Middlesex University London

CST3990 Undergraduate Individual Project

Classification

Neoplastic Lesions

Swin-Transformer

Edoardo Fratantonio
M00702510

Francois Chadebecq

INDEX

ABSTRACT

Identification and thorough excision of premalignant lesions result in a considerable reduction in the risk of dying from colorectal cancer in the long term. Since the implementation of bowel cancer screening programmes across the world, developments in colonoscopy methods and sophisticated endoscopic imaging techniques have resulted in a higher degree of complexity in polyp categorization, providing assistance to a large number of endoscopic surgeons [28]. This research paper will provide an overview of the performance of the Swin-Transformer, on image Classification of Neoplastic lesions, using the Kvasir Dataset. Five labels will be classified: Esophagitis, Polyps, Ulcerative Colitis, Dyed and Lifted Polyps and Dyed Resection Margins, all of which will be explained more in detail throughout the study. A classification software, using Python, of the Swin-T has been developed and it was compared to a Convolutional Neural Network (CNN), in order to evaluate the performance and reliability of the algorithm; and as the paper shows, the Swin-Transformer resulted to be more time-consuming than a convolutional neural network. Furthermore, the Swin-T performed worse than the CNN by a modest margin, reason being that Transformers need large datasets to be trained and report remarkable results. In light of the study's limitations, it is not possible to say with certainty that Swin-Transformer is unreliable for computer vision tasks. However, the architecture of the algorithm shows potential, and in fact, in some scenarios, when the dataset was large enough and the model was trained for enough epochs, Swin-T outperformed various well-known state of the art algorithms for computer vision [17], and this should be taken into consideration for further studies.

1 INTRODUCTION

Cancer of the gastrointestinal tract is the third leading cause of cancer-related mortality (Sharif, et al., 2021). It was discovered that 1.1 million people die from lung cancer every year, 765,000 die from stomach cancer, 525,000 die from colon cancer, and 385,000 die from breast cancer (Sharif, et al., 2021) [1].
Colon cancer is avoidable with the use of efficient screening procedures. Colonoscopy is the most effective method of colon cancer screening and prevention. The purpose of a colonoscopy is to remove colonic polyps before they may progress to colon cancer and cause death. Colonoscopy has shown to be a beneficial prophylactic technique, with the incidence of colorectal cancer having decreased by 30 percent as a result of its use.[22]

The size of a polyp is defined as tiny, if it is less than 5 mm in diameter, small if it is 6 to 9 mm in diameter, and giant if it is 1 cm or more in diameter. Depending on their shape, polyps may be flat, sessile, or pedunculated. Lipomas, carcinoids, and lymphoid aggregates are just a few of the polyps that may develop from the submucosa.[23] The fundamental difficulty with polyps is that they do not manifest themselves in any way, even those who do not identify themselves to be suffering from any form of disease might be living from this too. Despite the fact that persons over the age of 50 are the most at risk for GIT polyps, it is suggested to get a screening done on a regular basis to ensure safety.

In order to tackle this problem, the use of hand-engineered solutions has been attempted in the past, with the typical approach being to extract features and then use a classifier; however, this has been shown to be a poor strategy because it is extremely time-consuming and the characteristics are limited in their representation capabilities, and thus cannot be generalised [2]. The same cannot be said for Vision Transformers, recently, very renowned algorithms for computer vision, which take a different approach to the problem, using Multi-head Self Attention to automate the process, detecting and classifying abnormalities through images and videos.

As artificial intelligence has advanced in recent years, it has become possible to do automated diagnosis or categorization of disorders using medical pictures in a wide range of areas, including gastrointestinal. The area of colonoscopy has seen some encouraging results in recent years, not only in the diagnosis of colorectal polyps, but also in the categorization of colorectal polyps, among other things [24].

The reliance of computer vision, nowadays has arose exponentially, therefore, in some cases, an algorithm can be as accurate as a human eye. Moreover, as the next studies, that will be analysed, will demonstrate, it takes substantially less time to complete and, in certain cases, may be more accurate than other methods indeed. It is recommended that more studies should be conducted, in order to create alternative ways that are as dependable and efficient as possible in all scenarios. Although, the automatic diagnosis of illnesses by the use of computers is a significant, even though as of yet untapped, area of study [18]. Such breakthroughs have the potential to enhance medical practise and modify health-care systems throughout the globe. However, databases containing medical pictures are in short supply, making replication and comparison of techniques almost difficult nowadays.

The classification of Neoplastic Lesions will be the primary focus of this study, using a vision transformer algorithm, the Swin-Transformer, which will be illustrated later on, in the study. Among the topics covered in this paper are present constraints and how to overcome them at the same time, as well as various sorts of approaches produced by experts. After then, there will be a further debate regarding the project's development, which will be reviewed and discussed with its relative results.

2 BACKGROUND AND LITERATURE REVIEW

2.1 PROBLEMS & LIMITATIONS

This research will focus on Wireless Capsule Endoscopy (WCE), a non-invasive procedure that differs from the more well-known White Light Endoscopy technique (WLE). According to Gavriel Iddan's study (Iddan, et al. 2000), a video picture of the WCE (Wireless Capsule Endoscopy) is delivered to wires taped to the body, which enable image capture, and the signal strength is utilized to determine the capsule's location inside the body. The WCE is propelled through the gastrointestinal tract by peristalsis and does not require a pushing force to propel it through the bowel. The video images are transmitted using UHF-band radio-

telemetry to aerials taped to the body, allowing image capture.
The photos are recorded on a portable recorder and saved on a computer. [4] The drawback of this technology is that it does not allow for direct control of the WCE's orientation. Furthermore, as reported in an article, some other issues were encountered, such as the fact that out of 652 consecutive patients, 3 % experienced a battery failure, 11 % experienced an incomplete examination (capsule remained in the Esophagus, delayed gastric emptying, and retained food particles), and the remaining 86 % did not experience any complications (Gavriel Iddan, et al. 2000) [4]. Despite its difficulties, the WCE has continued to improve over time and is a viable alternative to the standard WLE, although for diagnosis purposes only. There are several open problems in the detection and classification of gastrointestinal lesions, which impact the performance and reliability of the system. Tumours differ in shape, size, noise, and colour, and there are similarities between healthy and diseased regions of interest
(ROI), making the diagnostic process extremely difficult [3]. The small intestine is another limitation that must be addressed because of its complexity, meaning that hidden polyps can be difficult to identify. Accessing it can be very difficult, and as a result, it is possible to miss detecting or misclassifying protruding lesions such as polyps, nodules, masses and tumors (Saito et al. 2020)[5].

2.2 LITERATURE REVIEW

A considerable number of investigations have been carried out on the detection and classification of lesions difficulties. Specialists are using a range of ways, each of which has its own set of weaknesses, minor difficulties that are deemed roadblocks to the development of entirely trustworthy solutions.

A range of methodologies has been tested in the following investigations to overcome as many restrictions as possible: Using a different method, Muhammad Sharif and his colleagues attempted a colour-based detection using CEFC and CNN feature extraction and other techniques.

CEFC uses a high level of contrast and stretching methods in the design of the frame.

In order to extract colour information from the images in the dataset, Hue, Saturation, and Variation (HSV) are conducted, after which a weight value is given to define a threshold function for lesion classification. For feature fusions, CNN employs the VGG16 and VGG19 models, which combine a CNN that is 16 and 19 layers deep. Later on, max pooling is conducted, and the Euclidean Fisher Vector is used for feature fusions, respectively.

At the conclusion of the process, it would be categorised using the K-Nearest Neighbor algorithm for classification. As a result, a strategy was developed to overcome the restrictions imposed by the many textures and colours present in the zones of interest. Several tests were carried out, by measuring the 'Accuracy', which evaluates how accurate the model is, the 'Precision', meaning that the model analyzed the same sample several times giving similar results, 'Specificity', which is the ability of a test to correctly exclude individuals who do not have a given disease or disorder, and Sensitivity which instead, it identifies people who have a given disease or disorder. The following findings were presented as a result:

- MGSVM & KNN: ratio training and testing 50:50. Accuracy is 98.6%, sensitivity rate is 0.987, specificity rate is 98.67%, precision rate 98.41%, computed time 131.2s.

- Geometric Features & EFV approach: ratio training and testing 50:50.The maximum classification accuracy of 99.32% is achieved on the KNN classifier. The other performance parameters for KNN, such as AUC, are 1.0, sensitivity rate 1.0, precision rate 99.34% and specificity 100%, computed time 136.2s.

- Conditional entropy: ratio training and testing 50:50.Accuracy results achieved are 99.42%, sensitivity rate at 1.0, specificity at 100%, and the precision rate at 99.51%, computed and achieved in 12.59s on KNN.

When dealing with minor quantities of training and testing data, KNN performed very well; however, SVM performed much better when dealing with large amounts of data. The second drawback identified was that, although VGG16 and VGG19 performed very well in classification accuracy, the fusion process increases the classification time, and as a result, the system cannot be employed in a real-time environment. The authors of this research have left out a significant point: after analysing the findings, they have concluded that the 50:50 ratio training and testing data technique outperforms any other ratio (Sharif et al., 2019)[7].

Another noteworthy method was the Single Shot MultiBox Detector, which aimed to evaluate CNN performance by calculating the area under the receiver operating characteristic curve (AUC) rather than using colour-based segmentation to determine performance. An interesting study was carried out by Hiroaki Saito and his colleagues, in which they used an object detection system called Yolo, which consists of a Deep CNN with 16 or more layers[5] and only takes one shot to detect multiple objects present in an image. Because of its increased speed and high accuracy in object detection algorithm[8], it is much more efficient. As training pictures, researchers utilised a dataset of 30,584 photos with protruding lesions from 292 patients, and as testing images, they used 17,507 images from 93 patients. The dataset included 10,000 images without lesions and 7507 images with protruding lesions, divided into two categories.

In order to categorise polyps, nodules, epithelial tumours, submucosal tumours (SMT), and venous structures, the system would attempt to classify them into five separate groups. Images were input into the SSD architecture during the CNN's training and testing through the Caffe deep learning framework, which was used to feed the images into the architecture. According to Hiroaki, (Saito et al., 2020)[5] the elaboration process was astounding, taking as low as 0.0303 seconds for every picture for a total of 530.462 seconds. Unfortunately, the identification rate for protruding lesions per patient was not excellent, as seen in the results.

It was determined that the polyps in the false-negative "patient" picture were caused by FAP (Familial adenomatous polyposis), a hereditary illness characterised by cancer of the large intestine and rectum. The typical variety of familial adenomatous polyposis is characterised by the development of several noncancerous (benign) growths (polyps) in the colon, which may occur as early as adolescence [9]. The findings of Hiroaki's study demonstrated a sufficient ability to detect lesions on a patient-by patient basis; however, certain limitations made it challenging to achieve the desired results, such as the fact that lesions in the small intestine vary in morphology depending on the disease, making automatic diagnosis difficult. First and foremost, the number of patients included in the test group was relatively limited. The dataset did not include a validation study, and Hiroaki concluded that when the CNN is trained with more photos of polyps and submucosal tunnel(s), the number of false-positive images of normal mucosa would likely grow. It is recommended that this issue be considered throughout additional training and validation, as well as future

investigations, to attain superior diagnostic performance (Saito et al., 2020)[5].

Some other well-known difficulties in recognising and categorising lesions are complicated to overcome and have a high potential to impact the final results, even when the technique utilised is not at fault. Those with a flat morphology or who are in the early stages of their development, for example, have unclear boundaries and low contrast with the background, which makes them more difficult to detect from the background. The WCE (Wireless Capsule Endoscope) is also susceptible to video frame redundancy, motion blur, and visual congestion due to its inability to control the camera's motion.[9]

However, Xiao Jia outlined various approaches for bypassing such restrictions, including the capacity to increase detection accuracy and reliability by modifying the texture and colour of the image and the ability to define and recognise polyps. Additionally, issues about image quality should be avoided by removing duplicate frames, optimising the image, and denoising the image during pre-processing. When the WCE moves slowly, duplicate frames are captured and removed as redundant data. Jia continued his analysis in order to find duplicate images in support of a strategy for assessing "image similarity."

Jia discovered that calculating colour and texture estimates or motion estimation of the WCE would result in duplicate images. Additionally, his team moved the experiment forward to test those assumptions in a real-world environment. They implemented the traditional AlexNet architecture to identify polyps in small bowel WCE images and

achieved a remarkable 99.88 % accuracy.
Xiao introduces the final hypothesis for classifying polyps, which he refers to as "Pixelwise Polyp Segmentation." Pixel-by pixel details are learned and provided for polyp recognition, as opposed to classification models, where they identify whether the image contains a polyp, and detection models, which place a bounding box around the polyp.
Finally, he claims that one of the most significant impediments to future research is the general lack of publicly accessible and high-quality labelled data, which comes from concerns about the private and personal information (Jia et al., 2019)[10].
Nowadays, WCE is frequently used in medical research since it is a fast and convenient way for researching to assist people and contribute to the progress of society as a whole. This is why experts

are attempting to do as much experimentation as possible on this topic in order to reach the radix and eventually conquer these barriers, although they are uncontrollable. A technique for detecting bleeding, tumours, and diseased regions using WE (Wireless Endoscopy) were investigated by Omid and his colleagues, who used colour similarity measures to develop their findings. The researchers employed two colour vector correlation coefficients to determine the degree of similarity between two colour vectors in the RGB colour space. Appropriate classifiers were developed and
implemented based on the similarity coefficients extracted from the colour vector. Kumar exhibited a technique that extracts colours and texture analysis from images using edge features at four distinct angles, colour characteristics based on the CIELUV colour space, and texture features based on Gabor filters. Following the extraction of the features, a support vector machine (SVM) was used to categorise the abnormal areas in the image.
Furthermore, in addition to the Gabor filter, the Susan edge detector was used to extract texture information from polyps, such as the centre of curvature. The SVM classifier was then used to identify the tissue in a frame using the retrieved data.
Lastly, the Multi-Layer
Perceptron (MLP) neural network
was used to evaluate frames.
Three networks were created:
three neural networks to differentiate between normal and tumour areas, typical and illness regions, and standard and bleeding regions in the frames, each with three hidden layers, to distinguish

between normal and tumour regions. Ultimately, they were dissatisfied with the final findings since some barriers were encountered throughout the research, resulting in the discovery of just one or two gastrointestinal disorders, while a wide range of other diseases remained undiscovered. There were no trustworthy findings obtained, particularly in the case of illnesses and tumours due to the unreliable results achieved, especially for diseases and tumours. (Omid, et al. 2019) [11]

In conclusion, several studies were examined, each using a different legitimate technique, but all of them accomplished the ultimate aim of detecting and classifying lesions using WCE (Wireless Capsule Endoscopy). However, even though several limitations prevent us from fully solving the problem, the researchers demonstrate very clearly how to deal with them as well as what kinds of obstacles there are and have been discovered while conducting the study, which can be figured out simply by discussing the findings. However, all of the varied approaches will undoubtedly aid in evaluating a state-of-the-art Detection and Classification methodology, for Neoplastic Lesions, as well as some improvements and precautions identified by the researchers, all of which will be beneficial to this and future studies.

