 VGG19

This architecture is similar to the one of VGG16; the difference resides on the number of layers in convolutional blocs. While VGG16 has 3 layers in the Bloc 3, 4 and 5, VGG19 comes with 4 layers, VGG19 comes with 4 layers resulting in an overall of 19 learnable layers(with weights).[6]. Its architecture is shown in figure 3

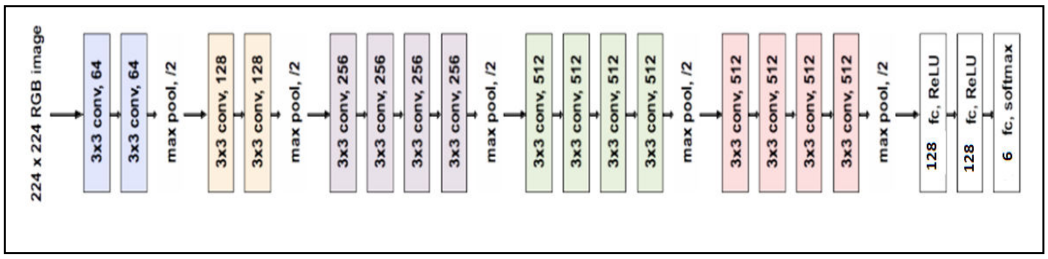


Figure 3: VGG19 architecture with extra layers

For this project, we added to the base model three layers which are:

- Fully connected (128) with ReLu activation function,
- Fully connected (128) with ReLu activation function
- Fully connected (6) with softwmax activation function

4.3 Xception (Extreme Inception)

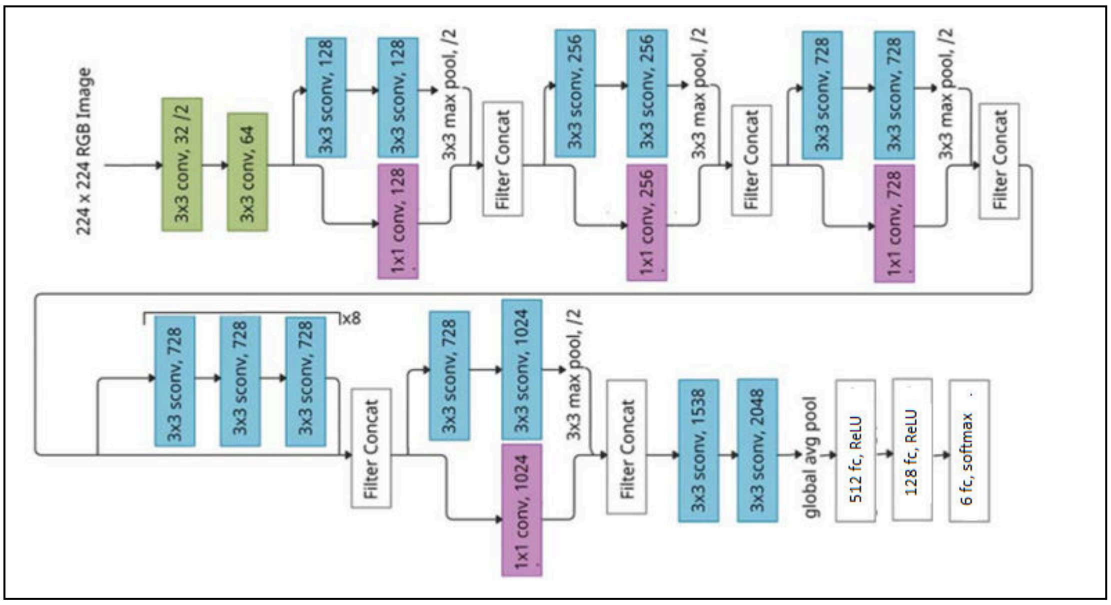


Figure 4: Xception architecture with extra layers

The model is inspired from CNN inception model, and it consists of [7] :

- 36 convolution layers that enable feature extraction.
- The layers are structured in 14 modules (one module is repeated eight times),
- Except the first and last one, all modules have linear direct connections to one another.
- The Depth Wise Separable Convolution (DWSC) is a spatial convolution that is executed independently of one another in parallel via each input channel and is followed by a point-by point 1×1 convolution that projects the output of the channel onto a new channel.

For this project, we added to the base model three layers which are:

- Fully connected (512) with ReLu activation function,
- Fully connected (128) with ReLu activation function
- Fully connected (6) with softwmax activation function

5. Dataset

To train our classification models, we used hyper-kvasir (HK)[8], kvasirV2 (K2)[9],and kvasir-Capsule(KC)[10].The merged dataset contained the following classes.

Class	**Sample size**	**Source**
Barretts	41	HK
Barretts-short-segment	53	HK
Bbps-0-1 Bbps-1-2	1794	HK
Cecum	2009	HK+K2
Dyed-lifted-polyps	1002	HK+K2
Dyed-resection-margins	1989	HK+K2
Esophagitis-a b d	1663	HK+K2
Hemorrhoids	6	HK
Ileum	9	HK

Impacted-stool	131	HK
Polyps	2695	HK+K2
Pylorus	3528	HK+K2
Retroflex-rectum	391	HK
Retroflex-stomach	764	HK
Ulcerative-colitis-grade-0-1	1851	K2
Z-line	1932	K2
Ulcer	854	KC
Ampulla_of_vater	10	KC
Angiectasia	866	KC
Blood_fresh	466	KC
Blood_hematin	12	KC
Erosion	506	KC
Erythema	159	KC
Foreign_body	776	KC
Ileocecal_valve	4189	KC
Lymphangiectasia	592	KC
Normal_clean_mucosa	34338	KC
Reduced_mucosal_view	2906	KC

Table 1: Kvasir Dataset(Hyper,V2, capsule)

5.1 Selection of Classes

According to the description of Kvasir dataset[9];

Anatomical Landmarks: recognizable feature within the GI tract that is easily visible through the endoscope. These features are vital for navigating and as a reference point to describe the location of a given finding. The landmarks are “Cecum”, “Z_line”, “and Pylorus”.

Pathological finding: is “abnormal feature within the gastrointestinal tract. Endoscopically, it is visible as a damage or change in the normal mucosa.” The main pathologies described are “Esophagitis”, “Polyps”, and “Ulcerative Colitis”.

As our objective is detection of abnormal images in the GI tract, we select the three pathological classes, and the three anatomical landmarks. Thus, our final dataset contains six classes (three normal, and three pathological).

To get a balanced dataset we selected the lowest class occurrence as reference which is Esophagitis (1663 image), so each class will have 1600 sample in the dataset.

For training the models, we follow the rule of 80:10:10; the dataset is split into train, validation, and test.

Data	Sample per class	Total
Train	1200	7200
Validation	200	1200
Test	200	1200

Table 2: Final Dataset

6. Evaluation Metrics:

6.1 Accuracy

Model accuracy is defined as the number of classifications a model correctly predicts divided by the total number of predictions made. It’s a way of assessing the performance of a model.

6.2 Precision

Attempts to answer the following question: What proportion of prediction as positive was actually correct?

6.3 Recall

Attempts to answer the following question: What proportion of actual positives was identified correctly?

6.4 Importance of the metrics

When all classes have the same importance in the model, Accuracy is enough to evaluate the model's performance. However, if the different classes vary in importance, precision and recall are used as reference. The best model would have a perfect precision and recall.[19]

7. Technical Environment

The models were trained using python 3.10 in Anaconda environment and Jupyterlab IDE.
TensorFlow and Keras libraries were used for training the models.

8. Results

8.1 VGG16

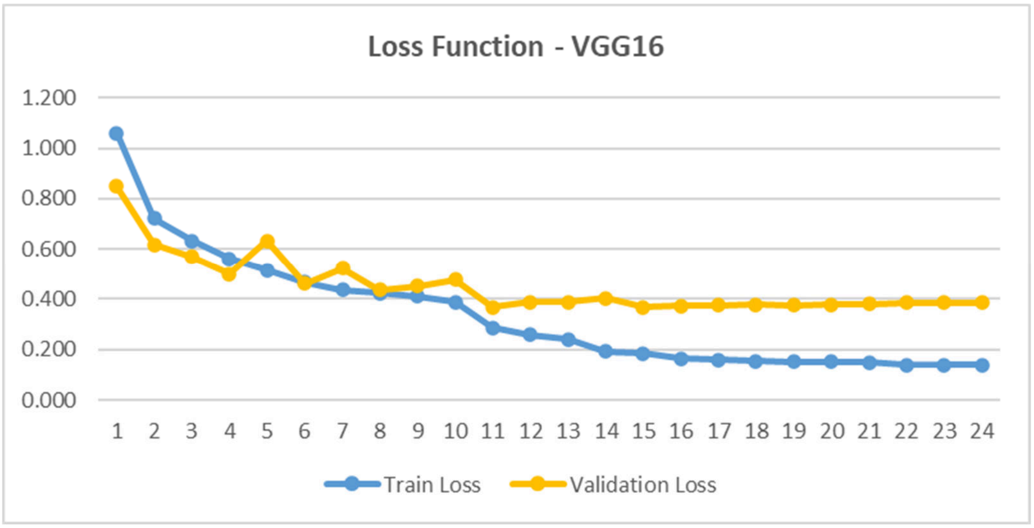


Figure 5: VGG16 Loss function evolution for training and validation

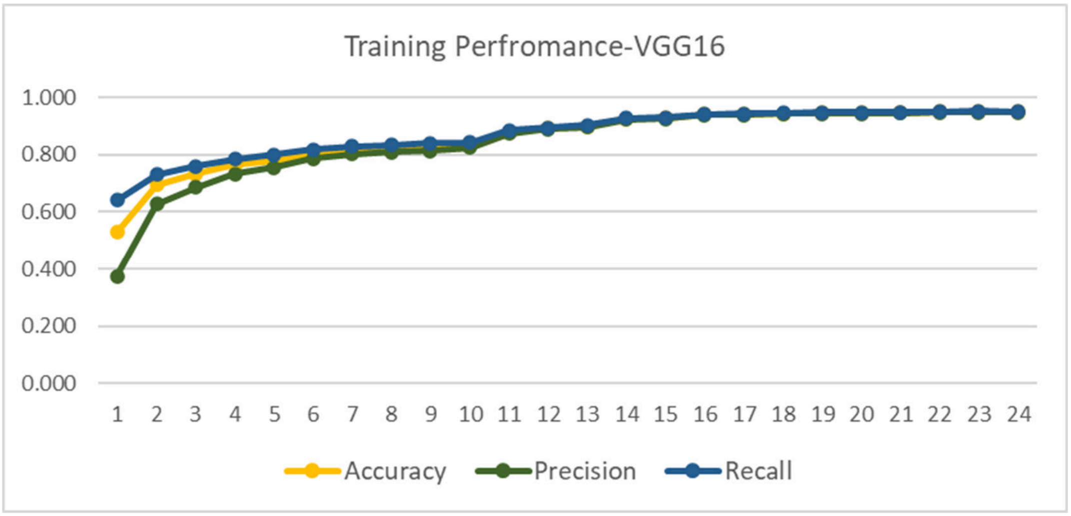


Figure 6: VGG16 performance evolution for training

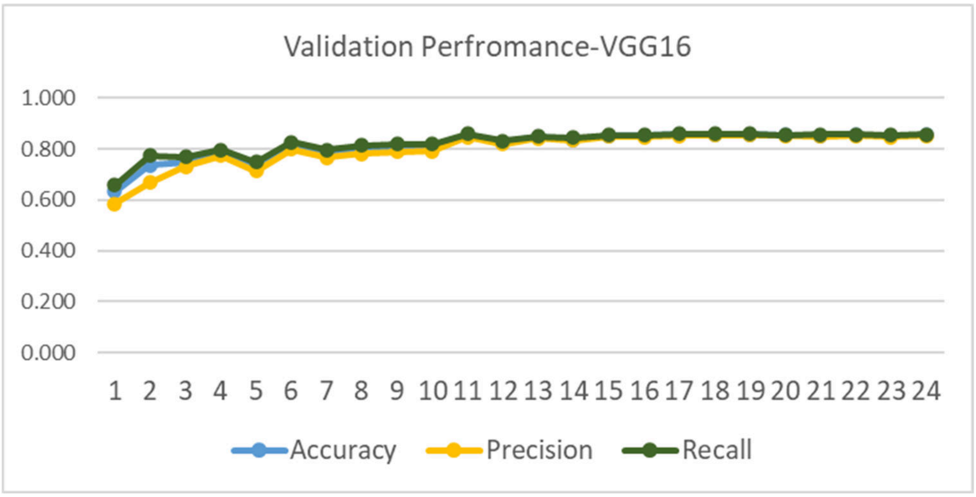
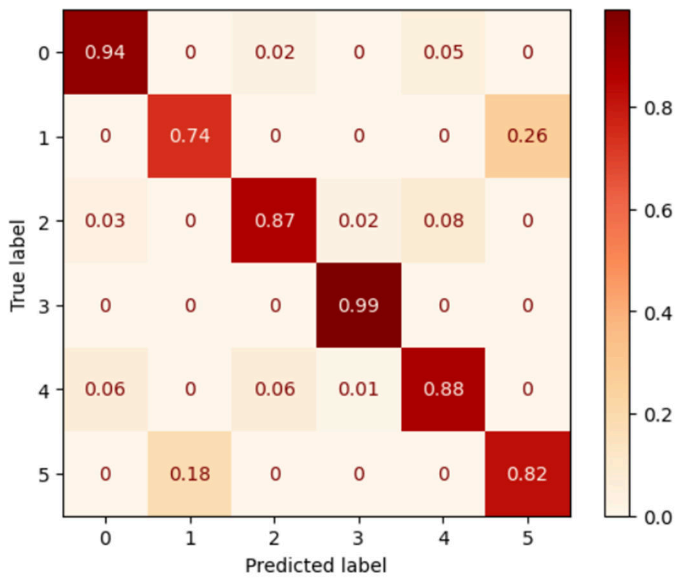


Figure 7: VGG16 performance evolution for validation



True label \ Predicted label	0	1	2	3	4	5
0	0.94	0	0.02	0	0.05	0
1	0	0.74	0	0	0	0.26
2	0.03	0	0.87	0.02	0.08	0
3	0	0	0	0.99	0	0
4	0.06	0	0.06	0.01	0.88	0
5	0	0.18	0	0	0	0.82