3 REQUIREMENTS & TESTING

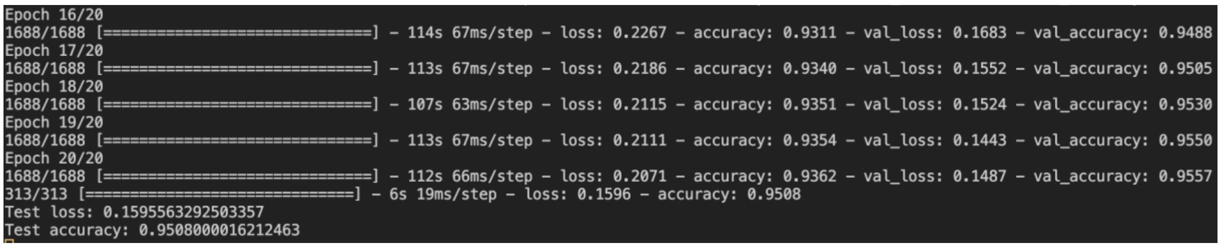
The purpose of the study is based on the classification of neoplastic lesions, as previously stated. However, even though this has already been tried by researchers and even with optimal results, the focus of the study is to attempt a very recent variance of the Vision Transformer family algorithms, Swin-Transformer, and evaluate its performance, as recently the vision transformers have been the subject of research in the field of computer vision, reporting interesting results that call into question the state-of-the-art CNN in certain scenarios and other typical algorithms used for computer vision tasks.

The testing phase has proved challenging to develop, in part because I needed to conduct some preliminary testing before attempting the final version in order to put myself in a position where I could see the performance of the algorithm with a simpler dataset for Image Classification.

To be specific, I attempted to compare the Swin-Transformer with a CNN, a well-known algorithm for image classification, as part of my testing. I used Python to develop the models, taking inspiration from publicly available data science websites [15][16], and I intend to utilise Python throughout the project since it is the language of choice when it comes to machine learning and deep learning. The vast number of libraries required me to set up the whole environment before I could begin working, which included installing all of the libraries, such as Tensorflow, Keras, and Anaconda, amongst others, before I could begin working. In order to obtain a dataset, I used the MNIST digit recognition with ten labels as output, which is composed of 60,000 pictures with a resolution of 28 x 28 pixels and one channel. In addition, I tested the CIFAR100, which is a slightly more complicated dataset since it comes with 100 classes and 60,000 pictures of 32 × 32 and three channels (RGB).

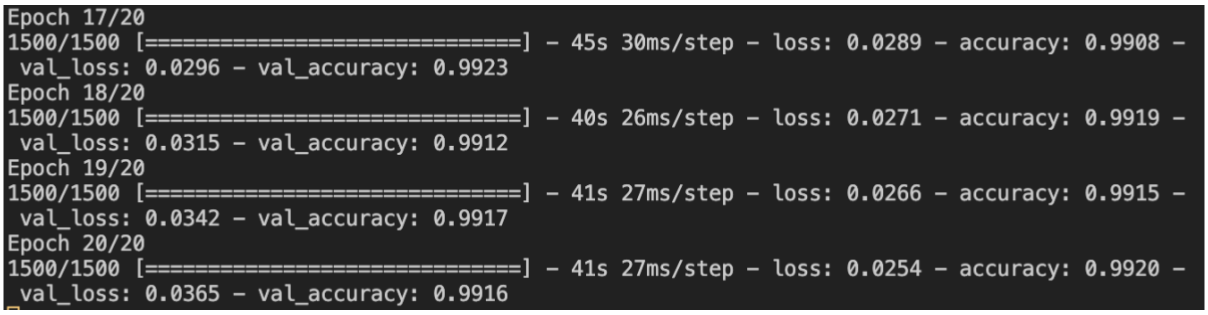
According to [fig 3.1], on the first dataset (MNISTS), CNN outperformed the Swin-Transformer, with an accuracy of 99.16 percent and a loss of 0.03 [figure 3.1]. The "matplot" package in Python was used to create the graph of the model's performance during the whole training process, which was then plotted at the conclusion of each session of the model's performance. As seen in the graph [fig 3.3], we can see that the training loss behaved quite well and that there was no evidence of overfitting in the data. The validation loss, on the other hand, displayed an upward gradient towards the conclusion of the experiment. Figure 3.0 shows that the Swin-T performed with an accuracy of 95.08 percent and a loss of 0.15. As seen in the graph [fig 3.2], both training loss and validation loss began to stabilise rather than decline towards the conclusion of the training, and this may have contributed to the accuracy observed in the study.

Fig 3.0 Swin-T on MNSIST



```
Epoch 16/20
1688/1688 [==============================] - 114s 67ms/step - loss: 0.2267 - accuracy: 0.9311 - val_loss: 0.1683 - val_accuracy: 0.9488
Epoch 17/20
1688/1688 [==============================] - 113s 67ms/step - loss: 0.2186 - accuracy: 0.9340 - val_loss: 0.1552 - val_accuracy: 0.9505
Epoch 18/20
1688/1688 [==============================] - 107s 63ms/step - loss: 0.2115 - accuracy: 0.9351 - val_loss: 0.1524 - val_accuracy: 0.9530
Epoch 19/20
1688/1688 [==============================] - 113s 67ms/step - loss: 0.2111 - accuracy: 0.9354 - val_loss: 0.1443 - val_accuracy: 0.9550
Epoch 20/20
1688/1688 [==============================] - 112s 66ms/step - loss: 0.2071 - accuracy: 0.9362 - val_loss: 0.1487 - val_accuracy: 0.9557
313/313 [==============================] - 6s 19ms/step - loss: 0.1596 - accuracy: 0.9508
Test loss: 0.1595563292503357
Test accuracy: 0.9508000016212463
```

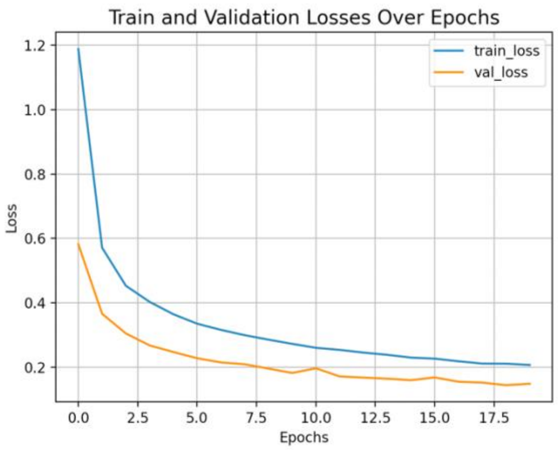
Fig 3.1 CNN on MNSIST

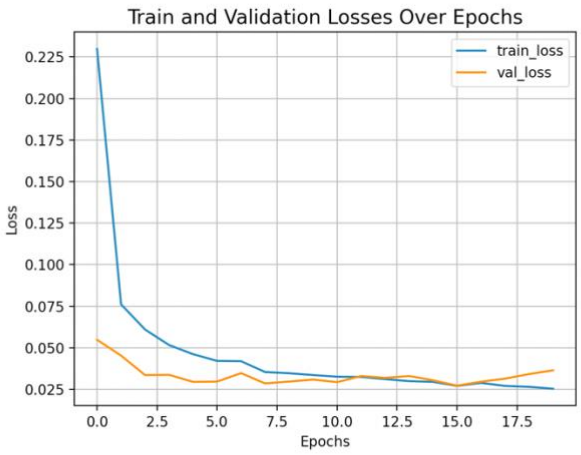


```
Epoch 17/20
1500/1500 [==============================] - 45s 30ms/step - loss: 0.0289 - accuracy: 0.9908 -
 val_loss: 0.0296 - val_accuracy: 0.9923
Epoch 18/20
1500/1500 [==============================] - 40s 26ms/step - loss: 0.0271 - accuracy: 0.9919 -
 val_loss: 0.0315 - val_accuracy: 0.9912
Epoch 19/20
1500/1500 [==============================] - 41s 27ms/step - loss: 0.0266 - accuracy: 0.9915 -
 val_loss: 0.0342 - val_accuracy: 0.9917
Epoch 20/20
1500/1500 [==============================] - 41s 27ms/step - loss: 0.0254 - accuracy: 0.9920 -
 val_loss: 0.0365 - val_accuracy: 0.9916
```

Fig 3.2 Swin-T on MNSIST

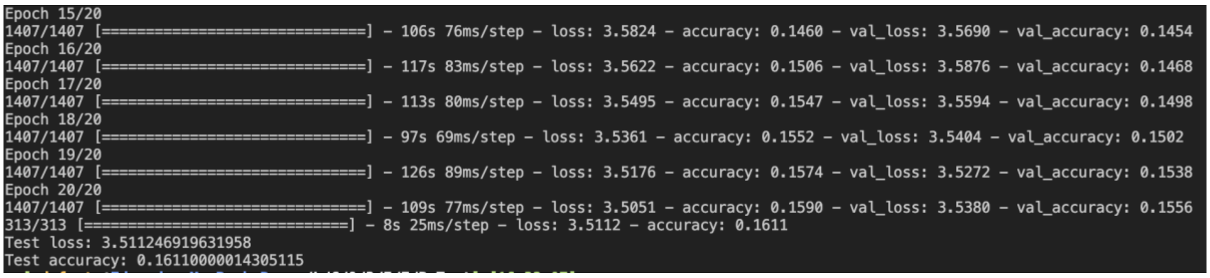
Fig 3.3 CNN on MNSIST





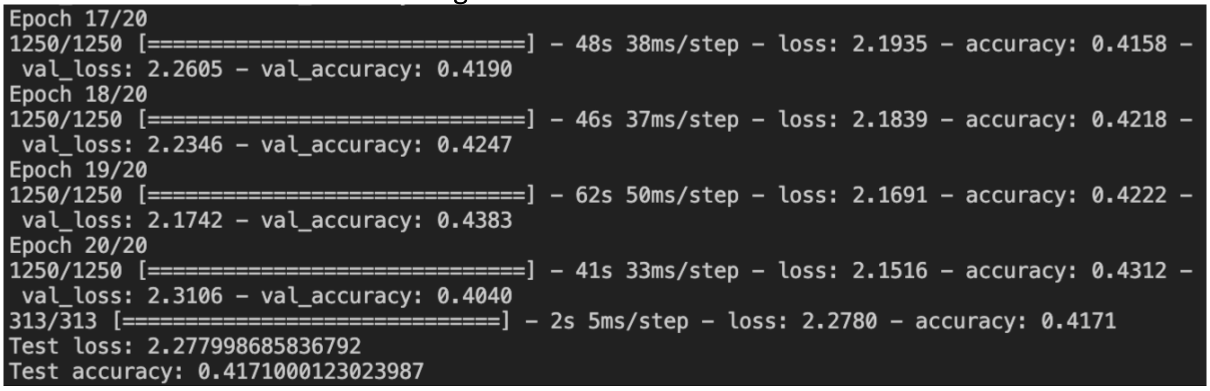
Whereas, in the second dataset, CIFAR100, the CNN outperformed the Swin-Transformer by a significant margin, as it had done in the first, but not as quite different. As seen in Fig. 3.4, the Swin-T obtained an accuracy of 16.11 percent and a loss of 3.51, which is not a very impressive result when compared to the CNN, which achieved an accuracy of 41.71 percent and a loss of 2.27 percent [fig 3.5]. While the Swin-T has a more uniform curve with a little upcurve around the 15th epoch, the graphs below clearly show that it might have performed somewhat better had there been more epochs in the model's development. While looking at the CNN graph, we can see that the validation loss curve had a bit of a struggle and had an upcurve on the final epoch, as well as that the overall line is not homogenous, which might indicate that the model may have been suffered of underfitting data due to the fact that there were 100 classes to classify.