Figure 8: VGG16 Normalized Confusion Matrix for test Set

8.2 VGG19

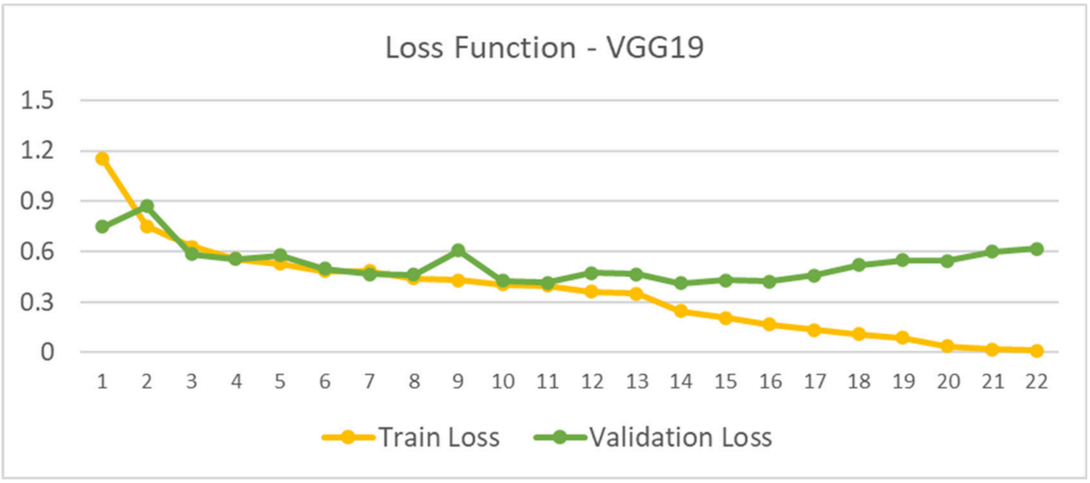


Figure 9: VGG19 Loss function evolution for training and validation

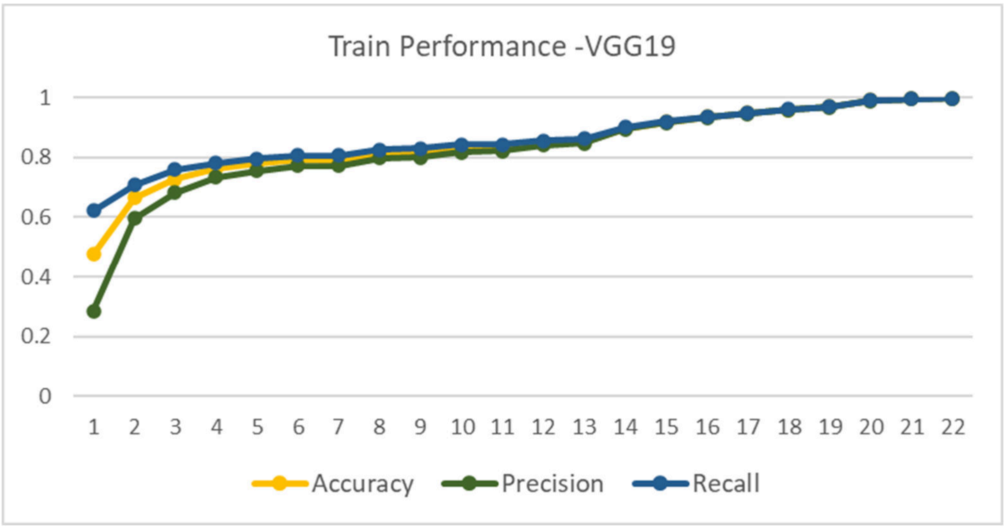


Figure 10: VGG19 performance evolution for training

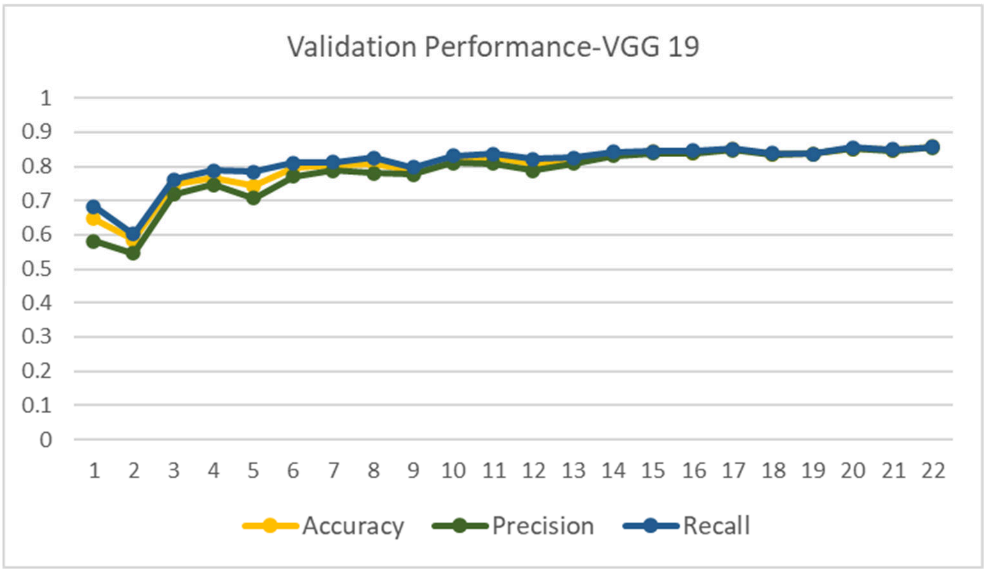
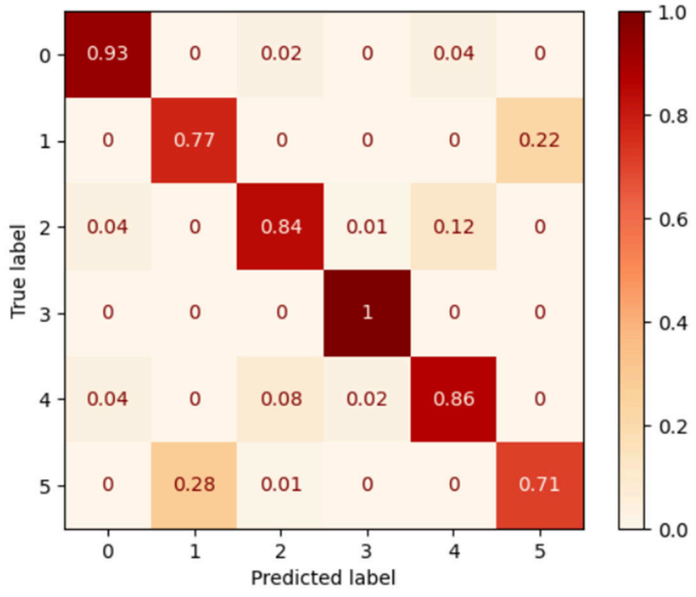


Figure 11: VGG19 performance evolution for validation



	0	1	2	3	4	5
0	0.93	0	0.02	0	0.04	0
1	0	0.77	0	0	0	0.22
2	0.04	0	0.84	0.01	0.12	0
3	0	0	0	1	0	0
4	0.04	0	0.08	0.02	0.86	0
5	0	0.28	0.01	0	0	0.71

Figure 12: VGG19 Normalized Confusion Matrix for test Set

8.3 Xception

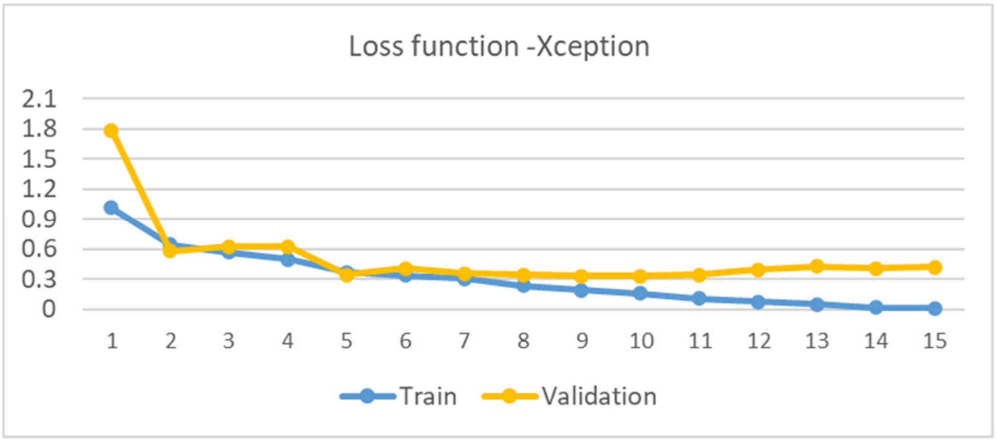


Figure 13: Xception Loss function evolution for training and validation

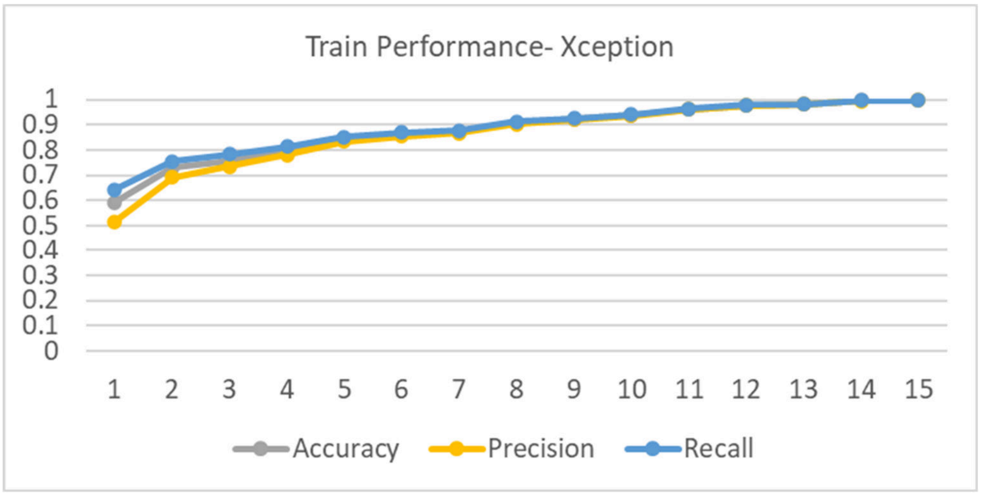


Figure 14: Xception performance evolution for training

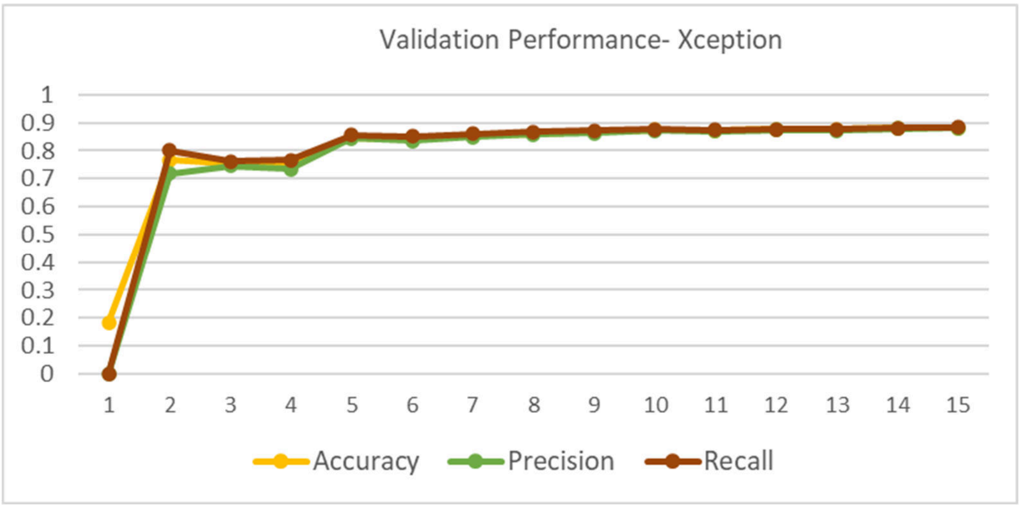


Figure 15: Xception performance evolution for validation

Normalized Confusion matrix.