Fig 3.4 Swin-T on CIFAR100



```
Epoch 15/20
1407/1407 [==============================] - 106s 76ms/step - loss: 3.5824 - accuracy: 0.1460 - val_loss: 3.5690 - val_accuracy: 0.1454
Epoch 16/20
1407/1407 [==============================] - 117s 83ms/step - loss: 3.5622 - accuracy: 0.1506 - val_loss: 3.5876 - val_accuracy: 0.1468
Epoch 17/20
1407/1407 [==============================] - 113s 80ms/step - loss: 3.5495 - accuracy: 0.1547 - val_loss: 3.5594 - val_accuracy: 0.1498
Epoch 18/20
1407/1407 [==============================] - 97s 69ms/step - loss: 3.5361 - accuracy: 0.1552 - val_loss: 3.5404 - val_accuracy: 0.1502
Epoch 19/20
1407/1407 [==============================] - 126s 89ms/step - loss: 3.5176 - accuracy: 0.1574 - val_loss: 3.5272 - val_accuracy: 0.1538
Epoch 20/20
1407/1407 [==============================] - 109s 77ms/step - loss: 3.5051 - accuracy: 0.1590 - val_loss: 3.5380 - val_accuracy: 0.1556
313/313 [==============================] - 8s 25ms/step - loss: 3.5112 - accuracy: 0.1611
Test loss: 3.511246919631958
Test accuracy: 0.16110000014305115
```

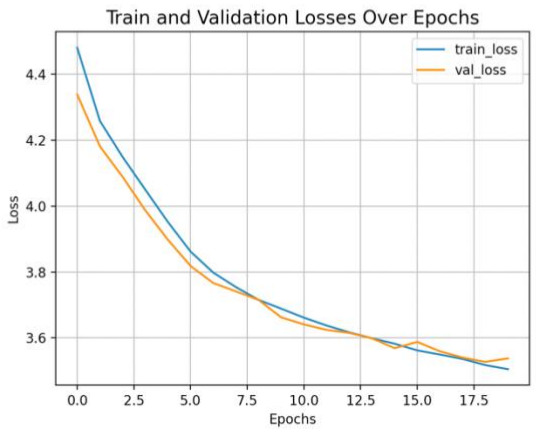
Fig 3.5 CNN on CIFAR100

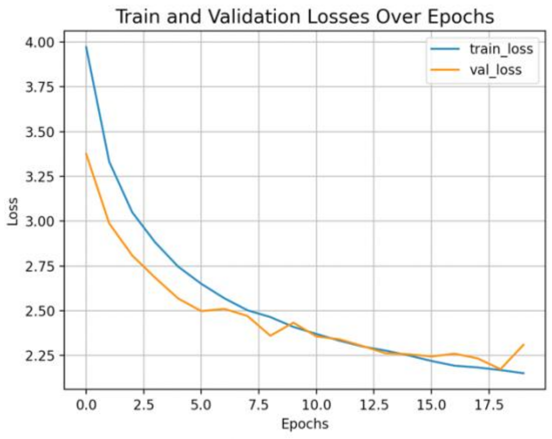


```
Epoch 17/20
1250/1250 [==============================] - 48s 38ms/step - loss: 2.1935 - accuracy: 0.4158 -
 val_loss: 2.2605 - val_accuracy: 0.4190
Epoch 18/20
1250/1250 [==============================] - 46s 37ms/step - loss: 2.1839 - accuracy: 0.4218 -
 val_loss: 2.2346 - val_accuracy: 0.4247
Epoch 19/20
1250/1250 [==============================] - 62s 50ms/step - loss: 2.1691 - accuracy: 0.4222 -
 val_loss: 2.1742 - val_accuracy: 0.4383
Epoch 20/20
1250/1250 [==============================] - 41s 33ms/step - loss: 2.1516 - accuracy: 0.4312 -
 val_loss: 2.3106 - val_accuracy: 0.4040
313/313 [==============================] - 2s 5ms/step - loss: 2.2780 - accuracy: 0.4171
Test loss: 2.277998685836792
Test accuracy: 0.4171000123023987
```

Fig 3.6 SWIN-T on CIFAR100

Fig 3.7 CNN on CIFAR100





The following parameters were used in order to develop the test:

CNN model [3.8]:

Fig 3.8 CNN Model

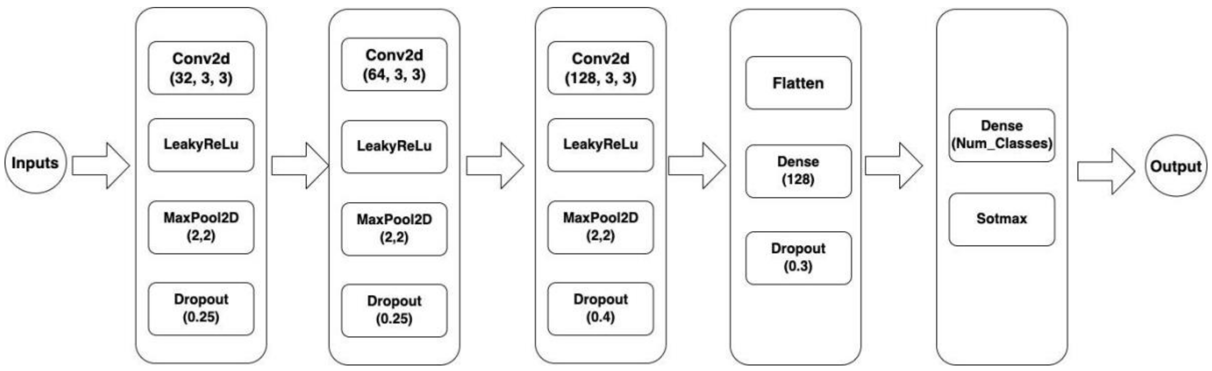
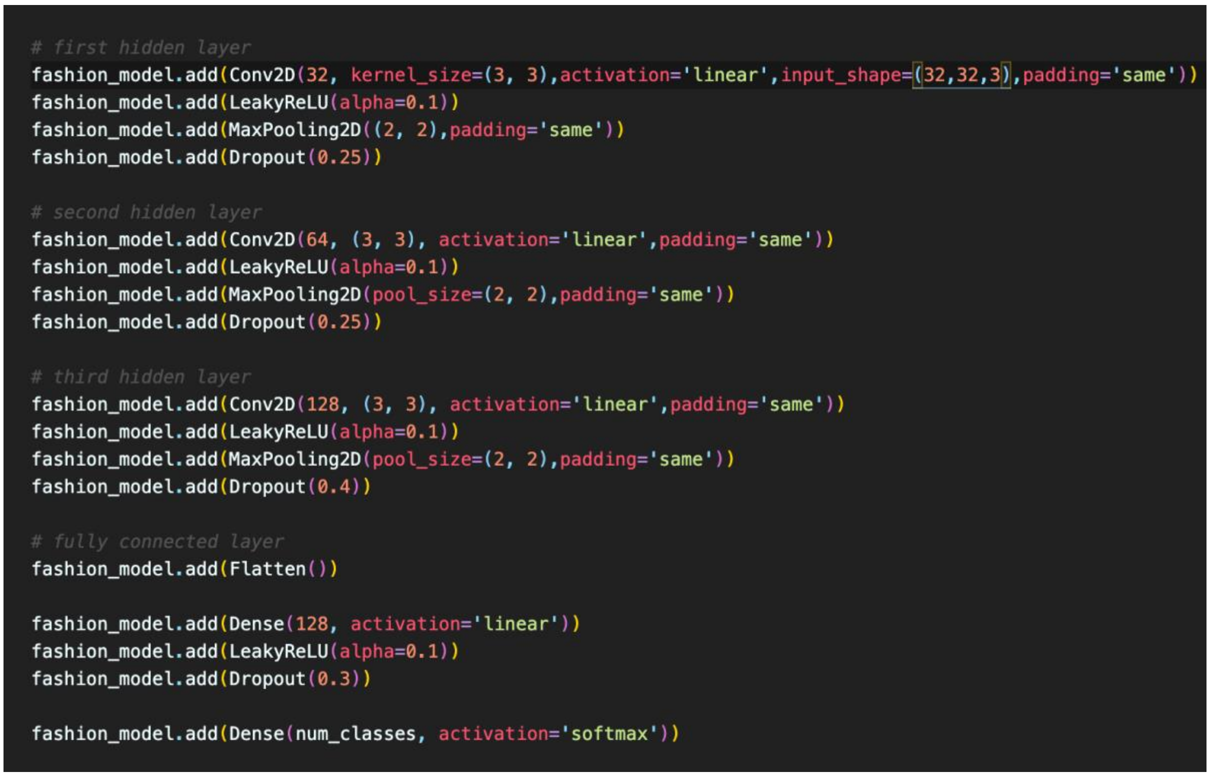


Fig 3.9 CNN Model on Python



```
# first hidden layer
fashion_model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',input_shape=(32,32,3),padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D((2, 2),padding='same'))
fashion_model.add(Dropout(0.25))

# second hidden layer
fashion_model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Dropout(0.25))

# third hidden layer
fashion_model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Dropout(0.4))

# fully connected layer
fashion_model.add(Flatten())

fashion_model.add(Dense(128, activation='linear'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(Dropout(0.3))

fashion_model.add(Dense(num_classes, activation='softmax'))
```

[Fig 3.8 and 3.9] show that the CNN model was composed of three hidden layers, with a Convolutional Layer, an Activation Function, Leaky ReLu was used, Max Pooling 2D and dropout for each layer, in order to avoid overfitting the data, and then a Dense layer, an activation function, and dropout for the final layer. It would then proceed to the last dense layer, where it would provide output with the predicted number of labels and use the activation function SoftMax to ensure that the total number of labels added up to one and, as a result, the percentage of accuracy and loss from labels would be determined.

Swin-T Model[fig4.0]:

Fig 4.0 Swin-T Model

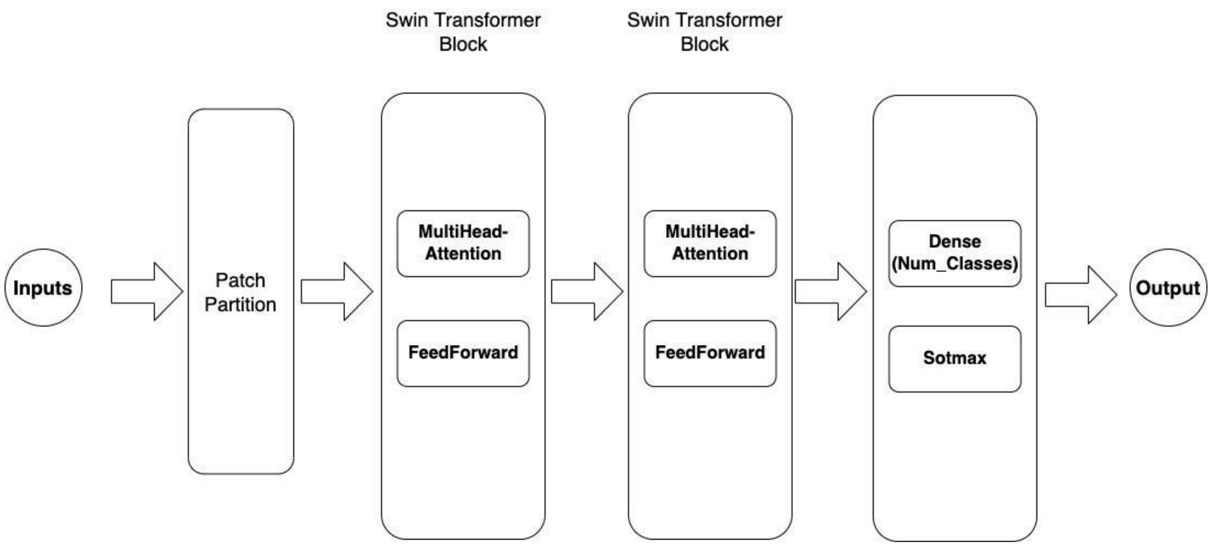
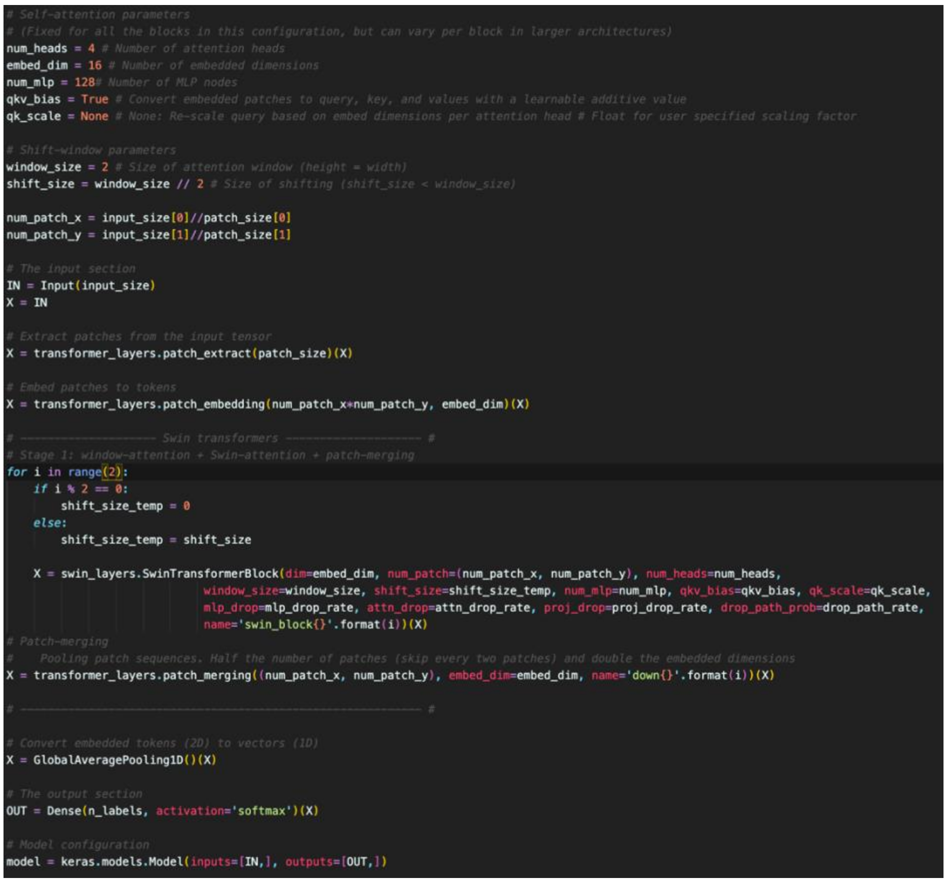