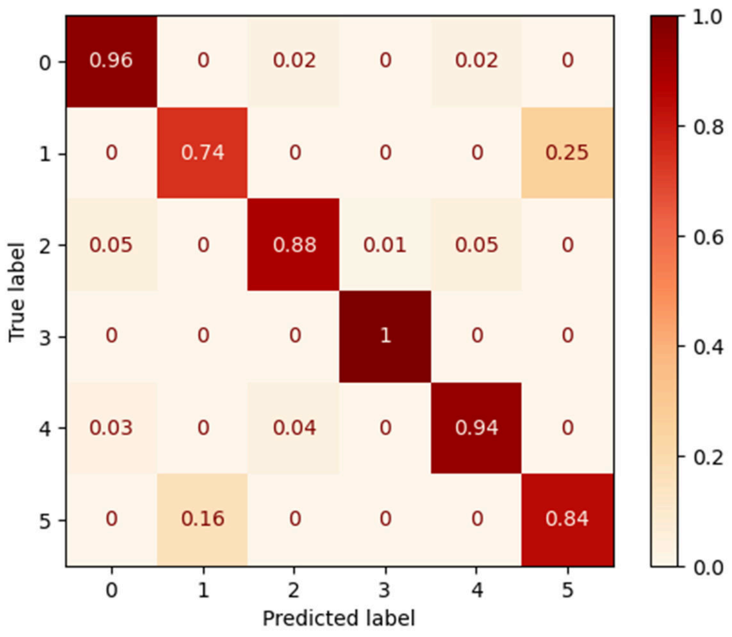


Figure 16: Xception Normalized Confusion Matrix for test Set

8.4 Results Comparison

In our project, the objective is to identify abnormal images, the best model should be able to catch all the abnormal images. The model should stricter toward misclassifying abnormal images as normal than having normal images classified as abnormal. The physician can correct a mistake in reviewing the abnormal images, however, if an abnormal is skipped he will miss an information needed for diagnosis. In this case, we should focus more on recall metric than precision metric.

To compare the performance of the models, we run the evaluation process with the test dataset. The summary of the performances of the models in training, validation, and test are presented in table 3

Model	**Train**			**Validation**			**Test**		
Metrics	**Acc**	**Pre**	**Rec**	**Acc**	**Pre**	**Rec**	**Acc**	**Pre**	**Rec**
VGG16	0.951	0.949	0.952	0.853	0.852	0.856	0.853	0.856	0.85
VGG19	0.998	0.998	0.998	0.857	0.856	0.858	0.852	0.855	0.85
Xception	0.997	0.997	0.997	0.882	0.881	0.883	0.882	0.881	0.883

Table 3: Performance Metrics of the three models for training, validation and test

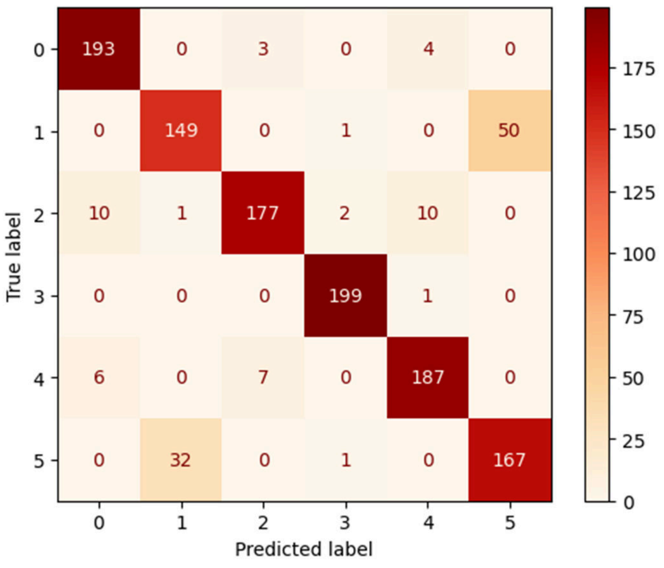
Results in table 3 show that Xception model outperforms VGG16 and VGG19 in validation and tests for all metrics.

To analyse, more, the performance of Xception model, we calculate the precision and Recall of the model for each class.

For normal classes, ***Cecum*** and ***Pylorus*** show the highest values for Recall and Precision, meaning the model can classify almost all the sample images of the two classes correctly (Recall). At the same time, it has low confusion with other classes, 92% of images classified as ***Cecum*** were ***Cecum***, and 98% of the images classified as ***Pylorus*** were ***Pylorus*** images.

From the normalized confusion matrix, we have 2% of ***Cecum*** images classified as ***Polyp*** and 2% classified as ***Ulcerative_colitis***. In case of ***Pylorus*** only one sample(figure17) was misclassified as ***Ulcerative_colitis***.

The model has the highest confusion between ***Z_line*** and ***Esophagitis***. **16%** of ***Z_line*** images are misclassified as ***Esophagitis***, and **25%** of ***Esophagitis*** are misclassified as ***Z_line***. the problem is the 25% , this mean that high number of Esophagitis anomaly are not Detected.



	0	1	2	3	4	5
0	193	0	3	0	4	0
1	0	149	0	1	0	50
2	10	1	177	2	10	0
3	0	0	0	199	1	0
4	6	0	7	0	187	0
5	0	32	0	1	0	167

Figure 17: Xception original confusion Matrix

Class		**Precision**	**Recall**
Cecum	0	0.92	0.96
Esophagitis	1	0.82	0.74
Polyp	2	0.95	0.88
Pylorus	3	0.98	0.99
Ulcerative_colitis	4	0.93	0.94
Z_Line	5	0.77	0.84

Table 4: Precision and Recall values for each class of the Xception model for test set

8.5 Possible improvement of the model's performance

An important factor that affects the model performance is the dataset size; in this project we only selected 1200 sample of each class of training. Increasing the dataset size will improve noticeably the performance of the model. However, for some classes like Z_line sample size is limited in the original kvasir dataset. We could use other approach to increase the dataset by applying data augmentation technique. Due to time limitation, we were not able to apply this technique on our dataset. But as perspective in improving the performance we plan to use the data augmentation technique in the future [11].

Part 2: Medical image visualisation and manipulation.

The second part of this project consists of implementing an application that allows the physician to explore and manipulate medical images in an intelligent way. Medical images are automatically classified to normal and abnormal classes. This classification is a computer-based decision helper tool. It also reduces the time needed to screen and explore the images.

1. Application features

Multiple images manipulation functions are implemented. The application has the following features:

1.1 Zoom

The image can be zoomed in or out. Predefined zoom sizes are set [0.25, 0.5, 0.66, 0.75, 1, 2,3,4]

1.2 Rotation

The image will be rotating with an angle to be defined,

1.3 Flip

An image can be flipped horizontally (right/left) or vertically (up/down)

1.4 Gray scale

The image is converted from RGB space to Gray.

1.5 Brightness/Contrast

Brightness and contrast of an image can be increased or decreased when needed,

1.6 Image slideshow

Images in a folder are displayed in a slideshow.

1.7 Edge detection

Three algorithms are implemented:

1.7.1 Sobel operator

The Sobel filter is one of the most used edge detectors. It is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. The Sobel edge enhancement filter has the advantage of providing differentiating (which gives the edge response) and smoothing (which reduces noise) concurrently [12].

1.7.2 Laplacien operator

A Laplacian filter is an edge detector used to compute the second derivatives of an image, measuring the rate at which the first derivatives change. This determines if a change in adjacent pixel values is from an edge or continuous progression.

Laplacian filter kernels usually contain negative values in a cross pattern, centred within the array. The corners are either zero or positive values. The centre value can be either negative or positive. The following array is an example of a 3x3 kernel for a Laplacian filter.[13]

1.7.3 Canny edge detection

Canny edge detection uses linear filtering with a Gaussian kernel to smooth noise and then computes the edge strength and direction for each pixel in the smoothed image, Candidate edge pixels are identified as the pixels that survive a thinning process called non-maximal suppression. In this process, the edge strength of each candidate edge pixel is set to zero if its edge strength is not larger than the edge strength of the two adjacent pixels in the gradient direction. Thresholding is then done on the thinned edge magnitude image using hysteresis. In hysteresis, two edge strength thresholds are used. All candidate edge pixels below the lower threshold are labelled as non-edges and all pixels above the low threshold that can be connected to any pixel above the high threshold through a chain of edge pixels are labelled as edge pixels.[14]

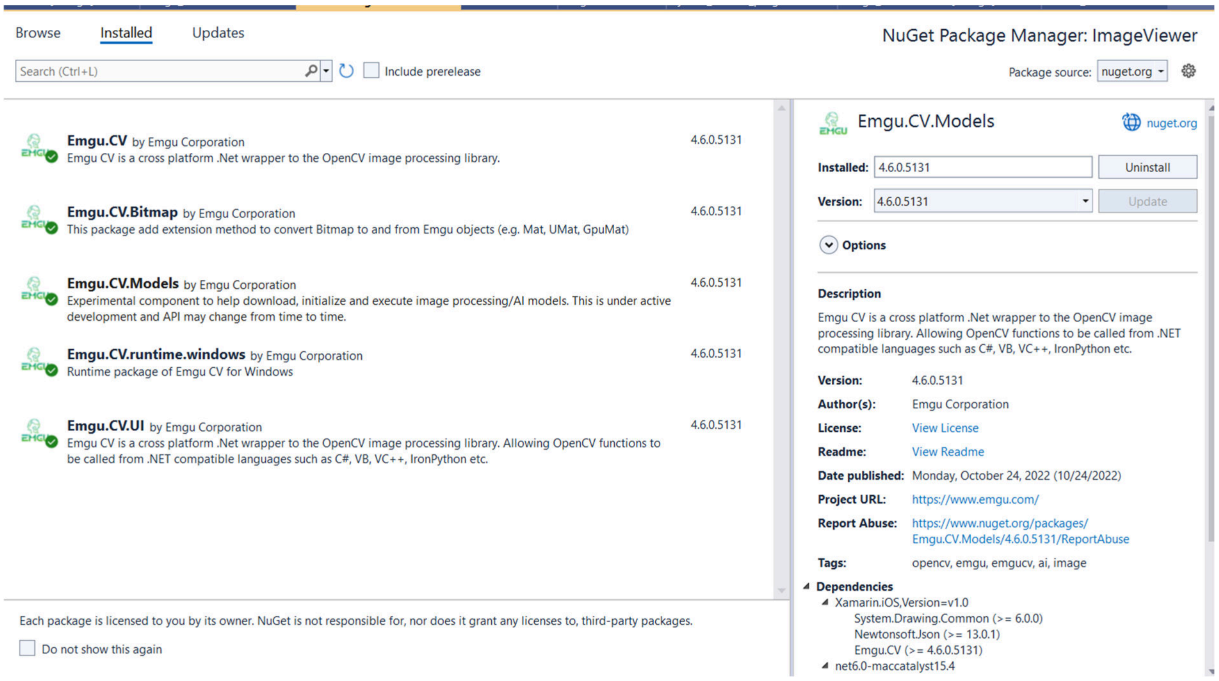
2. Implementation

2.1 Technical environment

Visual studio 2022 is used for implementing the application; we used C# as programming language for its ability to embed python language to call our classification model which was trained with python.

2.1.1 Emgu Library

Emgu CV is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages. The wrapper can be compiled by Visual Studio and Unity, it can run on Windows, Linux, Mac OS, iOS and Android. The library provides implementations of most of image processing functions and algorithms offered by OpenCV [15]. We installed this library in our project and used it to implement image processing functions. The installation can be done through Nuget Package Manager of Visual Studio as shown in figure 18.



Emgu.CV by Emgu Corporation 4.6.0.5131
Emgu CV is a cross platform .Net wrapper to the OpenCV image processing library.

Emgu.CV.Bitmap by Emgu Corporation 4.6.0.5131
This package add extension method to convert Bitmap to and from Emgu objects (e.g. Mat, UMat, GpuMat)

Emgu.CV.Models by Emgu Corporation 4.6.0.5131
Experimental component to help download, initialize and execute image processing/AI models. This is under active development and API may change from time to time.

Emgu.CV.runtime.windows by Emgu Corporation 4.6.0.5131
Runtime package of Emgu CV for Windows

Emgu.CV.UI by Emgu Corporation 4.6.0.5131
Emgu CV is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

Emgu.CV.Models nuget.org

Installed: 4.6.0.5131 Uninstall

Version: 4.6.0.5131 Update

Options

Description

Emgu CV is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc.

Version:	4.6.0.5131
Author(s):	Emgu Corporation
License:	View License
Readme:	View Readme
Date published:	Monday, October 24, 2022 (10/24/2022)
Project URL:	https://www.emgu.com/
Report Abuse:	https://www.nuget.org/packages/Emgu.CV.Models/4.6.0.5131/ReportAbuse
Tags:	opencv, emgu, emgucv, ai, image

Dependencies
- Xamarin.iOS,Version=v1.0
  - System.Drawing.Common (>= 6.0.0)
  - Newtonsoft.Json (>= 13.0.1)
  - Emgu.CV (>= 4.6.0.5131)
- net6.0-maccatalyst15.4

Figure 18: Emgu installation in Visual Studio

2.1.2 Call of python code in the C# application

to be able to call the classification model from our C# application we must satisfy some prerequisites.

- Install python 3.8
- Install TensorFlow
- Install pillow,
- Add python dll as environment variable in the project;[18]

2.2 Implementation results

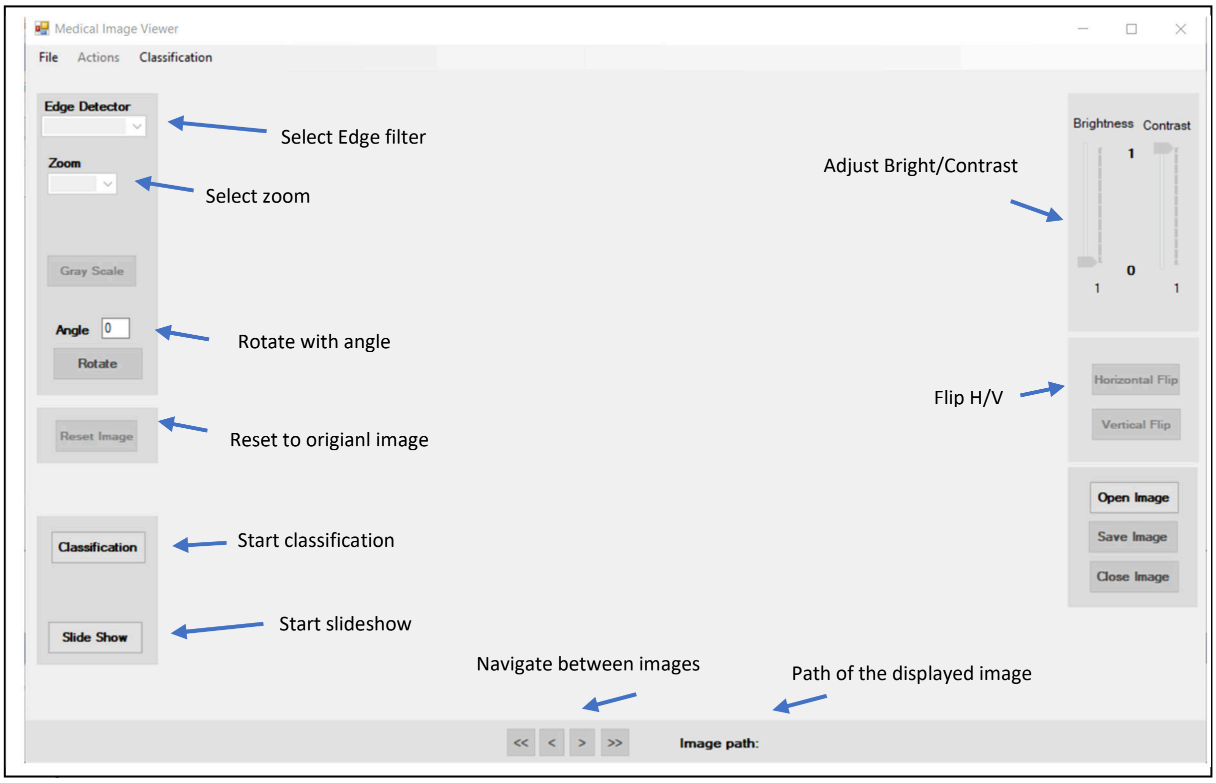


Figure 19: Main window of the image visualisation Application

2.2.1 Image Classification

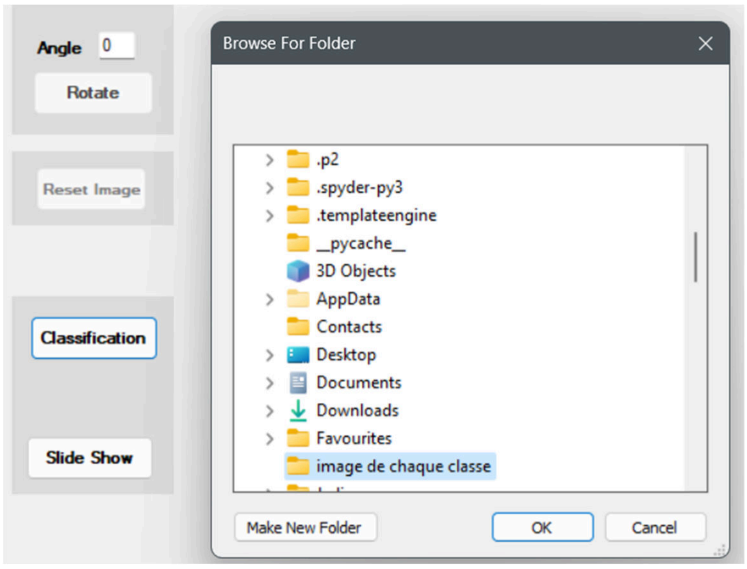


Figure 20: Folder Explorer to select images for classification.

Once the folder selected, Python execution environment is loaded, and image classes are predicted. We can see the progression of the classification process through the progress bar. We can notice a delay between selecting the folder of images and the beginning of the classification, this is due to the time needed to load python execution environment in the application. At the end of the process, images are copied to a new folder "***Classified_Images***" which will contain sub folders one for each class. The "Classified_Images" folder is created in the same drive (expel: c:\ Classified_Images, d:\ Classified_Images) of the installed application.