Fig 4.1 Swin-T Model on Python



```
# Self-attention parameters
# (Fixed for all the blocks in this configuration, but can vary per block in larger architectures)
num_heads = 4 # Number of attention heads
embed_dim = 16 # Number of embedded dimensions
num_mlp = 128# Number of MLP nodes
qkv_bias = True # Convert embedded patches to query, key, and values with a learnable additive value
qk_scale = None # None: Re-scale query based on embed dimensions per attention head # Float for user specified scaling factor

# Shift-window parameters
window_size = 2 # Size of attention window (height = width)
shift_size = window_size // 2 # Size of shifting (shift_size < window_size)

num_patch_x = input_size[0]//patch_size[0]
num_patch_y = input_size[1]//patch_size[1]

# The input section
IN = Input(input_size)
X = IN

# Extract patches from the input tensor
X = transformer_layers.patch_extract(patch_size)(X)

# Embed patches to tokens
X = transformer_layers.patch_embedding(num_patch_x*num_patch_y, embed_dim)(X)

# -------------------- Swin transformers -------------------- #
# Stage 1: window-attention + Swin-attention + patch-merging
for i in range(2):
    if i % 2 == 0:
        shift_size_temp = 0
    else:
        shift_size_temp = shift_size

    X = swin_layers.SwinTransformerBlock(dim=embed_dim, num_patch=(num_patch_x, num_patch_y), num_heads=num_heads,
                                         window_size=window_size, shift_size=shift_size_temp, num_mlp=num_mlp, qkv_bias=qkv_bias, qk_scale=qk_scale,
                                         mlp_drop=mlp_drop_rate, attn_drop=attn_drop_rate, proj_drop=proj_drop_rate, drop_path_prob=drop_path_rate,
                                         name='swin_block{}'.format(i))(X)
# Patch-merging
#    Pooling patch sequences. Half the number of patches (skip every two patches) and double the embedded dimensions
X = transformer_layers.patch_merging((num_patch_x, num_patch_y), embed_dim=embed_dim, name='down{}'.format(i))(X)

# ------------------------------------------------------------ #

# Convert embedded tokens (2D) to vectors (1D)
X = GlobalAveragePooling1D()(X)

# The output section
OUT = Dense(n_labels, activation='softmax')(X)

# Model configuration
model = keras.models.Model(inputs=[IN,], outputs=[OUT,])
```

The hyper - parameters used to compare it to the CNN were as follows: the number of attention heads was 4, the number of embedding dimensions was 16, the number of dense layer nodes was 128, and the size of the windows was two pixels. It is possible to change these hyper - parameters depending on the necessities; however, as you add more nodes and heads, the number of parameters increases, and it becomes very slow to run if your laptop does not have a powerful GPU capable of efficiently and quickly processing data.

To sum up the testing phase, I was able to determine that the CNN outperforms the Swin-T in smaller datasets, which is also due to the fact that the Swin-Transformer requires more parameters to be elaborated, as its architecture can be more complex than a CNN, of course, depending on the model parameters chosen as previously mentioned, as well as the architecture of the CNN. When compared to the CNN, the Swin-T requires significantly more time to train. Furthermore, by increasing the parameters of the Swin-T, such as the attention heads and embedding dimensions, the accuracy of the algorithm can be improved significantly. However, this will significantly slow down the entire algorithm, and the time complexity will increase exponentially, making it difficult to work with.

To summarise, according to the paper "An Image is Worth 16x16 Words" (Alexey , et al.,)[17], the Swin-Transformer outperforms the CNN and other popular algorithms in computer vision in the presence of a very large dataset. For example, 100 million images would be sufficient to make the Swin-T perform better than CNN, whereas the CNN's performance would remain constant regardless of the size of the dataset.

4 ANALYSIS AND DESIGN

4.1 Dataset

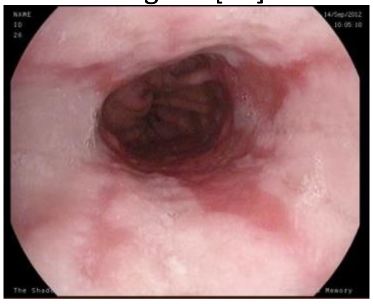
There are several classes of images in the Kvasir dataset [18], each of which has been annotated and confirmed by medical specialists (experienced endoscopists), and each of which shows anatomical landmarks, morphological findings, or endoscopic operations in the gastrointestinal system. The data is acquired via the use of

endoscopic technology at the Vestre Viken Health Trust (VV) in Norway. Furthermore, the photos are rigorously labeled by one or more medical professionals from VV and the Norwegian Cancer Registry before being made available for distribution (CRN). Through cancer research, the CRN contributes to the advancement of cancer knowledge. In addition to being a member of the South-Eastern Norway Regional Health Authority, it is structured as an autonomous institution under the Oslo University Hospital T
rust.[18]

There are 80,000 photos of "720x576" resolution, separated into labels, and eight classes in the Kvasir dataset, however only five of them will be taken into account for this research since the others are not relevant to the study, the five classes will be the following:

- Esophagitis :

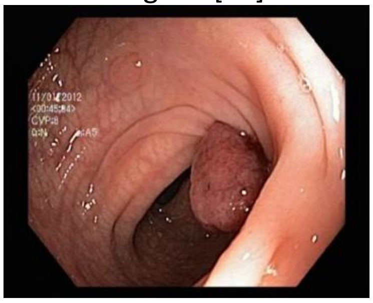
Fig 2.1 [18]



Esophagitisis an esophageal inflammation that manifests as a breach in the esophageal mucosa. The picture [Fig 2.1] depicts an example of red mucosal tongues protruding from the white esophageal lining. This is most usually caused by disorders such as gastroesophageal reflux, vomiting, or hernia, in which stomach acid rushes back into the oesophagus. Computer detection would be very useful for determining severity and automating reporting.[18]

- Polyps :

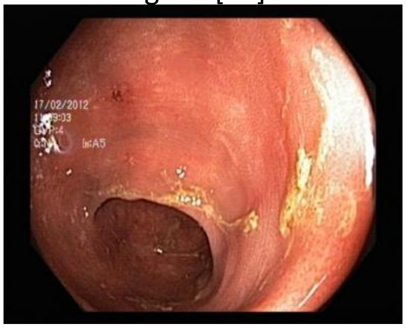
Fig 2.2 [18]



Polyps are lesions inside the colon that may be detected as mucosal outgrows. It is possible to differentiate polyps from normal mucosa by the colour and surface pattern of the polyps, which are either flat, raised, or pedunculate in shape. The majority of intestinal polyps are non-cancerous; however, some have the potential to progress to malignancy. The early detection and excision of polyps are consequently critical in the prevention of colorectal cancer growth. Because polyps are often ignored by clinicians, automated identification would most certainly increase the overall quality of the examination.

- Ulcerative Colitis:

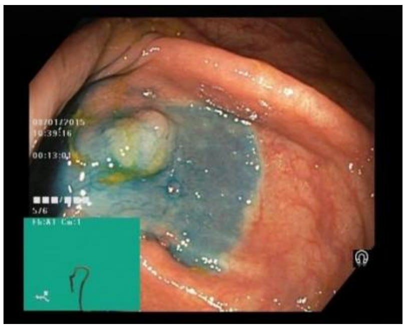
Fig 2.3 [18]



Ulcerative colitis is a chronic inflammatory illness of the large intestine that affects the colon. The condition has the potential to have a significant influence on one's quality of life, and it is mostly diagnosed by colonoscopy examination. The severity of inflammation ranges from none to mild, moderate, and severe, each with its own set of endoscopic characteristics. Swelling and redness of the mucosa are visible, and ulcerations are seen in more severe instances of the condition.

- Dyed and Lifted Polyps:

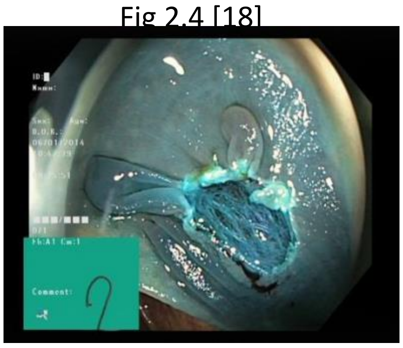
Fig 2.4 [18]



The bright blue polyp borders are plainly apparent against the darker normal mucosa. Additional relevant information connected to automated reporting may include the effectiveness of the lifting as well as the existence of non-lifted regions that may be indicative of malignancy.

- Dyed Resection Margins:

Fig 2.4 [18]



If the polyp has been totally removed, the resection margins are critical in determining if the polyp was completely eliminated. The presence of residual polyp tissue may result in ongoing growth and, in the worst-case scenario, the development of malignancy. Automatic identification of the place of polyp removals is useful for automatic reporting systems as well as computer-aided evaluation of the extent to which the polyp removal has been completed successfully.

These five labels have been selected because they are essential in the colonoscopy field in order to diagnose the presence of polyps, as well as whether or not they have been correctly exported, as well as the other two pathologies that must be treated in order to avoid any other worse risk from occurring.

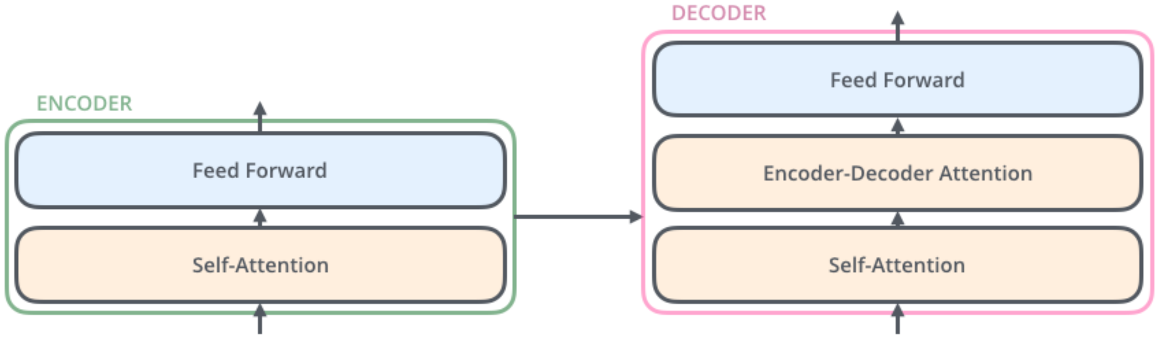
The dataset has been manually reorganised in order to reduce the likelihood of the computer making missteps. The dataset was divided into numbered folders, and within each folder were all of the labelled classes; as a result, I selected the classes and merged them all with the relative class, reducing the size of the dataset from a total of 88,000 images between training and testing to a total of 50,000 images. As a consequence, since the folders had been organised in a certain manner that could not be used by the software, as well as because classes of images that were not required for the study were there, this was essential.

4.2 Transformers

It is necessary to develop an algorithm in order to train a model and analyse its performance allowing to evaluate this research. The Swin-Transformer architecture has been selected because it is a member of the Vision Transformer family, which is well-known in the Natural Language Processing (NLP) area and has a number of advantages. Despite the fact that recent papers have shown that they may be used with computer vision as well, after adapting the architecture, it can provide intriguing results, sometimes outperforming a state-of-the-art CNN in some scenarios.

Typically, transformers implement an attention architecture known as "self-attention," which offers a number of advantages, including the ability to perform parallel computing, which means that it can perform multiple, smaller calculations from a very large problem, and the fact that it does not necessitate the use of a deep network to search for all of the data in one go.

Fig 2.5 [20]



How Transformers work is through the use of an Encoder Block and a Decoder, both of which contain the self-attention layer, as shown in Figure 2.5 [20] and their structure work as follows:

4.3 Encoder

- The word embeddings for the first layer will be used as input. It will be the output of the preceding layer for future layers.
- Within each layer, the multi-head self-attention is calculated first, utilising the layer's inputs as keys, queries, and values.
- The output is passed via a feed-forward network layer. Every patch, is fed into the same feed-forward, which includes two linear transformations followed by a ReLU:
  - Input vector.
  - Linear transformed hidden 1.
  - Linear transformed hidden 2.
  - ReLU output.

4.4 Decoder

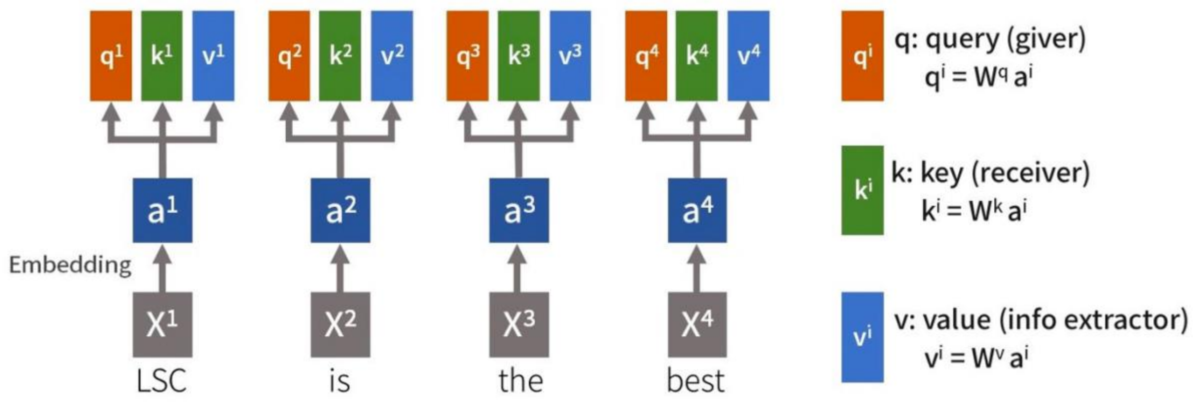
- As input, the first layer's word embeddings will be used. It will be the output of the preceding layer in future levels.
- Within each layer, self-attention is calculated in many heads utilising the layer's inputs as keys, queries, and values (i.e., generated decoder outputs so far, padded for rest of positions).