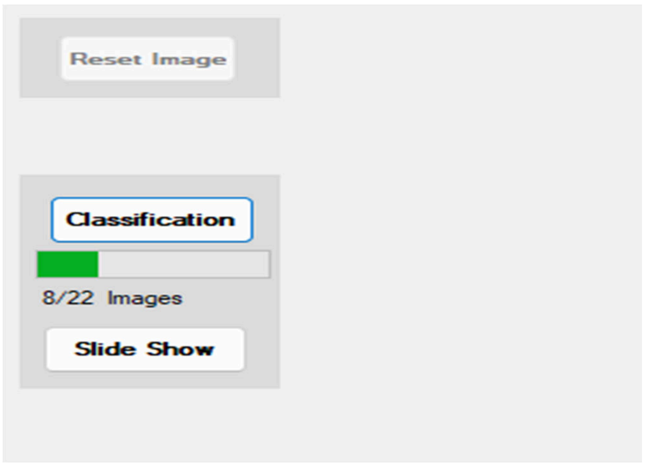
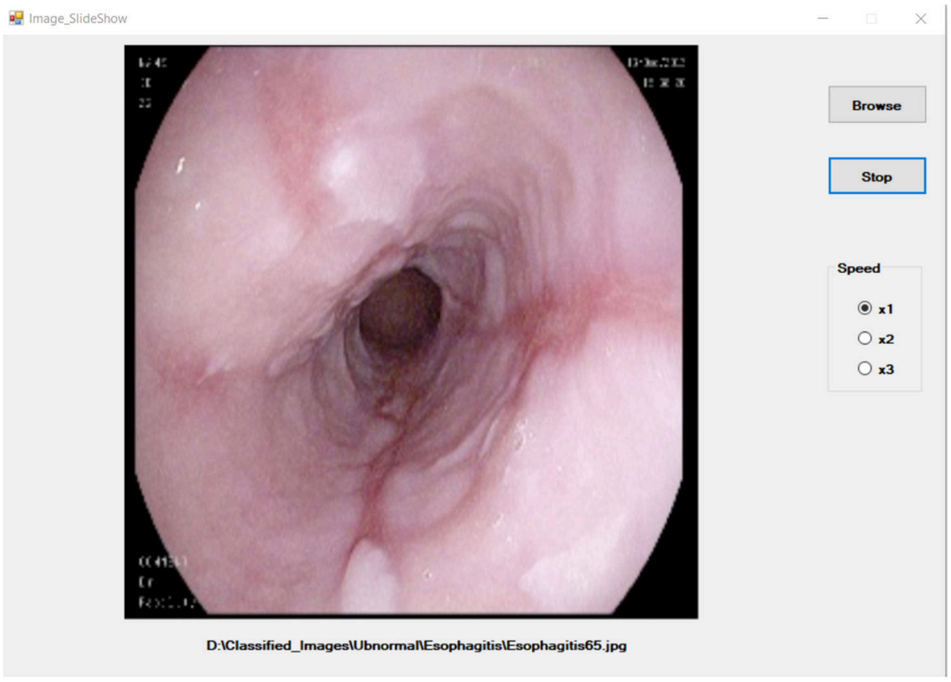


Figure 21: Progression of the classification process

2.2.2 Image Slideshow

We can visualise the images as a slideshow as show in figure by clicking on the "Slide show" button on the main form. We can vary the speed of visualisation of the image to by 2 or 3 times the original speed which is 3 seconds.



Speed: [✓ x1] x2 x3

D:\Classified_Images\Ubnormal\Esophagitis\Esophagitis65.jpg

Figure 22: SlideShow Window

2.2.3 Single Image Processing and visualisation

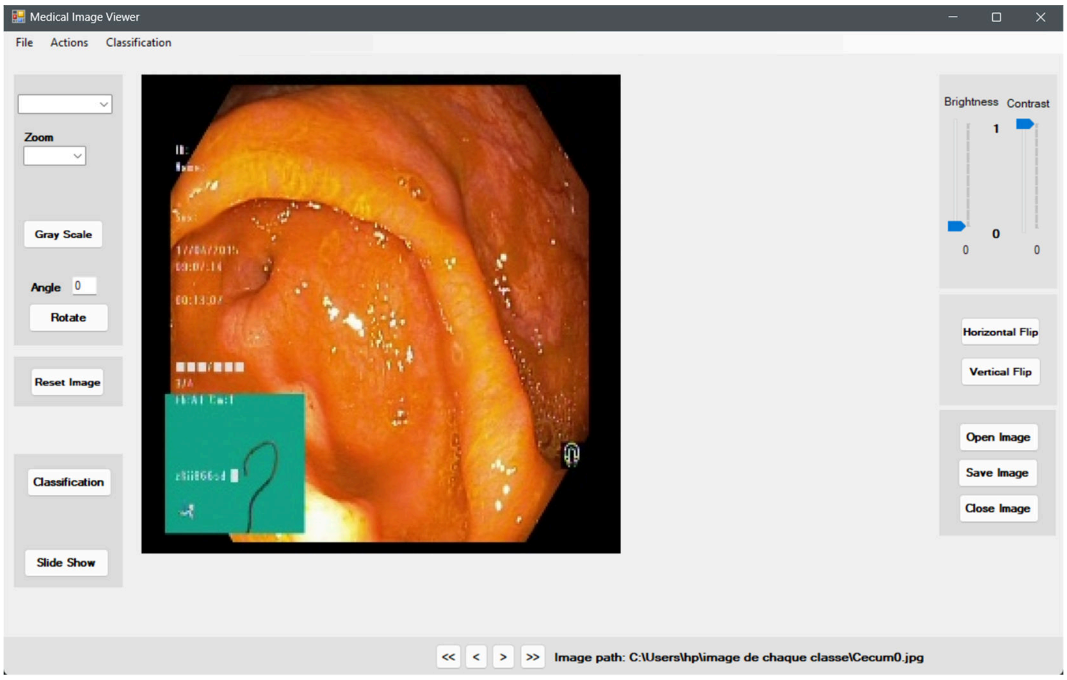


Image path: C:\Users\hp\image de chaque classe\Cecum0.jpg

Figure 23: Image zoom *2

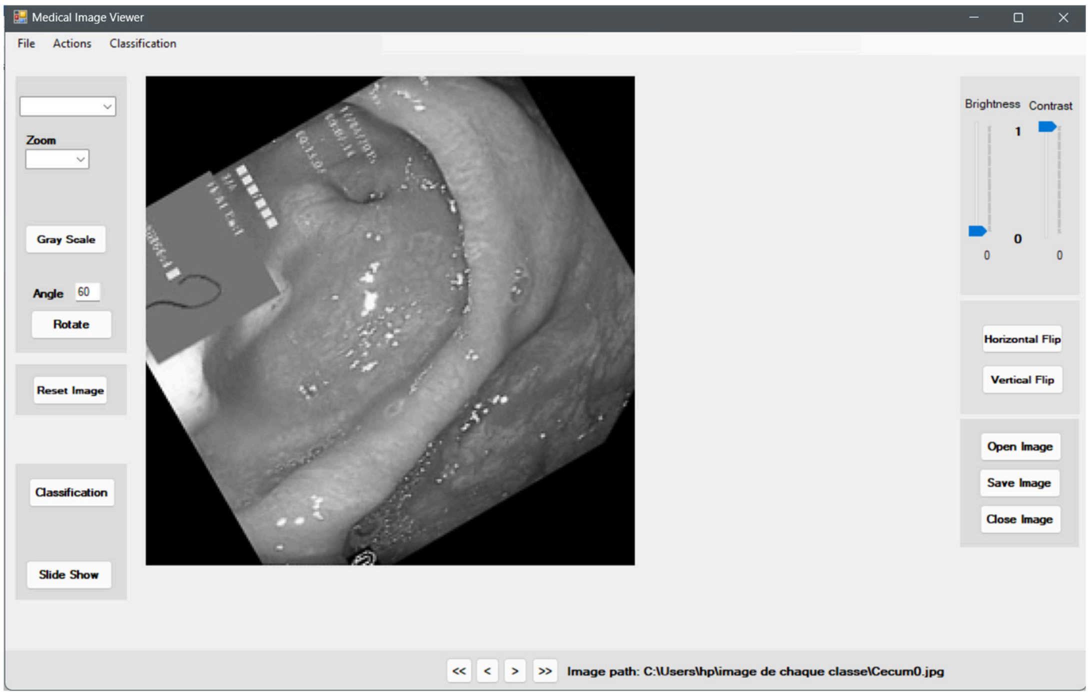


Image path: C:\Users\hp\image de chaque classe\Cecum0.jpg

Figure 24: Zoom*2, rotation (60°), and Gray scale

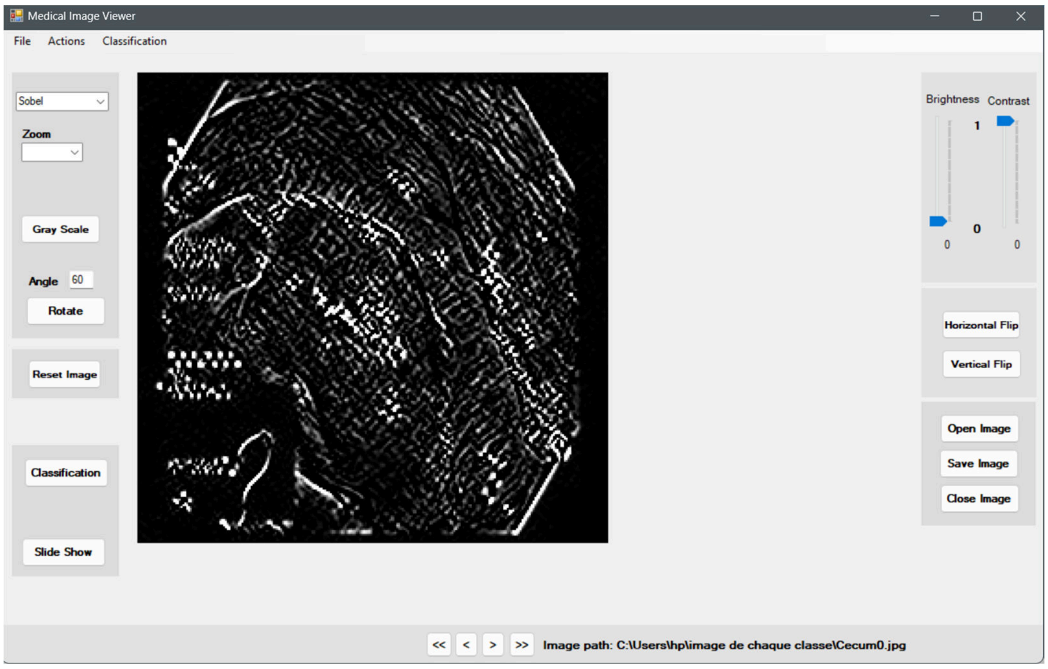


Image path: C:\Users\hp\image de chaque classe\Cecum0.jpg

Figure 25: Sobel Filter on a zoomed Image

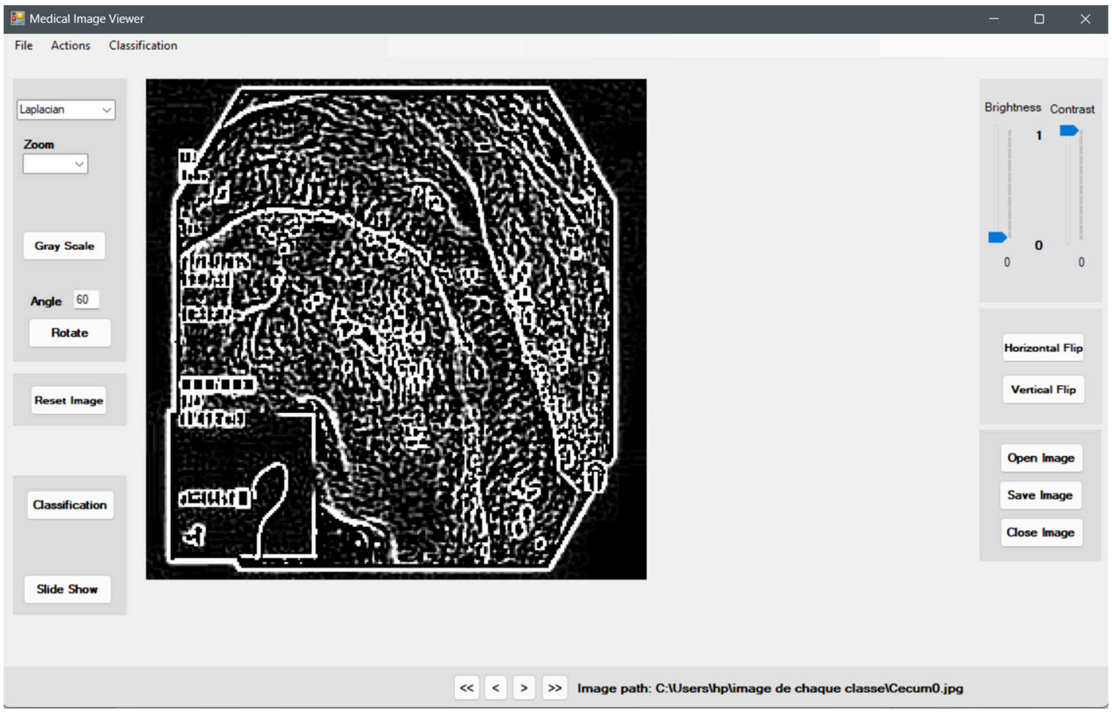


Image path: C:\Users\hp\image de chaque classe\Cecum0.jpg

Figure 26: Lapacien filter on a zoomed image

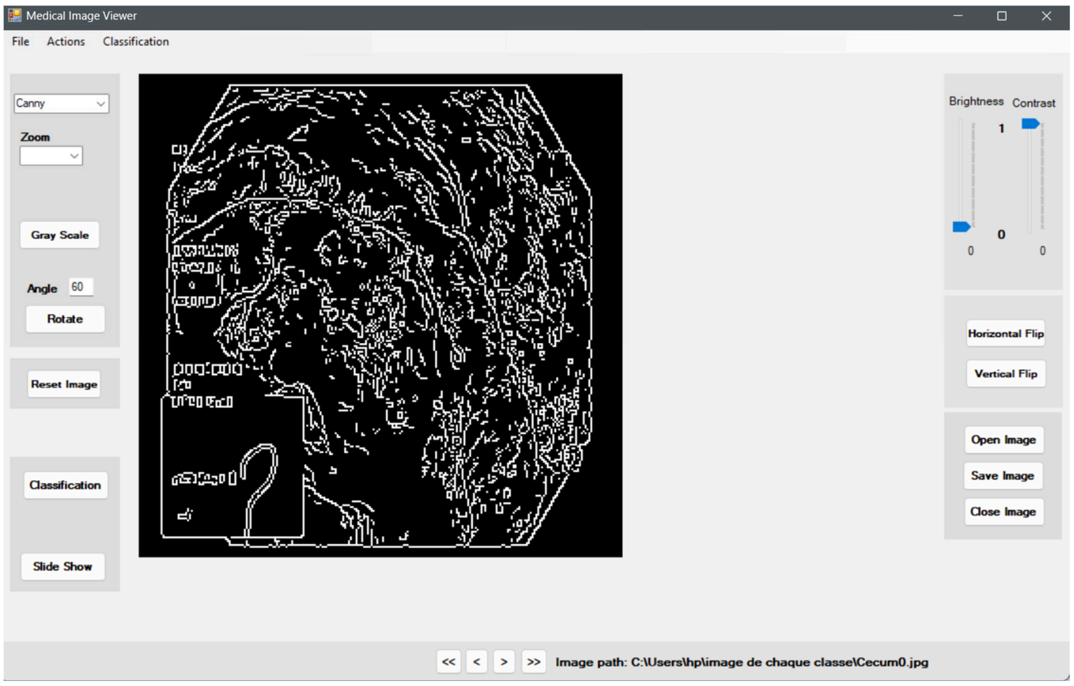


Figure 27: Canny filter on a zoomed image

2.3 Installation of the application

To be able to run properly the application, the following tools should be installed on the computer.

Prerequisites

<u>Framework .NET</u>

The framework might already be present on the computer , if not install the following version .Net Framework 4.8 from this link : https://go.microsoft.com/fwlink/?linkid=2088631

<u>Python 3.80</u> .

- Download from this link : https://www.python.org/ftp/python/3.8.0/python-3.8.0-amd64.exe
- Add python to environment variables by ticking the checkbox.

<u>Tensorflow</u>

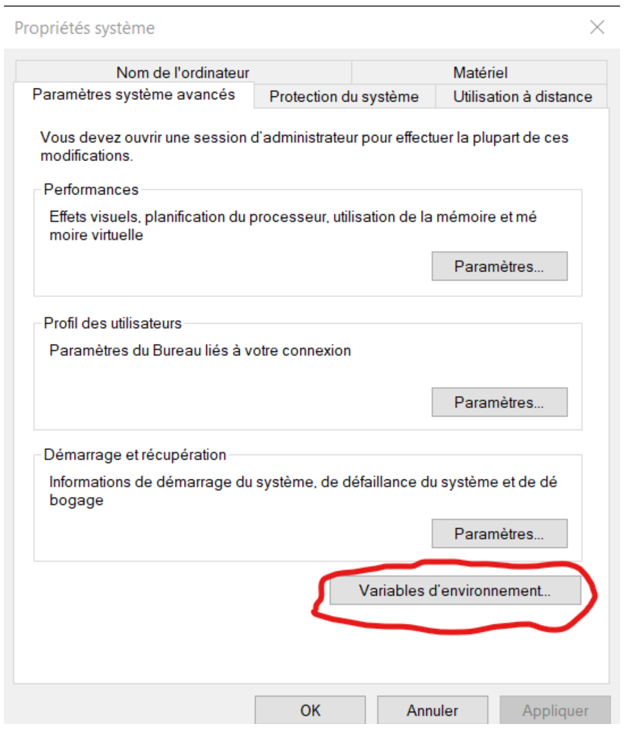
- From the windows command execute the following command: ***pip install tensorflow***

<u>Pillow</u>

- From the windows command execute the following command: ***pip install pillow***

<u>Add python to System Environment Variable:</u>

A system environment variable “PYTHONDLL” should be added to System Variables You can access the system variables from system settings, then search for system variables; the system properties window will appear as shown in figure 28.