- The output of the second step is sent to a layer called "multi-head encoder-decoder-attention." Another level of attention is calculated here, this time utilising the second step outputs as queries and the encoder outputs as keys and values.

- Similar to an encoder, the output of the third step is delivered to a position-aware feed-forward network layer.

Query, **Key**, and **Value** [Fig 2.5] are three additional values that will be required in the self-attention layer. Transformers divide the input into patches, which will be embedded to vectors, and therefore for every patch, there will be three additional values, which would be the Query, Key, and Value that will be required in the self-attention layer.[19]

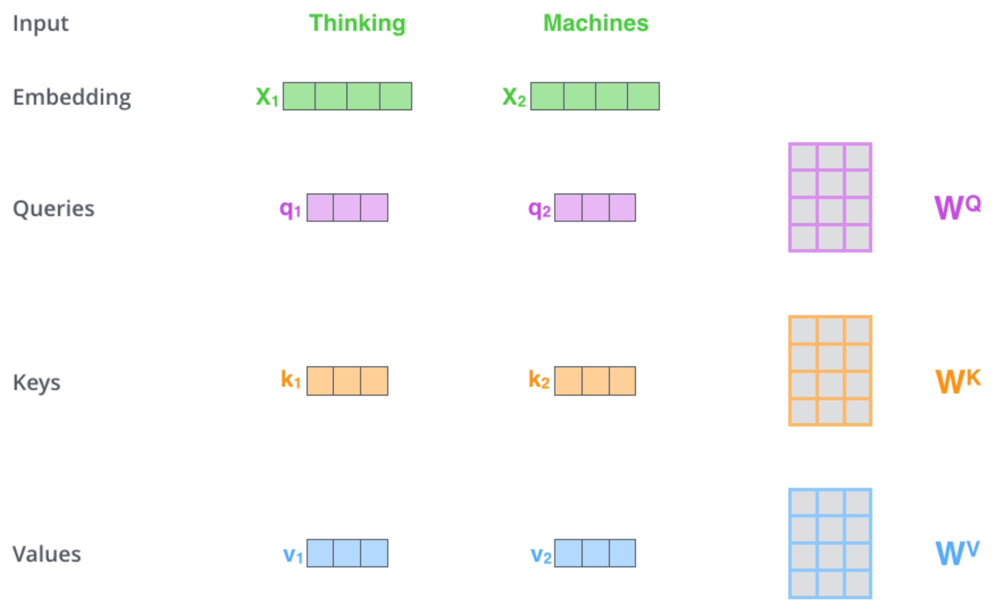
Fig 2.5 [19]



q^1 k^1 v^1 q^2 k^2 v^2 q^3 k^3 v^3 q^4 k^4 v^4
a^1 a^2 a^3 a^4
Embedding
X^1 X^2 X^3 X^4
LSC
is
the
best
q^i q: query (giver) $q^i = W^q a^i$
k^i k: key (receiver) $k^i = W^k a^i$
v^i v: value (info extractor) $v^i = W^v a^i$

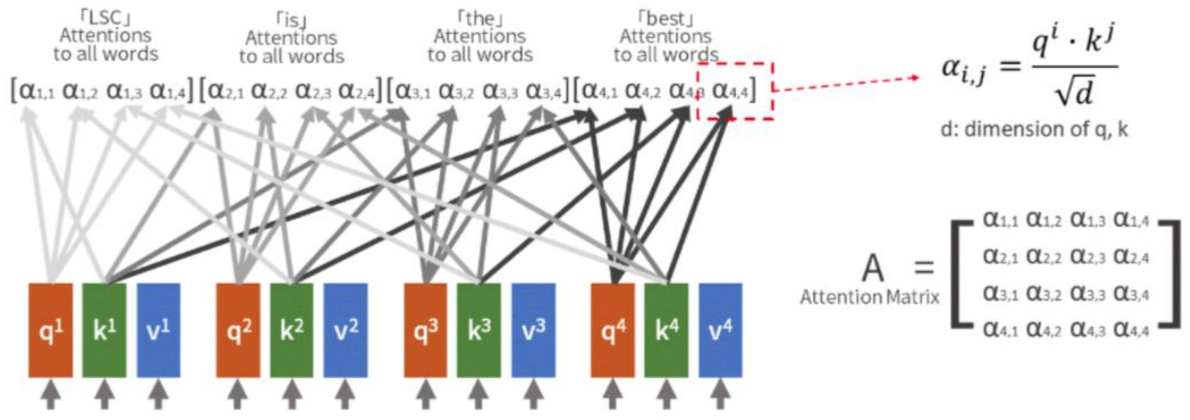
The way they are created, is not very convoluted, as Fig 2.6 illustrates multiplying X1 by WQ, which is the weight matrix, produces “q1”, the query vector associated to that patch, and the process would repeat for Key and Value with their respective weight matrices. The weight matrices are calculated from a pre-training which has been evaluated before starting the actual one, therefore the weights present in those matrices are learnt weights.

Fig 2.6 [20]



The attention is the target to train in a layer, and it is a combination of the three values mentioned previously, with the "Query" acting as the giver and the "Key" acting as the receiver; however, the "Value" is required for information extractors, as it will extract a unique value based on the patch's attentions.

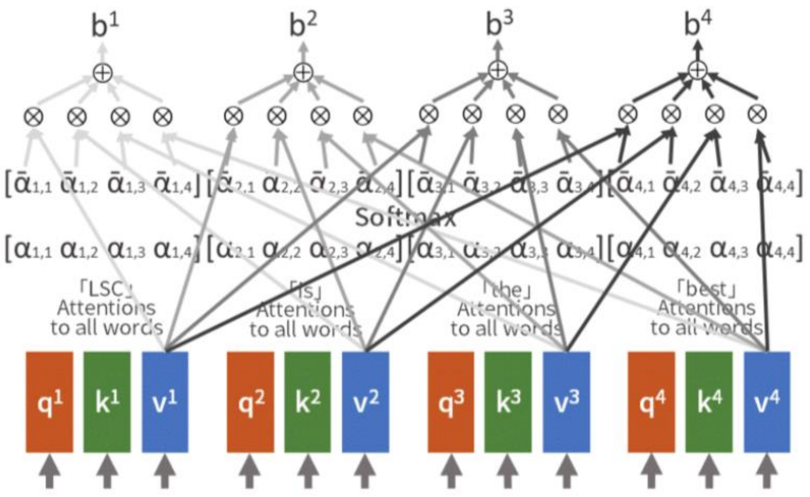
Fig 2.7 [19]



「LSC」 Attentions to all words
「is」 Attentions to all words
「the」 Attentions to all words
「best」 Attentions to all words
$[\alpha_{1,1}\ \alpha_{1,2}\ \alpha_{1,3}\ \alpha_{1,4}][\alpha_{2,1}\ \alpha_{2,2}\ \alpha_{2,3}\ \alpha_{2,4}][\alpha_{3,1}\ \alpha_{3,2}\ \alpha_{3,3}\ \alpha_{3,4}][\alpha_{4,1}\ \alpha_{4,2}\ \alpha_{4,3}\ \alpha_{4,4}]$
$\alpha_{i,j} = \frac{q^i \cdot k^j}{\sqrt{d}}$
d: dimension of q, k
A = Attention Matrix
$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} \end{bmatrix}$
q^1 k^1 v^1 q^2 k^2 v^2 q^3 k^3 v^3 q^4 k^4 v^4

Attention is determined by the following product of Query and Key divided by the square root of the dimensions, of the object being see, as the formula on Fig 2.7 suggests. By supplying a Query to match Key, which is the target patch of attention, every patch draws the reader's attention to itself and all other tokens (patches). Because larger patches result in a greater number of inner products, the square root serves as a variance balancing in this situation. Finally, the matrix of attention A will be produced.

Fig 2.8 [19]



b^1
b^2
b^3
b^4
$[\bar{\alpha}_{1,1}\ \bar{\alpha}_{1,2}\ \bar{\alpha}_{1,3}\ \bar{\alpha}_{1,4}]\ [\bar{\alpha}_{2,1}\ \bar{\alpha}_{2,2}\ \bar{\alpha}_{2,3}\ \bar{\alpha}_{2,4}]\ [\bar{\alpha}_{3,1}\ \bar{\alpha}_{3,2}\ \bar{\alpha}_{3,3}\ \bar{\alpha}_{3,4}]\ [\bar{\alpha}_{4,1}\ \bar{\alpha}_{4,2}\ \bar{\alpha}_{4,3}\ \bar{\alpha}_{4,4}]$
Softmax
$[\alpha_{1,1}\ \alpha_{1,2}\ \alpha_{1,3}\ \alpha_{1,4}]\ [\alpha_{2,1}\ \alpha_{2,2}\ \alpha_{2,3}\ \alpha_{2,4}]\ [\alpha_{3,1}\ \alpha_{3,2}\ \alpha_{3,3}\ \alpha_{3,4}]\ [\alpha_{4,1}\ \alpha_{4,2}\ \alpha_{4,3}\ \alpha_{4,4}]$
「LSC」 Attentions to all words
「is」 Attentions to all words
「the」 Attentions to all words
「best」 Attentions to all words
q^1 k^1 v^1
q^2 k^2 v^2
q^3 k^3 v^3
q^4 k^4 v^4

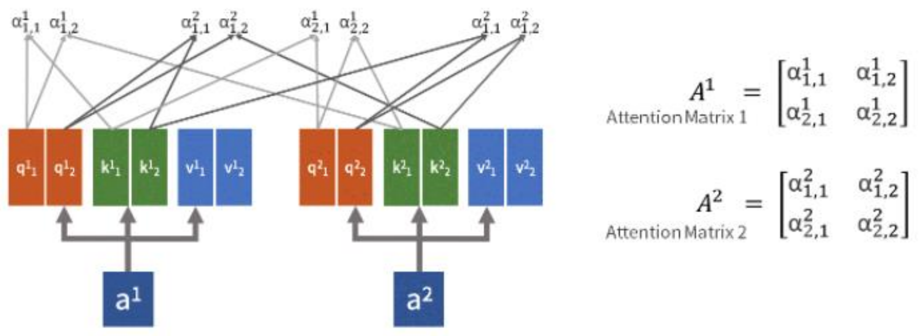
$$b^i = \sum_j \bar{\alpha}_{i,j} v^j$$

In order to build all of the relationships at the conclusion of the attention block, the Value vector will be required. It will be added to the output supplied to it by the previous operation, and all of the values will be shared with all of the other patches, just as they were with the Keys. In the next step, the output **b** is computed by using the SoftMax activation function, which will then convert the result to a percentage value.

To summarise, Query and Key create the relationships, Value summarises the relationships inside, and finishes with an output **b** that contains relationships between input X and all other patches. [Fig 2.8]

Although, in this study I will use another type of attention, called “Multi-Head” Attention, structurally the same as “Self-attention”, however it elaborates more data at the same time. More specifically, multi-head is a feature that allows to construct many Attention Matrixes in a single layer by combining them. Simply doubling the Query, Key, and Value [Fig 2.9] combinations in the Self-Attention Layer results in the calculation of the Attention Matrix on its own. The Self-Attention Layer would generate several outputs if the Multiple-Head option was enabled.

Fig 2.9 [19]



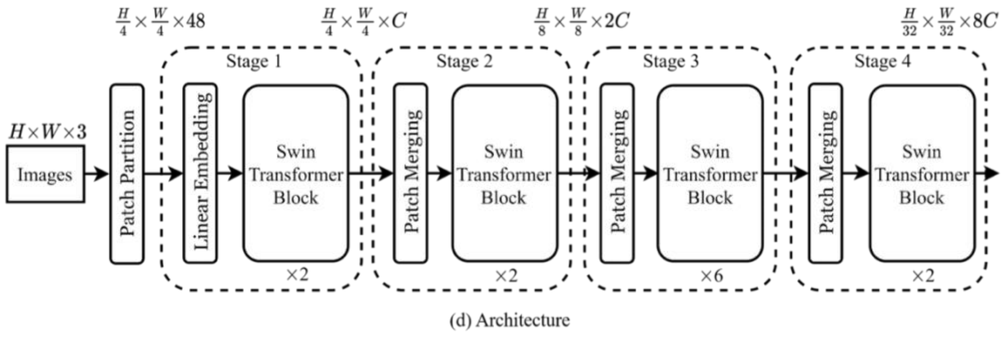
$A^1 = \begin{bmatrix}\alpha^1_{1,1} & \alpha^1_{1,2}\\ \alpha^1_{2,1} & \alpha^1_{2,2}\end{bmatrix}$
Attention Matrix 1
$A^2 = \begin{bmatrix}\alpha^2_{1,1} & \alpha^2_{1,2}\\ \alpha^2_{2,1} & \alpha^2_{2,2}\end{bmatrix}$
Attention Matrix 2
a^1
a^2

4.5 Swin-Transformer

As previously stated, transformers use a sliding window design; with standard vision transformers, all patches maintain their original size throughout the process. Meanwhile, the Swin-Transformer, which originates from **S**hifted **Win**dow, works with smaller patches on the first transformer layer before merging them into larger ones on the deeper transformer layer. [12] The shifted window architecture differs from the conventional sliding window structure in that when the self-attention layer is executed, by confining the computation of self-attention to non-overlapping local windows but still allowing for cross-window connection, the shifted windowing approach may be able to achieve more efficiency than other techniques. Due to the properties of Swin-Transformer, it is appropriate for a broad range of vision applications, including image classification and dense prediction tasks such as object identification and semantic segmentation, among others.

The attention, in the Shifted Window architecture, is spread across Y patches. This means that, unlike the previous self-attention process, where one patch communicates with all other patches, in this case, the one patch communicates with only Y neighbours. This reduces the time complexity, which becomes $O(Y * N),$ instead of being $O(N^2)$.

Fig 3.0 [12]



As shown in Figure 2.0, after the Swin-Transformer Block, there is a "Merging Layer" where the output from the previous block is merged, this layer concatenates the outputs into a bigger patch, and then the Swin-T block repeats itself, the only difference being that with Shifted Window, the attention windows is shifted with respect to the previous layer each time. As a result, patches that couldn't communicate in the previous layer because attention was limited to neighbourhoods of that region are shifted in the next layer, allowing patches to communicate with each other, and then the entire hierarchy repeats depending on the number of layers chosen or when no more merging is possible.[12]

5 DEMONSTRATION & EVALUATION

5.1 Development Phase

The system was developed in Python and made use of deep learning libraries such as Tensorflow, Keras, NumPy, Pandas, SciKit-Learn and the Swin-Transformer library, which is provided by Keras [25]. They will be required in order to obtain the greatest possible result while also ensuring that everything is in place in order to avoid making any errors that may negatively impact the outcome.

A website called "Weight & Biases"[27] was used to evaluate the results; it can be used with Python and, while running the model, whereas training and evaluating, it will track the entire performance of the model, by plotting graphs of the main key points, such as the Number of Epochs, Accuracy, and Loss, among others. It was beneficial to get a sense of how the model was behaving and to determine whether or not there were any incorrect parameters that may have made a difference in the outcome.

The development process followed the GANTT chart, and even though there have been minor delays, which are likely to occur for any project, the plan was structured to account for them, and as a result, the window time for each milestone was generous because it was predicted that problems would arise during the course of the research. Despite this, everything went according to plan.

The performance of the Classification using Swin-Transformer will be evaluated in this research, which will make use of the well-known Kvasir Dataset. An analysis of the behaviour will be carried out in comparison to that of a CNN (Convolutional Neural Network).

The first step in the development section was to set up the whole Python environment, since certain newer versions of the program may create faults or errors if not set up correctly. As previously mentioned, Keras supplies the Swin-Transformer library for Classification, which, although useful, must be applied by the user and, as a result, may be adjusted to the user's demands. This provides the flexibility to modify factors such as the number of heads paying attention, the number of embedding dimensions, and the number of neurons in the Multi-Layer Perceptron component, among other things, in the model.

The Swin-Transformer library implementation has been quite straightforward, as it is composed of essentially two components: the classic Transformer algorithm and then the Shifted Window component, and in order to implement those components, the entire code must be understood beforehand, as it is not a library such as Tensorflow that can be imported in Python, but rather must be hard coded, despite the fact that the Keras documentation is extremely well written.

Following that, the model needed to be designed, and the parameters needed to be determined. Despite the fact that there is no rule of thumb for how the parameters should be set, some understanding is necessary in order to set them up effectively so that the model works as smoothly as possible. The three most important critical principles that are essential to the model are as follows:

- The number of heads paying attention to N patches, due to the fact that multi-head attention is being used.
- The number of embedding dimensions, which means that the less there are, the less accurate the model could be since this parameter along with the number of perceptrons is prone to making the whole model quite heavy.
- The number of perceptrons in a system.

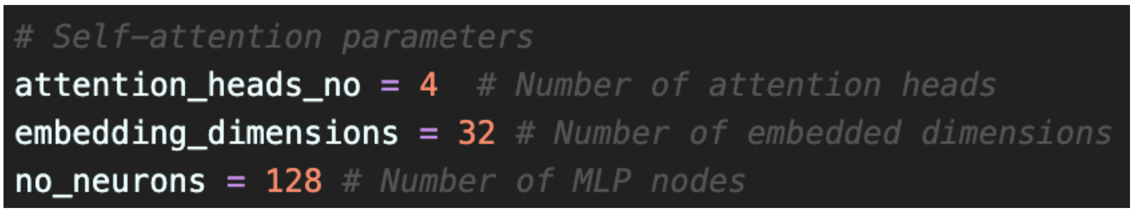
However, there are other parameters that are very important as well, such as multi-layer dropping out rates, attention dropping out rates, and the attention window size. Even though the Swin-Transformer usually starts from a fixed size, which is a "2x2" patch, which will then expand each layer as discussed in previous sections.

The process of importing the dataset into Python, and, of course, rearranging the whole dataset, took longer than expected. Following some research, it was discovered that Python has a module that allows users to provide the path to the dataset and that would automatically recognise the classes, one-hots, and the number of pictures in the dataset. It enables the user to make numerous modifications as possible owing to the fact that it allows the user to input several parameters. The library is called “ImageDataGenerator” [26], and it is already provided by Keras. It was useful for the study as it helped to resize the dataset images being from “720x576” to “512x512”. As a square image is usually preferable to work with. Although, it is typically used for expanding the dataset, even if just by a little amount, it may be useful in certain cases. Nevertheless, can be tricky to give to the model different versions of an image, because sometimes they can lead to making the model misclassing, reason why this study will not use this functionality. Furthermore, this library needs the dataset to be treelike, which is one of the reasons why the dataset had to be reorganised in a form that was acceptable to the library.

5.2 Swin-Transformer Results & Evaluation

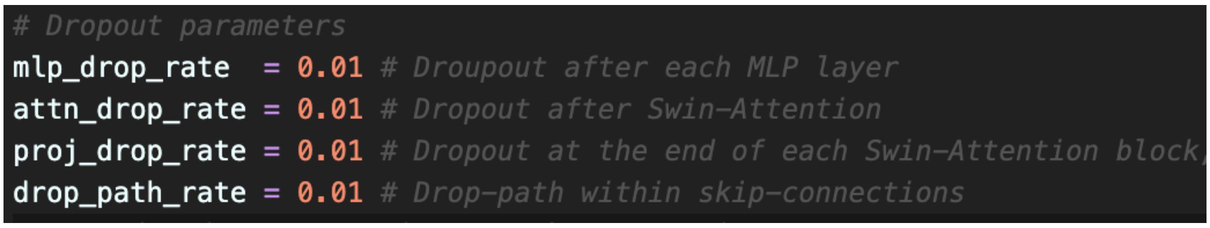
The Swin-Transformer has been set up in the following manner:

Fig 3.1 CNN Self-attention



```
# Self-attention parameters
attention_heads_no = 4  # Number of attention heads
embedding_dimensions = 32 # Number of embedded dimensions
no_neurons = 128 # Number of MLP nodes
```

Fig 3.2 Dropout



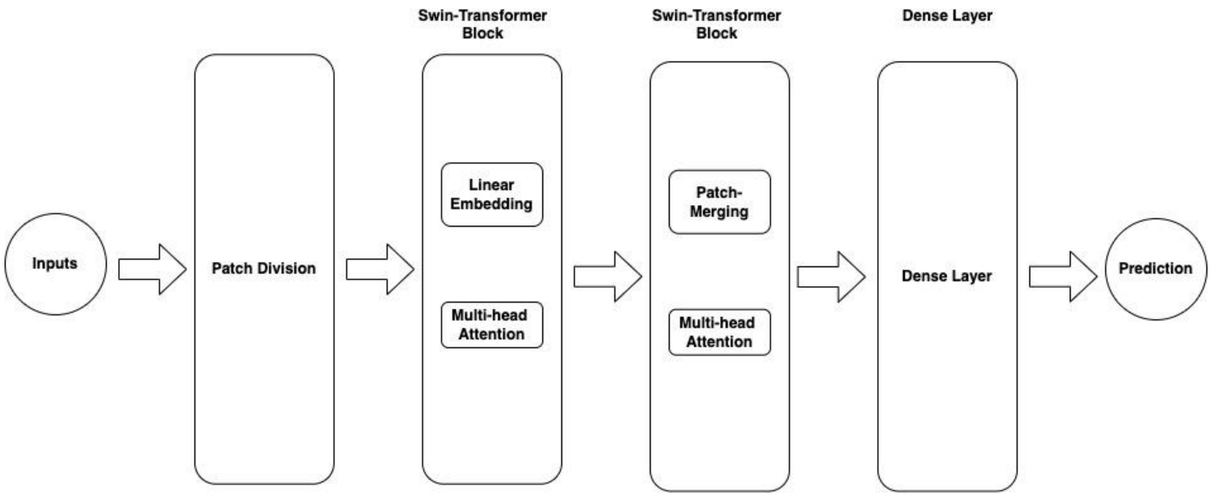
```
# Dropout parameters
mlp_drop_rate  = 0.01 # Droupout after each MLP layer
attn_drop_rate = 0.01 # Dropout after Swin-Attention
proj_drop_rate = 0.01 # Dropout at the end of each Swin-Attention block,
drop_path_rate = 0.01 # Drop-path within skip-connections
```

- Patch-size: 2 x 2

- Number of Epochs: 15
- Batch size: 64

The model has been designed in such a manner that it is not as data-intensive, if not the training phase would have taken an excessive amount of time and would have necessitated the use of a very high-performance GPU, which was not accessible in this occasion. The reason for the modest number of epochs is because the more cycles there are, the more data the computer has to elaborate on and the longer it will take to train the model. The number of heads attention was set to four as Fig. 3.1 shows, in order not to let the machine focus on too many patches, which can lead to misleading results of features extraction, the number of embedding dimensions was set to 32, which will help to extract features from patches, it usually uses standard parameters between 16, 32 or 64, and the number of perceptron was set to 128 so to avoid being as data-intensive, but at the same time being efficient as previously mentioned.

Fig 3.1.1 Swin-T



The way this model of Swin-Transformer will work, as Fig 3.1.1 illustrates, is in the following order: As a first step it will start to divide the image in the “Patch Division”, where it will divide the image in patches, which will be then treated as tokens and as mentioned before patches of “2x2” have been chosen as starting point. After this layer, because the Swin-Transformer does not know the order of the patches, which comes first or last, there is the “Linear Embedding” process, which will enumerate all the patches numerically, thus telling the algorithm the order of the image and how they are related to each other, and just after this process the multi-head attention will begin the process by summing each patch with the relative “Query, Key and Value” vector. Furthermore, the second layer will continue by merging the patches, as explained in the Swin-T section, the “Shifted windowing” architecture comes in, and will merge some of the patches, resulting in bigger patches, in order to let all the tokens, share values and weight each other. In the case of this approach, the multi-head attention starts again its process for the last time. Finally, the Dense Layer starts and as set in the software 128 nodes will wait for inputs, in order to elaborate a prediction for the image just elaborated.

Fig 3.2 Epochs Swin-T [27]

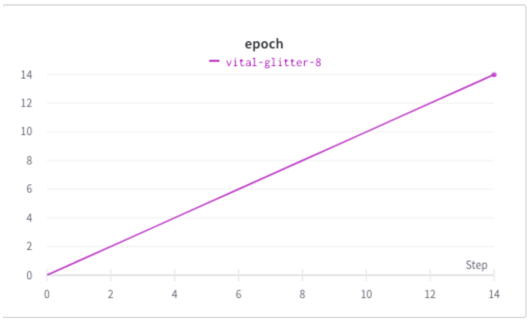
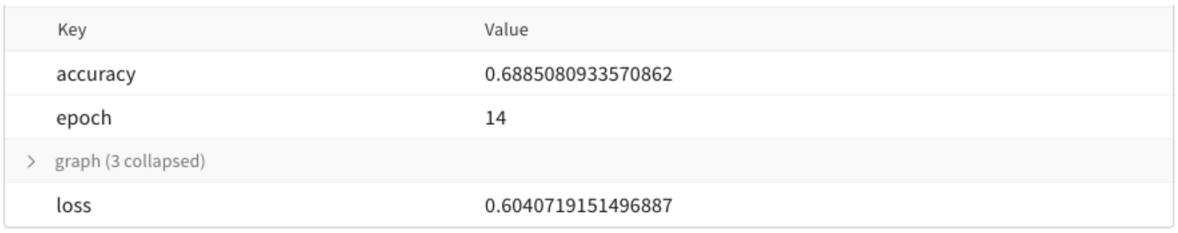


Fig 3.1 Duration Swin-T

Duration	10h 44m 27s

It took a total of 10 hours and 44 minutes to complete the training process, as shown in Fig. 3.1. As demonstrated in the testing phase, the Swin-Transformer is a data-intensive algorithm that requires a significant amount of elaboration. Additionally, because of the attention layers, it can be a time-consuming algorithm to run. In this case, given that the parameters chosen were intended to keep the Swin-T as light as possible while also being as efficient as possible, running the software for more epochs would have taken days, which would not have been possible for the machine to be under such pressure for that length of time. This may have had an impact on the results of the algorithm and should be taken into consideration.

Fig 3.5 Swin-T Stats [27]



Key	Value
accuracy	0.6885080933570862
epoch	14
> graph (3 collapsed)	
loss	0.6040719151496887

Fig 3.3 Swin-T Accuracy [27]

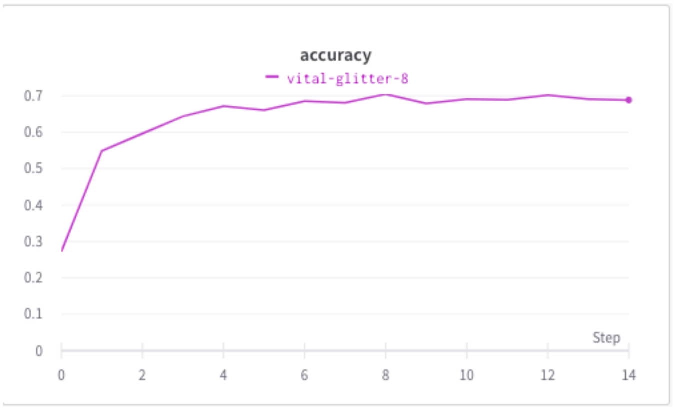
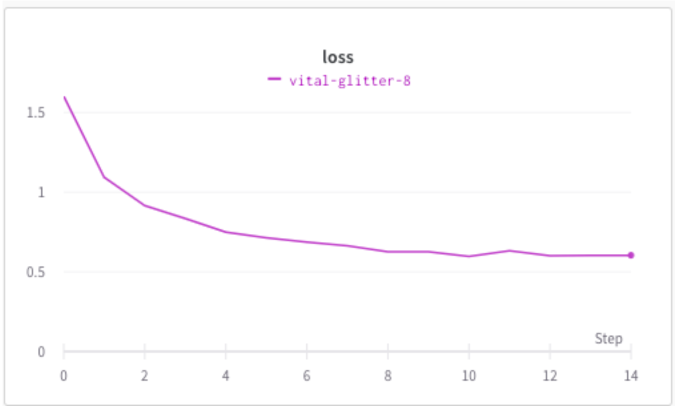


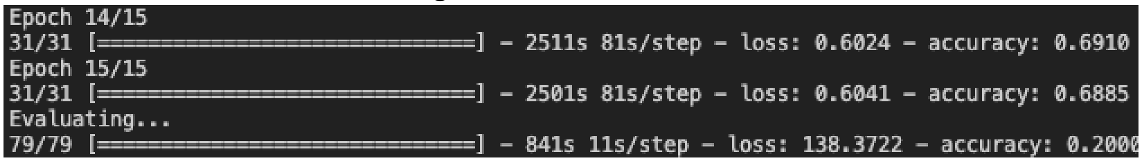
Fig 3.4 Swin-T Loss [27]



In 15 epochs, the model was able to reach an accuracy of 68.85 percent while incurring a loss of 0.60. As shown in Fig. 3.3, the accuracy level increased very quickly in the first two epochs, after which it began to normalise and the curve began to go up and down quite frequently. This is due to the local minima, which the algorithm is attempting to escape in order to find the possible global minima, which means the best accuracy possible, but in order to do so, the accuracy and loss might become worse than in the previous epoch.

After the fourth epoch, the model began to straighten the curve and learn less than it should have, which may have been due to the fact that it was trained for only 15 epochs, which is a small cycle number. Although this may have had an impact on the accuracy level, it is well known that Swin-Transformer requires a large dataset in order to exceed or achieve an interesting accuracy level.

Fig 3.6 Swin-T Evaluation



```
Epoch 14/15
31/31 [==============================] - 2511s 81s/step - loss: 0.6024 - accuracy: 0.6910
Epoch 15/15
31/31 [==============================] - 2501s 81s/step - loss: 0.6041 - accuracy: 0.6885
Evaluating...
79/79 [==============================] - 841s 11s/step - loss: 138.3722 - accuracy: 0.2000
```

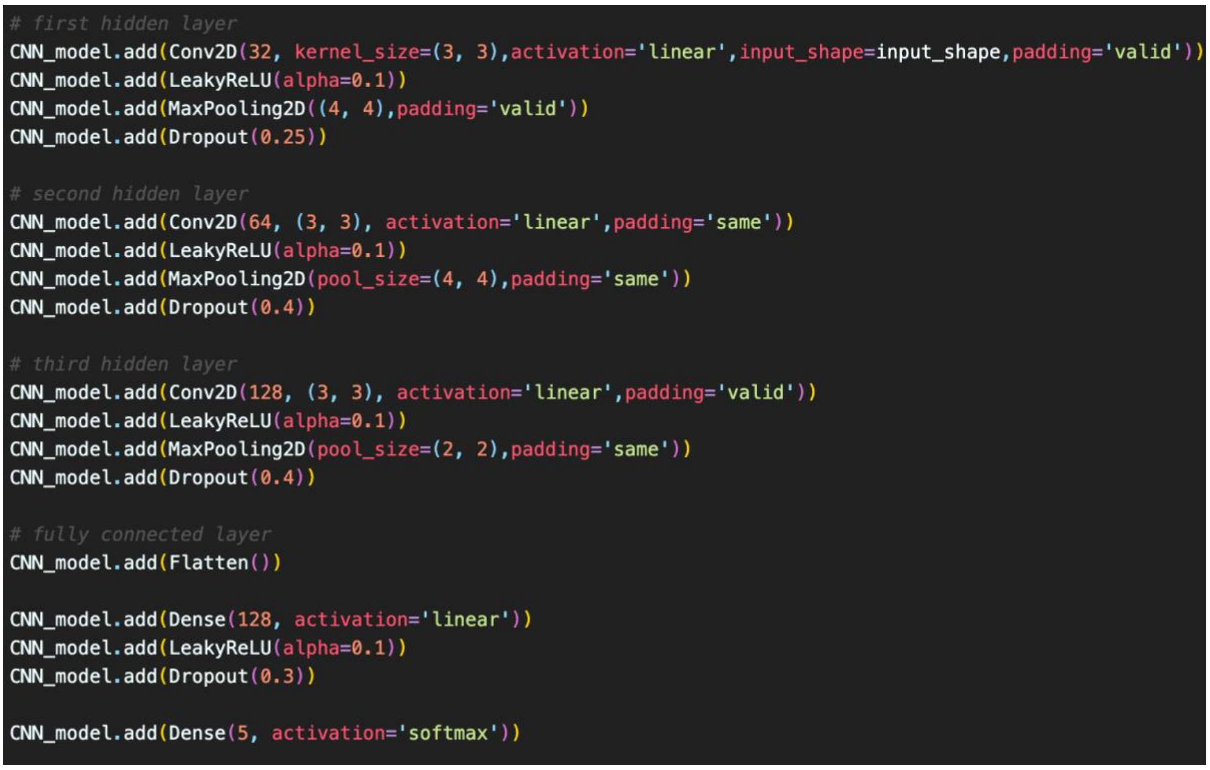
Fig 3.5 proves the point as the model did not perform that well in the training phase and performed really bad with unseen data, indicating that it suffered of underfitting, meaning that the data was not enough for the model. This is due to the fact that attention takes small improvements at a time, and hence requires a large amount of data in order to understand where the attention should be focused. The dataset could have been more larger, even though it is very difficult to find a public dataset large enough of neoplastic lesions these days and this has undoubtedly affected the performance of the Swin-Transformer as Alexey Dosovitskiy and colleagues reported in the “Image is worth 16x16” paper, when a dataset of at least 100 million images is available, the Swin-T outperforms well known algorithms in computer vision, however as previously mentioned is very challenging to find one publicly.

Furthermore, as Fig 3.4 illustrates the loss performance of the model during the training phase, and as it clearly shows, that after the sixth epoch, the model started to normalise the loss and not improve by as much, meaning that more epochs would certainly made the model perform better, even though by not as much, the graph suggests that in order to make the model achieve interesting results, it would have needed a large amount of epochs, which would have led the computer to be under pressure for an excessive amount of time. Even though the performance during the evaluation phase did not give good results, it still needs to be taken into account that Transformers in general, do need an extraordinary number of images in order to start seeing competitive results in computer vision tasks.

5.2 CNN Results & Evaluation

In order to understand the performance of a Swin-Transformer, it is good practise to compare it to a well-known algorithm for computer vision, this Is the reason why a model of Convolutional Neural Network was built, in a specific manner to be comparable. There have been used libraries such as Tensorflow and Keras to build the model and also the library called “wandb” which stands for “Weight and Biases” has been used for making graphs out of the model training phase, and for importing the dataset, the same library used for the Swin-Transformer has been used, which is the “ImageDataGenerator”.

Fig 3.7 CNN Code



```
# first hidden layer
CNN_model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',input_shape=input_shape,padding='valid'))
CNN_model.add(LeakyReLU(alpha=0.1))
CNN_model.add(MaxPooling2D((4, 4),padding='valid'))
CNN_model.add(Dropout(0.25))

# second hidden layer
CNN_model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
CNN_model.add(LeakyReLU(alpha=0.1))
CNN_model.add(MaxPooling2D(pool_size=(4, 4),padding='same'))
CNN_model.add(Dropout(0.4))

# third hidden layer
CNN_model.add(Conv2D(128, (3, 3), activation='linear',padding='valid'))
CNN_model.add(LeakyReLU(alpha=0.1))
CNN_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
CNN_model.add(Dropout(0.4))

# fully connected layer
CNN_model.add(Flatten())

CNN_model.add(Dense(128, activation='linear'))
CNN_model.add(LeakyReLU(alpha=0.1))
CNN_model.add(Dropout(0.3))

CNN_model.add(Dense(5, activation='softmax'))
```

The model was built in order to evaluate images of modest dimension, 512 x 512 pixels to be exact, reason why the model structure has three convolutional layers, in order to extract as many features as possible, reason why the kernel size was set to 3x3 in order to keep the have an image with features extracted and not distorted. Because the picture is still of large dimensions, the padding in the first layer was not applied. For this reason, the Max Pooling was kept at 4 x 4, which ensured that the image size would not be reduced by a significant amount. As a function activation the Leaky ReLu has been selected, because it does not suffer from gradient dissolving and together with the Dropout, which will deactivate randomly 25 percent of the neurons, so to avoid overfitting the model.

In the second layer 64 neurons were set and what is changed from the previous layer is the dropout which would deactivate the 40 percent of neurons in this stage and also the MaxPooling has decreased to 2 x 2 and the padding has been included in order to prevent the image size from being completely wiped out. Similarly, the neurons in the third layer have increased again to a total of 128 perceptrons, which will be used to extract as many features as possible. However, the dropout and MaxPooling parameters have remained unchanged, as has the kernel size, which remains a 3x3 size, in order to avoid distorting the image too much and keeping the extrapolated features in place.

Fig 3.8 CNN Accuracy

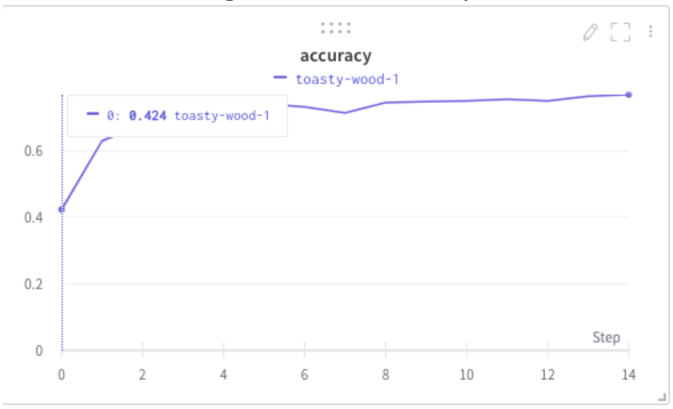
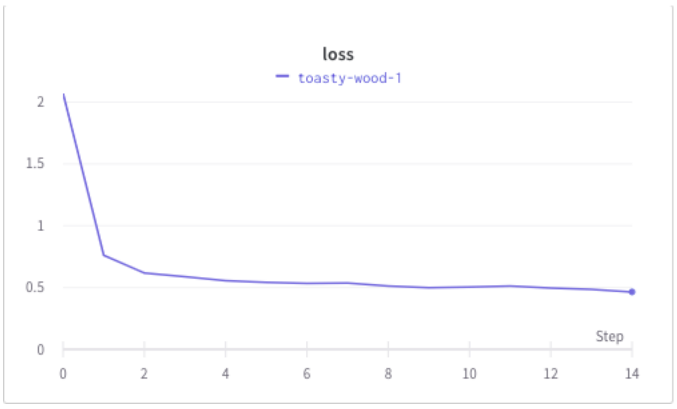


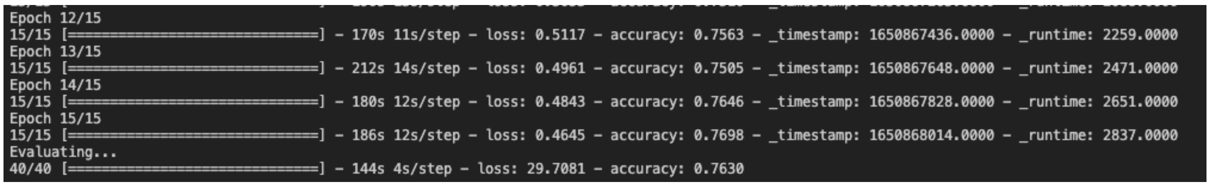
Fig 3.9 CNN Loss



As with the CNN, the accuracy and loss acquired in fifteen epochs are 76.97 percent and 0.46 percent, respectively, for the findings. Though, when compared to each other in terms of performance, the CNN outperformed the Swin-Transformer by a significant margin, they both behaved in the same way when it came to learning new features, at the beginning of the training phase, the model would learn new features very quickly before starting to normalise and becoming very uniform, by remaining on the same sort of level throughout the remainder of the training process. This suggests that with more epochs, which could have been carried on, the accuracy level would have become very interesting, however, this study is focussing on the Swin-T performance, reason why the epochs were kept at fifteen.

Furthermore, the CNN accomplished this performance by finishing the training in just 49 minutes, which surpasses the Swin-T by a significant margin of time. This is owing to the attention layers required for dealing with more data and more processes, which is more elaborated to do with the Swin-T.

Fig 4.0 CNN Evaluation



```
Epoch 12/15
15/15 [==============================] - 170s 11s/step - loss: 0.5117 - accuracy: 0.7563 - _timestamp: 1650867436.0000 - _runtime: 2259.0000
Epoch 13/15
15/15 [==============================] - 212s 14s/step - loss: 0.4961 - accuracy: 0.7505 - _timestamp: 1650867648.0000 - _runtime: 2471.0000
Epoch 14/15
15/15 [==============================] - 180s 12s/step - loss: 0.4843 - accuracy: 0.7646 - _timestamp: 1650867828.0000 - _runtime: 2651.0000
Epoch 15/15
15/15 [==============================] - 186s 12s/step - loss: 0.4645 - accuracy: 0.7698 - _timestamp: 1650868014.0000 - _runtime: 2837.0000
Evaluating...
40/40 [==============================] - 144s 4s/step - loss: 29.7081 - accuracy: 0.7630
```

After passing through the training phase, the CNN with unseen data performs well and stays consistent with its training, achieving an accuracy of 76.30 percent and a loss of 29.70 in the evaluation phase.

As evaluated by the Swin-Transformer, this dataset did not suffer from underfitting data; in fact, even though the scores obtained during the training and testing phases were not truly remarkable, the statistics and behaviour of the model's performance suggest that this dataset would have been sufficient to produce interesting results and proves the fact of being a relatively small dataset for the Swin-T.

6 CONCLUSION

To conclude, the results of this investigation demonstrated the effectiveness of the Swin-Transformer in the classification of neoplastic lesions. After looking at the algorithm's architecture, which is the same as those of the "Vision Transformers" family, with the main variation being that it applies "Shifted Windowing" attention in the attention layers rather than the "Sliding Windowing" method in the attention layers. This has been shown to be more efficient and less difficult in terms of time management. It is quite regrettable that there have been limits in terms of confirming the efficacy of such an algorithm, since this is extremely important. The dataset has been the first restriction, since there are currently no exhaustive datasets published, at least not publicly available databases. A thorough examination of the system and evaluation using graphs revealed that the model had underfitting data, which means that the data from the dataset was insufficient to allow the model to train effectively and extract sufficient features to determine where the attention should be focused. That this was the case was shown by the fact that, in contrast to the CNN, the model performed very poorly with unseen data, but in the training phase it did rather well with a small number of training epochs.

Due to the lack of a high-performance GPU and the fact that the model was trained on a CPU, the performance of the model could not be structured in a complex manner. As a result, the number of training epochs and the hyperparameters were set in a specific manner so that the model could still perform effectively while also being careful not to stress the CPU for an excessive amount of time. The reason for this is that if the model had been trained for more epochs, the more the better, it would have learned more and extracted enough features to achieve modest results, as well as with unseen data, even though this is more due to the fact that the dataset was not large enough, it would have performed better.

Although, this paper has shown how powerful is the structure of Transformers in general, it has been very interesting exploring and understand how attention layers work and their architecture, which is very different from a Convolutional Neural Network and other renowned algorithms like Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Support Vector Machines (SVM) among others, as the attention layer tries to focus its attention indeed only into the object in question, but it does it in very specific manner, which covers all the edges and areas of just the object itself, leaving outside what it is near it, the only downside for the moment is that it requires new data in order to understand where to focus its attention. Furthermore, the Shifted window architecture which differs from the sliding window as in the first, the patches get merged each layer when getting out from the Swin-Transformer block as long as there is no more to merge, so allowing the new data sharing between neighbour patches, in this way every patch can communicate with another one allowing the transformer understand the correlation with each one, and this is what makes it very well structured.

It is important to note that the results presented here, given the limitations just discussed, may not reflect the real-world potential of the algorithm, even though they can be used as a reference point to better understand the algorithm's behaviour, which should be further investigated with more powerful machines, which will undoubtedly get the most out of the Swin-Transformer; in this case, it has not behaved better than a Convolutional Neural Network, even though with further studies it may do so.

This study has assisted me in comprehending the fundamentals of conducting research, analysing data and articles with an analytical perspective, and identifying information that would be used for future research-based projects, which was beneficial for me. If I were to start over with a new project, I would definitely conduct a more thorough literature review at the early stage to gain a better understanding of the problem and ideas on how to approach it. This can be useful in discovering new approaches or small details that other papers are missing, which I did for this paper as well, but I would have liked to have discovered even more in order to be as thorough as possible in the first place. Along with this, I would ensure that I had all of the equipment necessary from the very beginning of the project and that I was familiar with them, so that I would not have to struggle or push back milestones. Overall, this paper was indeed a challenge, but it allowed me to discover the world of Vision Transformers, which, for many, will be the future of computer vision, and even though it has been demonstrated to be a time-consuming algorithm, with more research, further studies, and tests, it could truly bring the computer vision field to the next level and may be a turning point for many tasks.

7 Appendices

In order to run the software, which has been developed in Visual Studio Code, even though, a text editor it is not essential to run the software, as long as the dataset is in the same folder. It will then require to have installed, on the machine, at least Python 3.7, and then install the following libraries:

- Tensorflow 3.2.0
  It will be needed in order to train and construct the model architecture.
  (pip install tensorflow=2.8.0)

- Keras 2.8.0
  It is needed alongside with Tensorflow, as they work together, and they share similar functionalities.
  (pip install keras=3.2.0)

- Wandb (Facultative)
  This is needed in order to create the graphs for evaluating the model and see the performance of each epoch.
  (pip install wandb)

- Vision transformer and Swin Transformer library from Keras, which will be together with the code, which needs to be inside the same folder of the actual code in order to use it.

After installing all these libraries, the software should be ready to be used, even though the dataset path must be specified, which can be done by setting the "**train_it"** and “**test_it”** variables. After all this, the software is ready to be launched, and the model will start to train as soon as the environment is all set.

8 REFERENCES

[1] Sharif, M., & Khan, M. A. (n.d.). *Deep CNN And Geometric Features-based Gastrointestinal Tract Diseases Detection And Classification From Wireless Capsule Endoscopy Images.* https://www.tandfonline.com/doi/abs/10.1080/0952813X.2019.1572657.

[2] Wireless Capsule Endoscopy: A New Tool for Cancer Screening In the Colon With Deep-Learning-Based Polyp Recognition. (n.d.). Wireless Capsule Endoscopy: A New Tool for Cancer Screening in the Colon With Deep Learning-Based Polyp Recognition. https://ieeexplore.ieee.org/abstract/docu ment/8903282.

[3] Computer-aided Small Bowel Tumor Detection for Capsule Endoscopy - ScienceDirect. (2011, February 24). Computer-aided small bowel tumor detection for capsule endoscopy - ScienceDirect. https://www.sciencedirect.com/science/ar ticle/pii/S0933365711000042.

[4] M., A., & S. (n.d.). *Problems, Complications And Failures Of Wireless Capsule... : Official Journal Of the American College Of Gastroenterology | ACG.* LWW. https://journals.lww.com/ajg/Fulltext/2007/09002/Problems,_Complications_and_F ailures_of_Wireless.1130.aspx.

[5] Saito, H., Aoki, T., & Ayoama, K. (2020, February 19). *Automatic Detection And Classification Of Protruding Lesions In Wireless Capsule Endoscopy Images Based On a Deep Convolutional Neural Network.* Automatic detection and classification of protruding lesions in wireless capsule endoscopy images based on a deep convolutional neural network - ScienceDirect. **https://www.sciencedirect.com/science/article/pii/S0016510720301322.**

[6] Oh, C. ., Kim, T., & Cho, Y. K. (n.d.). *Convolutional Neural Network-based Object Detection Model To Identify Gastrointestinal Stromal Tumors In Endoscopic Ultrasound Images.* Convolutional neural network-based object detection model to identify gastrointestinal stromal tumors in endoscopic ultrasound images.

https://onlinelibrary.wiley.com/doi/full/10.1 111/jgh.15653.

[7] Sharif, M., & Khan, M. A. (n.d.). Deep CNN And Geometric Features-based Gastrointestinal Tract Diseases Detection And Classification From Wireless Capsule Endoscopy Images. Deep CNN and geometric features-based gastrointestinal tract diseases detection and classification from wireless capsule endoscopy images. https://www.tandfonline.com/doi/abs/10.1 080/0952813X.2019.1572657.

[8] Khandelwal, R. (2019, November 30). SSD : Single Shot Detector for Object Detection Using MultiBox. Medium. https://towardsdatascience.com/ssd single-shot-detector-for-object-detection using-multibox-1818603644ca.

[9] Familial Adenomatous Polyposis: MedlinePlus Genetics. (2020, August 18). Familial adenomatous polyposis: MedlinePlus Genetics. https://medlineplus.gov/genetics/conditio n/familial-adenomatous-polyposis/.

[10] Jia, X. (n.d.). Wireless Capsule Endoscopy: A New Tool for Cancer Screening In the Colon With Deep Learning-Based Polyp Recognition. Wireless Capsule Endoscopy: A New Tool for Cancer Screening in the Colon With Deep-Learning-Based Polyp Recognition. https://ieeexplore.ieee.org/abstract/docu ment/8903282.

[11] Maghsoudi, O. H., Alizadeh, M., & Mirmomen, M. (n.d.). A Computer Aided Method To Detect Bleeding, Tumor, And Disease Regions In Wireless Capsule Endoscopy. A computer aided method to detect bleeding, tumor, and disease regions in Wireless Capsule Endoscopy. https://ieeexplore.ieee.org/abstract/docu ment/7846852.

[12] Liu, Z., & Lin, Y. (2021, August). Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. https://arxiv.org/abs/2103.14030.

[13] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, Pål Halvorsen, Kvasir: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection, In MMSys'17 Proceedings of the 8th ACM on Multimedia Systems Conference (MMSYS), Pages 164-169 Taipei, Taiwan, June 20-23, 2017.

https://datasets.simula.no/kvasir/

[14] EndoVis - Grand Challenge. (n.d.). grand-challenge.org. https://endovis.grand-challenge.org/.

[15] Brownlee, Jason. “How to Develop a CNN for MNIST Handwritten Digit Classification - Machine Learning Mastery.” *Machine Learning Mastery*, machinelearningmastery.com, 7 May 2019, https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/.

[16] Team, Keras. “Keras Documentation: Image Classification with Swin Transformers.” *Image Classification with Swin Transformers*, keras.io, https://keras.io/examples/vision/swin_transformers/.

[17] Alexey Dosovitskiy, and Lucas Beyer. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 22 Oct. 2020, https://arxiv.org/abs/2010.11929.

[18] “Simula Datasets - Kvasir.” *Simula Datasets - Kvasir*, datasets.simula.no, https://datasets.simula.no/kvasir/.

[19] Wu, Neil. “Introduction of Self-Attention Layer in Transformer | by Neil Wu | LSC PSD | Medium.” *Medium*, medium.com, 3 Oct. 2019, https://medium.com/lsc-psd/introduction-of-self-attention-layer-in-transformer-fc7bff63f3bc#:~:text=Self%2DAttention%20Layer%20check%20attention,associated%20word%27s%20distance%20in%20sentence.

[20] Alammar, Jay. “The Illustrated Transformer.” *The Illustrated Transformer*, jalammar.github.io, 27 June 2018, https://jalammar.github.io/illustrated-transformer/.

[21] Giacaglia, Giuliano. “Transformers.” *Medium*, towardsdatascience.com, 14 Aug. 2021, https://towardsdatascience.com/transformers-141e32e69591.

[22] “Elsevier Enhanced Reader.” *Elsevier Enhanced Reader*, reader.elsevier.com, https://reader.elsevier.com/reader/sd/pii/S0016508505017725?token=05988A6B40082346E3F6B87577E6BC3DFA202154185069EB500B9E7B0B426A4B98F8B5FA8AD01A4061287C59DEA28E4B&originRegion=eu-west-1&originCreation=20220421003432.

[23] Meseeha, Marcelle, and Maximos Attia. “Colon Polyps - StatPearls - NCBI Bookshelf.” *Colon Polyps - StatPearls - NCBI Bookshelf*, www.ncbi.nlm.nih.gov, 15 Aug. 2021, https://www.ncbi.nlm.nih.gov/books/NBK430761/.

[24] Yang, Young Joo, et al. “JCM | Free Full-Text | Automated Classification of Colorectal Neoplasms in White-Light Colonoscopy Images via Deep Learning.” *MDPI*, www.mdpi.com, 24 May 2020, https://www.mdpi.com/2077-0383/9/5/1593.

[25] Team, Keras. “Keras Documentation: Image Classification with Swin Transformers.” *Image Classification with Swin Transformers*, keras.io, https://keras.io/examples/vision/swin_transformers/.

[26]"Tf.Keras.Preprocessing.Image.ImageDataGenerator | TensorFlow Core v2.8.0." *TensorFlow*, www.tensorflow.org, https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

[27] "Weights & Biases – Developer Tools for ML." *Weights & Biases – Developer Tools for ML*, wandb.ai, https://wandb.ai/site.

[28] "Review: Endoscopic Approach to Polyp Recognition." *PubMed Central (PMC)*, www.ncbi.nlm.nih.gov, 1 Apr. 2017, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5369434/#:~:text=Narrow%20band%20imaging%20international%20colorectal%20endoscopic%20(NICE)%20classification12%20is,include%20type%20III%20(deep%20submucosal.

[29] "An Image Is Worth 16x16 Words: ViT | Is This the Extinction of CNNs? Long Live the Transformer?" *YouTube*, www.youtube.com, 8 Oct. 2020, https://www.youtube.com/watch?v=DVoHvmww2lQ.

PREAMBLE

"I hereby confirm that the work presented here in this report and in all other associated material is wholly my own work"

Edoardo Fratantonio