Propriétés système

Nom de l'ordinateur | Matériel | Paramètres système avancés | Protection du système | Utilisation à distance

Vous devez ouvrir une session d'administrateur pour effectuer la plupart de ces modifications.

Performances

Effets visuels, planification du processeur, utilisation de la mémoire et mémoire virtuelle

Paramètres...

Profil des utilisateurs

Paramètres du Bureau liés à votre connexion

Paramètres...

Démarrage et récupération

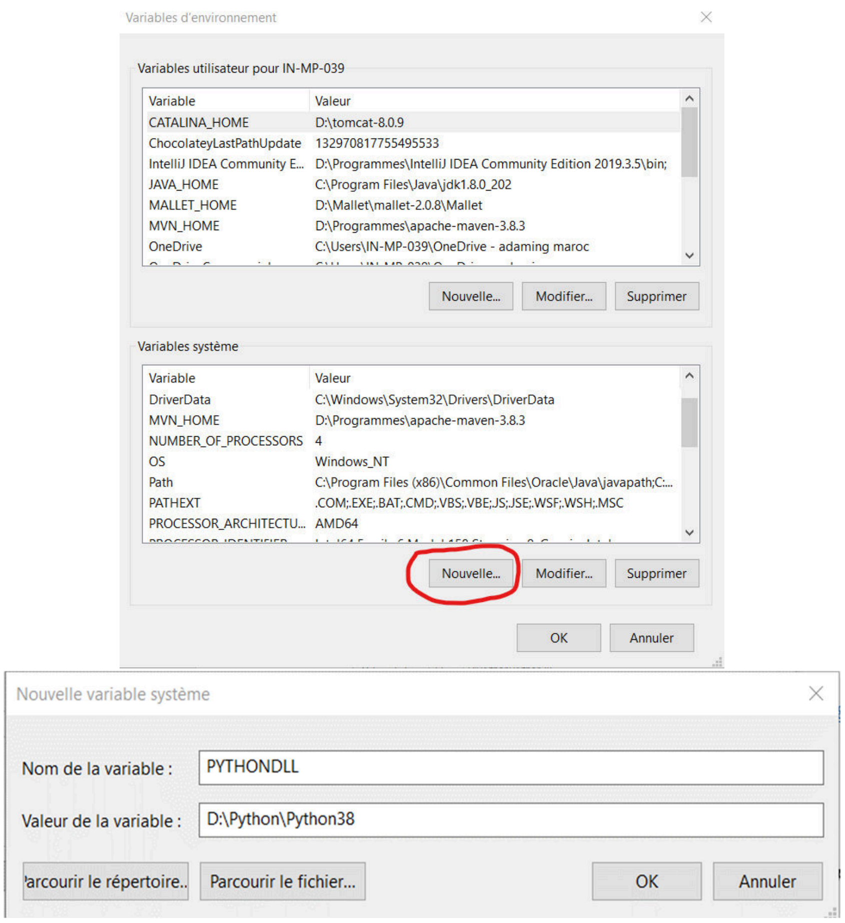
Informations de démarrage du système, de défaillance du système et de débogage

Paramètres...

Variables d'environnement...

OK | Annuler | Appliquer

Figure 28: System variables window



Variables d'environnement

Variables utilisateur pour IN-MP-039

Variable	Valeur
CATALINA_HOME	D:\tomcat-8.0.9
ChocolateyLastPathUpdate	132970817755495533
IntelliJ IDEA Community E...	D:\Programmes\IntelliJ IDEA Community Edition 2019.3.5\bin;
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_202
MALLET_HOME	D:\Mallet\mallet-2.0.8\Mallet
MVN_HOME	D:\Programmes\apache-maven-3.8.3
OneDrive	C:\Users\IN-MP-039\OneDrive - adaming maroc

Nouvelle... | Modifier... | Supprimer

Variables système

Variable	Valeur
DriverData	C:\Windows\System32\Drivers\DriverData
MVN_HOME	D:\Programmes\apache-maven-3.8.3
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64

Nouvelle... | Modifier... | Supprimer

OK | Annuler

Nouvelle variable système

Nom de la variable : PYTHONDLL

Valeur de la variable : D:\Python\Python38

'arcourir le répertoire... | Parcourir le fichier... | OK | Annuler

Figure 29: Creating a python system variable.

The python variable can be created by setting its name as "“PYTHONDLL” and value the path to the folder containing python.exe (refer to the folder where python is installed).

Once the requirement installed, the application can be installed. The installation will create a shortcut on the desktop. The application can be uninstalled from “remove application” in windows system.

General Conclusion

Summary

The objective of this project was to develop an application that allows a medical practitioner to visualize in an intelligent way medical capsule endoscopic images and easily manipulate them, to help him defining a diagnisis. The solution developped, allows the classification of images from Gastrointestinal tract and classify them as normal and abnormal. Three normal classes were used which are “Cecum” , “Z_line”, and “Pylorus”. Abnormal classes that are detected are “Polyps” , “Esophagitis”, and “Ulcerative_colitis”. A windows desktop application use inference to the model to classify images when needed. The application also allows to visualise images as slideshow, or manipuate a single image and apply several filters and functions on it to help the practionner is defining a final diagnosis.

This project allowed me to apply AI and deep learning in real problems; First, how to collect data and build dataset, and how to label them. I also worked on training models and using different metrics to compare the performances, and study the role and importance of the metrics for the problem to solve. for the develoment part, I worked on an application, and learned how to infer AI models from widows applications, and how to appy image processing, I also worked with several libraries .

Limitations

The important limitation in this work is about the classification model; Xception model showed better performances than VGG models, however it has high confusion for some classes. One of the important problem was the dataset. The size was not big enough for the model to learn more. Due to time limitation for the project, I couldn’t solve this problem.

For the application, I implemented the commun image processing functions, other useful functions and filters could be more adequate for medical field.

Perspectives

To improve the performance of the model, I plan to apply data augmentation, to increase the size of data. Multiple augmentations can be applied to generate synthetics images from the existing one to increase the size of the dataset can improve the performance. We added three layers dense layers at the ouput of the base models, we plan to test other possible fine tunings with variation of the parameters like number of neurons or injection of dropout layers and compare the results to improve the performnce.

Due to time limitation of the project, I couldn’t apply it and train again the models. To improve the application, I plan to refer to a physician for advices on what is needed in this applications as functions and filters.

References

[1] Silva, J., Histace, A., Romain, O., Dray, X., & Granado, B. (2014). Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer. *International journal of computer assisted radiology and surgery*, *9*, 283-293.

[2] Muruganantham, P., & Balakrishnan, S. M. (2021). A survey on deep learning models for wireless capsule endoscopy image analysis. *International Journal of Cognitive Computing in Engineering*, *2*, 83-92.

[3] Muhammad, K., Khan, S., Kumar, N., Del Ser, J., & Mirjalili, S. (2020). Vision-based personalized wireless capsule endoscopy for smart healthcare: taxonomy, literature review, opportunities and challenges. *Future Generation Computer Systems*, *113*, 266-280.

[4] Dheir, I. M., & Abu-Naser, S. S. (2022). Classification of anomalies in gastrointestinal tract using deep learning.

[5] https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918#:~:text=VGG16%20is%20object%20detection%20and,to%20use%20with%20transfer%20learning.

[6] Leonardo, M. M., Carvalho, T. J., Rezende, E., Zucchi, R., & Faria, F. A. (2018, October). Deep feature-based classifiers for fruit fly identification (Diptera: Tephritidae). In *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)* (pp. 41-47). IEEE.

[7] Westphal, E., & Seitz, H. (2021). A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks. *Additive Manufacturing*, *41*, 101965.

[8] https://datasets.simula.no/hyper-kvasir/

[9] https://datasets.simula.no/kvasir/

[10] https://datasets.simula.no/kvasir-capsule/

[11] https://www.tensorflow.org/tutorials/images/data_augmentation

[12] https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e

[13] https://www.l3harrisgeospatial.com/docs/laplacianfilters.html

[14] https://indiantechwarrior.com/canny-edge-detection-for-image-processing/

[15] https://www.emgu.com/wiki/index.php/Main_Page

[16] https://somegenericdev.medium.com/calling-python-from-c-an-introduction-to-pythonnet-c3d45f7d5232

[17] Bashar, M. K., Kitasaka, T., Suenaga, Y., Mekada, Y., & Mori, K. (2010). Automatic detection of informative frames from wireless capsule endoscopy images. Medical Image Analysis, 14(3), 449-470.

[18] https://somegenericdev.medium.com/calling-python-from-c-an-introduction-to-pythonnet-c3d45f7d5232

[19] https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall