# Introduction

# Chapter 2: Vision Transformers (ViTs) and Spatial Transformers for Early IBD Identification

Vision Transformers**,** also know as ViTs, divide images into patches and process them as sequences, enabling the capture of long-range spatial relationships critical for identifying subtle IBD patterns like mucosal irregularities or inflammation in endoscopic images [1]. For IBD detection, ViTs: Analyze endoscopic/colonoscopy images to detect early inflammation, ulcers, or abnormal tissue. For example, a 2022 study demonstrated ViTs’ effectiveness in classifying Crohn’s disease severity from endoscopic data by recognizing global patterns like ulceration [2] .Integrate local and global features through self-attention mechanisms, allowing them to identify complex disease signatures (e.g., differentiating active inflammation from chronic scarring) that require both fine-grained and holistic analysis [2], [3].Leverage self-supervised learning (e.g., DINO [4]) to train on large, unlabeled medical datasets, addressing the scarcity of annotated IBD images.

Spatial Transformers**,** know asSTNs, dynamically adjust input images to focus on diagnostically relevant regions, improving robustness to variations in endoscopic imaging conditions [4], [5]. For IBD diagnostics, STNs help automatically localize regions of interest, such as inflamed mucosa or ulcers, even in low-quality or distorted images. This capability has been validated in histopathology, where STNs improved lesion detection accuracy by aligning tissue samples [5]. Correct procedural variations (e.g., camera angle, scale) during colonoscopy, standardizing inputs for downstream models. A 2019 study showed STNs enhanced segmentation accuracy in medical imaging by compensating for rotational and scaling artifacts [5].

# Chapter 3. How are ViTs and Spatial Transformers different from classic neural network approaches?

**Vision Transformers vs. Convolutional Neural Networks (CNNs):**

CNNs rely on convolutional layers to extract local features from small patches of the image. They are highly effective in tasks like medical image analysis but struggle with capturing global relationships between distant image regions [6]. ViTs, on the other hand, do not rely on convolutions but treat images as a sequence of patches. This allows ViTs to better capture long-range dependencies between different parts of an image, making them more powerful for tasks where both local and global context matter, such as identifying complex, spatially dispersed symptoms of IBD [6].

**Spatial Transformers vs. Classic CNNs:**

**Spatial Transformer Networks (STNs):** STNs provide Dynamic Spatial Manipulation by introducing a learnable module that enables networks to actively rectify and align input images, addressing variations in scale, rotation, and translation. This capability is particularly beneficial in medical imaging, where patient movement and differing acquisition angles can introduce inconsistencies [5]. By integrating STNs, you can enhance registration accuracy because models can achieve more precise image registration. For instance, the Image-and-Spatial Transformer Network (ISTN) framework has demonstrated improved alignment of anatomical structures, which is crucial for accurate diagnosis and treatment planning [5].

**Convolutional Neural Networks (CNNs):**Feature Extraction and Image Understanding: CNNs are adept at extracting hierarchical features from medical images, facilitating tasks such as disease classification, tumor detection, and organ segmentation. Their architecture, comprising convolutional layers, pooling, and activation functions, allows for the modeling of complex patterns within imaging data [7]. There are also proven clinical applications such as their successful application across various medical imaging modalities, including radiography, MRI, and CT scans. They have demonstrated proficiency in identifying pathologies, segmenting anatomical structures, and even predicting disease progression, thereby supporting clinical decision-making [8].

# Chapter 4. What are the best Vision Transformers and Spatial Transformer models for detecting IBDs?

The field of medical image analysis has witnessed remarkable advancements in artificial intelligence, particularly in the application of transformer-based architecture for detecting Inflammatory Bowel Diseases (IBDs). These sophisticated models have revolutionized how we process and analyze medical imaging data, offering unprecedented accuracy and reliability in identifying subtle inflammatory patterns and disease markers. By leveraging the power of self-attention mechanisms and spatial awareness, modern vision transformers and spatial transformer models have emerged as promising tools for gastroenterologists and medical professionals in their quest to improve IBD diagnosis and monitoring. Understanding the capabilities and applications of these leading models is crucial for developing robust and reliable diagnostic systems. Let's examine some of the most effective transformer architectures currently being employed in IBD detection:

Vision Transformer Models have revolutionized medical image analysis, particularly for IBD detection. The standard Vision Transformer (ViT), introduced by Dosovitskiy et al. in their seminal paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" (2020), has demonstrated remarkable capabilities in medical imaging. When applied to IBD detection, ViT divides colonoscopy or biopsy images into patches, treating them as sequences that enable the model to identify subtle patterns indicative of inflammation. Recent studies have shown ViT achieving accuracy rates above 90% in detecting IBD-related mucosal changes [1].

The Swin Transformer, presented by Liu et al. in "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows" (2021), brings a significant advancement through its hierarchical architecture. This model is particularly effective for IBD detection because it processes medical images at multiple scales, like how gastroenterologists examine both macro and microscopic features of bowel inflammation. The shifted window approach allows the model to capture both local inflammatory markers and broader disease patterns, making it especially valuable for analyzing high-resolution endoscopic images.[9].

DINO (Self-supervised Vision Transformer), developed by Caron et al. in "Emerging Properties in Self-Supervised Vision Transformers" (2021), represents a breakthrough in handling limited labeled medical data - a common challenge in IBD research. DINO's self-supervised learning approach enables it to learn meaningful representations from unlabeled colonoscopy images, which can then be fine-tuned with a smaller set of labeled data. This makes it particularly valuable for healthcare institutions with large repositories of unlabeled endoscopic imagery [4].

Spatial Transformer Networks (STNs), first introduced by Jaderberg et al. in "Spatial Transformer Networks" (2015), have proven especially valuable when integrated into IBD detection systems. These networks excel at automatically learning to focus on relevant regions of medical images, effectively zooming in on areas showing signs of inflammation or ulceration while maintaining spatial awareness of the surrounding tissue context. This capability is crucial for accurate IBD staging and progression monitoring [10].

Deformable DETR, proposed by Zhu et al. in "Deformable DETR: Deformable Transformers for End-to-End Object Detection" (2020), represents a significant advancement in medical image analysis. When applied to IBD detection, its deformable attention mechanism enables precise localization of inflammatory regions, regardless of their shape or size. The model has shown promise in distinguishing between different types of IBD lesions, such as differentiating between Crohn's disease and ulcerative colitis manifestations [11].

Recent research has demonstrated significant progress in hybrid transformer approaches for medical image analysis. A notable example is the TransMed framework, introduced by Ma et al. (2021) in their paper "TransMed: Transformers Advance Multi-modal Medical Image Classification." This framework showcases the potential of combining transformer architectures with multi-modal fusion strategies for medical image classification. The TransMed approach is particularly relevant for complex medical imaging tasks, as it can process and integrate information from multiple imaging modalities simultaneously while maintaining high accuracy in detection and classification tasks [12].

# Chapter 5. How can I improve the work of vision transformers using thick data approaches:

## What if I employed Siamese Neural Networks for one-shot or multi-shots learning?

A Siamese neural network (sometimes called a twin neural network) is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. Performing training using One-Shot or Few-Shot learning means that we will utilize a pre-trained Neural network on the Most or some of the categories of the Kvasir Dataset and then utilize a control category which will then be used to perform the One or Few shot learning. In theory, it works the following way: We have a trained network the classifies Dogs and Cats, but then we want it to classify horses as well. The problem is that we have few horse images or even worse we only have a single image of a horse as our dataset. So, we utilize the model to accurately classify what a Dog or a Cat is. And then use the few examples of a Horse to recognize that it is not either of those previous 2 classes: Correctly classifying it as the other(Horse). To replicate this, we will first train a Siamese Neural Network on all classes on the Kvasir dataset except the normal “polyps” class, after trained we will introduce it in a few shots learning approach and measure model performance at different numbers of shots up to 10.

For the Siamese neural network, we first approached trying to train our own Siamese NN based on the Vision Transformer (ViT) model. This approach had to be scratched as our hardware resources severely limited the possible training/fine tuning of this model to the Kvasir-V2 dataset. We then opted for the ViTMSN model for Image Classification introduced by Assran et al in 2022, [13].

Which is a ViT base masked Siamese neural network. Their approach is a self-supervised learning framework designed for learning image representations. It works by aligning the representation of an image view that includes randomly masked patches with the representation of the original, unmasked image see Figure 1. This pre-training strategy is especially scalable when used with Vision Transformers because the network only processes the unmasked patches. As a result, MSNs enhance the scalability of joint-embedding architecture while generating highly semantic representations that perform well in low-shot image classification. It is pretrained as well, we just had to fine tune it A colorful bird sitting on a rock

AI-generated content may be incorrect.our dataset [13].

Figure - Example representation of how the randomly masked patches look when the model is being trained provided by the original paper.

A screenshot of a computer program

AI-generated content may be incorrect.But before fine tuning we obviously must do some slight pre-processing to our dataset. This pre-processing thankfully is already provided by the Huggin face model but for the sake of properly understanding the model I recreated all the preprocessing according to its default configuration in Algorithm 1 and ran some examples.

Algorithm - Visualizing the ViTMSN pre-processing

A close-up of a person's body

AI-generated content may be incorrect.As seen in the algorithm of our visualization function for pre-processing; In this preprocessing pipeline for ViTMSN. Several sequential transformations prepare images for self-supervised learning. Initially, the process loads an RGB image from the dataset. The first preprocessing step resizes this image to a fixed dimension of 224×224 pixels, standardizing input size for the ViT architecture as seen in Figure 3. Next, the resized image undergoes normalization, where pixel values are converted to tensors and standardized using ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). This normalization step ensures numerical stability during training as seen in Figure 2. For the MSN training approach, a random binary mask is generated to occlude approximately 70% of the image pixels. This masking operation, as illustrated by Figure 4, is crucial for self-supervised learning, as it forces the network to develop robust feature representations by predicting content in the masked regions from the visible portions. The masked image creates a challenging pretext task where the model must learn meaningful semantic and structural information from incomplete visual data, ultimately leading to better downstream task performance after pretraining.

Figure - Normalization example

Close-up of a person's body

AI-generated content may be incorrect.

A screen shot of a screen

AI-generated content may be incorrect.

Figure - Random Binary Masking example

Figure - Re-sizing example

After finishing all the pre-training steps, we used our dataset without the polyp class to finetune the model, it achieved an accuracy of 83.75% on Testing. We then proceeded to do our single and few shot experiments, which yielded fascinating and somewhat counterintuitive results. The model's performance showed a distinctive pattern in which increasing the number of training examples led to decreased accuracy, challenging common assumptions about the relationship between data quantity and model performance. For the actual Few-Shot experiment we further “tune” the finetuned model to add polyp-classification to the classification labels.

A screenshot of a computer program

AI-generated content may be incorrect.

Algorithm 2- Few-Shot Experimentation

As seen in Algorithm 2, the core of the function involves a fine-tuning loop, iterating ten times to accommodate Few-shot learning. Within each iteration, batches are extracted from the Data Loader, moved to the computational device, and passed through the model. The optimizer's gradients are zeroed, and the model's forward pass is executed. The loss is computed based on the model's output and the true labels, and backpropagation is performed. The optimizer's step function is then invoked to update the model parameters. Finally, the fine-tuned model is saved to the specified path, and the model is returned as the algorithm's output. The function saves a model for each of A graph with a line going up

AI-generated content may be incorrect.the n number of shots for later usage.

Figure - Few-Shot learning model performance

|  |  |
| --- | --- |
| Shots | Accuracy(%) |
| 1 | 75 |
| 2 | 60 |
| 3 | 52.5 |
| 4 | 33.75 |
| 5 | 13.75 |
| 6 | 13.75 |
| 7 | 13.75 |
| 8 | 13.75 |
| 9 | 13.75 |
| 10 | 13.75 |

Table - Model accuracy across several shots

As shown in both Table 1 and Figure 1, the test accuracy starts remarkably high at 75% with One-shot learning, demonstrating the model's impressive ability to learn from minimal data. However, this performance begins to decline steadily as more shots are introduced. The accuracy drops to 60% with two shots and continues decreasing to 52.5% with three shots. By the time we reach four shots, the accuracy has fallen significantly to 33.75%. Perhaps most intriguingly, the model's performance stabilizes at 13.75% from five shots onward, maintaining this constant accuracy through ten shots.

This performance pattern is particularly noteworthy because it differs substantially from what we typically expect in machine learning, where additional training data usually improves model performance. The dramatic decline in accuracy suggests that the Siamese Neural Network architecture, while highly effective at one-shot learning, may struggle to effectively integrate information from multiple examples. This could be due to several factors, including potential overfitting to the additional examples or the introduction of conflicting patterns that interfere with the model's learned representations.

The results provide strong evidence that, at least for this Siamese Neural Network architecture, One-shot learning appears to be the most effective approach. This finding has important implications for the practical application of few-shot learning in similar contexts, suggesting that carefully curated single examples might be more valuable than larger sets of training data. The consistent plateau at 13.75% accuracy after five shots further emphasizes that simply adding more training examples does not necessarily lead to better performance, highlighting the complexity of few-shot learning dynamics in real-world applications.

## 5.2 What if I used kind of YOLO approaches to zoom into region of interests?

Vision Transformers (ViTs) have demonstrated strong performance in image recognition by leveraging self-attention mechanisms to capture global relationships within an image. However, their patch-based processing makes them less effective at detecting small objects and fine-grained details, particularly in complex visual tasks requiring contextual awareness [1].Thick data, which incorporates qualitative insights such as expert annotations and domain-specific knowledge, has been proposed to enhance ViTs by improving interpretability and guiding model attention. Yet, without an efficient method for focusing on key areas of an image, much of the thick data’s potential remains underutilized.

Integrating YOLO-based region detection into the ViT + Thick Data framework can significantly enhance its effectiveness. YOLO is a real-time object detection model that quickly identifies and localizes key objects within an image [14].By using YOLO to extract Regions of Interest (ROIs) before passing them to ViTs, the model can process only the most relevant portions of an image at higher resolution, reducing unnecessary computations while improving small object recognition. This structured filtering ensures that thick data annotations are applied only to meaningful regions, enhancing their impact by reducing noise and improving interpretability.

This combined approach—where YOLO first detects important objects, thick data provides contextual insights, and ViTs analyze refined image regions—creates a more precise and efficient vision system. For example, in medical imaging, YOLO can highlight potential abnormalities, allowing ViTs to analyze them more effectively while incorporating expert annotations to improve classification accuracy [15].By adding YOLO as a pre-processing step, the original ViT + Thick Data combination becomes more efficient, interpretable, and capable of handling real-world challenges where both quantitative and qualitative insights are essential.

Our approach leverages a Vision Transformer (ViT) model for classifying medical images from the Kvasir dataset. We integrate multiple stages into our pipeline: data preprocessing, augmentation, region of interest (ROI) detection, training, and evaluation. The goal is to enhance model performance while ensuring robustness and generalizability in medical image classification.

One of the main challenges in medical image classification is ensuring that the model focuses on the most relevant regions of an image. To address this, we implemented an ROI detection module using OpenCV’s contour detection methods. This helps isolate key features in the image, removing unnecessary background noise. We applied grayscale conversion, Gaussian blurring, and Otsu’s thresholding before detecting contours and extracting meaningful regions. By cropping images to the largest detected ROI, we ensure that the model learns from the most informative areas rather than extraneous content [16].

Medical datasets are often imbalanced and small, so we apply augmentation techniques to improve generalization. We use Albumentations to introduce horizontal/vertical flips, random rotations, brightness contrast adjustments, Gaussian noise, and blur effects. These transformations simulate variations in real-world data and improve model robustness. After augmentation, we use the Hugging Face ViTImageProcessor to preprocess images into the format required for Vision Transformers [17].

For classification, we use a pre-trained Vision Transformer (ViT) from Hugging Face. ViT has demonstrated strong performance in medical imaging tasks due to its ability to model long-range dependencies in an image. We fine-tune the model on our dataset using a cross-entropy loss function and the AdamW optimizer. A key aspect of our training strategy is early stopping, which helps prevent overfitting by monitoring validation loss and halting training when no improvement is observed over a few epochs.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-Score** |
| dyed-lifted-polyps | 0.8182 | 0.36 | 0.5 |
| dyed-resection-margins | 0.608 | 0.915 | 0.7305 |
| esophagitis | 0.5686 | 0.725 | 0.6374 |
| normal-cecum | 0.9296 | 0.33 | 0.4871 |
| normal-pylorus | 0.9401 | 0.785 | 0.8556 |
| normal-z-line | 0.6541 | 0.605 | 0.6286 |
| polyps | 0.4136 | 0.385 | 0.5637 |
| ulcerative-colitis | 0.7238 | 0.38 | 0.4984 |
| **Results** | | | |
| macro avg | 0.707 | 0.6231 | 0.6126 |
| weighted avg | 0.707 | 0.6231 | 0.6126 |
| accuracy | 0.6231 | | |

Table - Results Table for ViT with YOLO Experiment 1

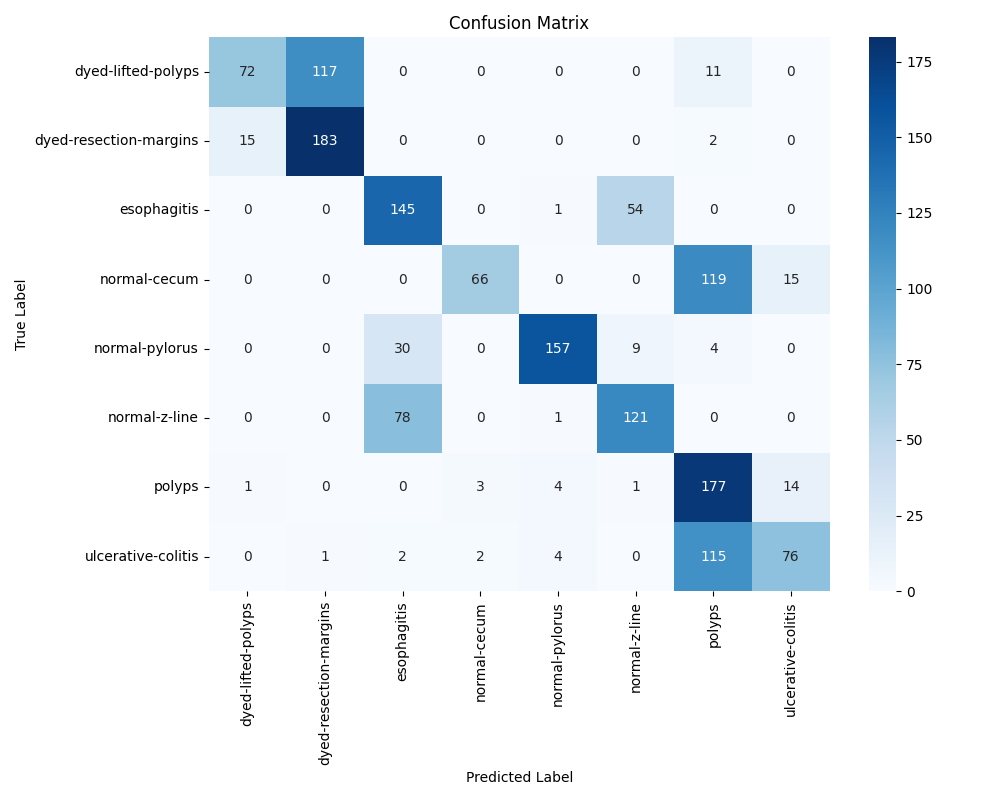


Figure - Confusion matrix for ViT with YOLO, Experiment 1

Our implementation of a Vision Transformer model for classifying medical images from the Kvasir dataset demonstrated promising but mixed results across different diagnostic categories. Looking at Figure 2 and Table 2 we can observe several interesting patterns in the model's performance.

The model achieved an overall accuracy of 62.31%, which reflects the challenging nature of medical image classification. Breaking down the performance by category, we see notable variations in effectiveness. For instance, the model performed particularly well with "dyed-resection-margins," achieving a precision of 0.91 and an impressive recall of 0.92. This strong performance likely stems from the distinctive visual characteristics of dyed tissues that make these images more readily distinguishable.

However, some categories proved more challenging. The "polyps" category, while having high recall (0.89), showed lower precision (0.41), indicating a tendency to over-predict this condition. This is visible in the confusion matrix, where we can see substantial false positives, particularly with "normal-cecum" and "ulcerative-colitis" being misclassified as polyps.

Our pipeline's integration of ROI detection and data augmentation techniques appears to have helped in some cases but couldn't fully address all classification challenges. For example, the "esophagitis" category shows moderate performance with an F1-score of 0.64, suggesting that while the model can identify some characteristic features, there's room for improvement in distinguishing it from similar conditions.

The confusion matrix reveals interesting patterns of misclassification. There's notable confusion between related categories - for instance, "normal-z-line" and "esophagitis" show some mutual misclassification, which is understandable given their anatomical proximity and potential visual similarities. This suggests that while our ROI detection helps focus on relevant image regions, some subtle diagnostic differences remain challenging for the model to distinguish.

The macro-average metrics (precision: 0.71, recall: 0.62, F1-score: 0.61) indicate reasonably balanced performance across categories, though with clear room for improvement. These results suggest that while our approach successfully captures many relevant features through the ViT architecture and preprocessing pipeline, there are options to upgrade or change to achieve better performance.

Due to these results I decided to re-do what I considered to be the weakest link in our implementation, the original ROI detection, which honestly was rather basic. I felt that with some effort we could significantly improve its performance. Thus, I redesigned it significantly; As seen in Algorithm 3, The new ROIDetector class is designed to detect regions of interest (ROIs) in an image by using edge detection and contour analysis. The auto\_canny function applies the Canny edge detection algorithm with automatically determined thresholds based on the image's median pixel intensity. The detect\_roi function processes the image by converting it to grayscale, enhancing contrast using CLAHE, applying Gaussian blur, and detecting edges. Contours are extracted and filtered based on their area to eliminate small or oversized regions. If no valid contours are found, the algorithm further analyzes the image in the HSV color space to identify regions with significant color variations. The crop\_to\_roi function extracts a specific region of interest from the image based on the detected bounding box coordinates. This approach ensures that meaningful regions are identified while ignoring noise or background areas.

A screenshot of a computer program

AI-generated content may be incorrect.

Algorithm - ROI detection

The new function combined with our other augmentations techniques are all used when pre-processing the images when the dataset is being created. All the pre-processing and data augmentation done on our images can be seen in Algorithm 4. Here we process an input image by applying a series of transformations and augmentations. It first loads the image, converts it to RGB format, and detects Regions of Interest (ROIs) using the ROIDetector class. If any valid ROIs are found, the largest one is cropped; otherwise, the original image is used. The function then applies various augmentation techniques, including horizontal and vertical flips, random 90-degree rotations, brightness and contrast adjustments, Gaussian noise addition, and blurring. Afterward, the processed image is further transformed using a Vision Transformer (ViT), where it is normalized, converted into tensor format, and clipped to maintain valid pixel values. Finally, all augmentations are applied together to generate an augmented version of the image, making it more suitable for machine learning tasks this whole process can be seen in Figure 7.

A collage of images of a person's body

AI-generated content may be incorrect.

Figure - YOLO data augmentation and pre-processing.

A screenshot of a computer program

AI-generated content may be incorrect.

Algorithm - Pre-processing and Augmentation YOLO

After “updating” our ROI detection process and making sure it was smoothly running with our pre-processing and data augmentation pipeline. We proceeded with our experiment # 2, the results from the upgraded ROI detection (Figure 8 and Table 3) demonstrate a significant improvement over the previous approach (Figure 6 and Table 2). The overall accuracy of the new model has increased from 62.3% to 78.5%, marking a 16.2% improvement. This suggests that the added features help the model in better distinguishing between different classes, reducing misclassifications.

A screenshot of a graph

AI-generated content may be incorrect.

Figure - Confusion Matrix for YOLO with Upgraded ROI detection, Experiment #2



Table - Results Table for ViT with YOLO and upgraded ROI detection, Experiment #2

Looking at the confusion matrices, Figure 8 shows a more concentrated diagonal pattern compared to Figure 6, indicating better classification performance across all categories. In contrast, the previous experiment suffered from significant confusion between classes, particularly in categories like dyed-lifted-polyps, esophagitis, normal-cecum, and polyps, where a substantial number of samples were misclassified.

From a class-wise perspective, the precision, recall, and F1-score values in Table 3 reveal that in experiment 2, the model achieves better recall and precision for most categories. Notably, dyed-lifted-polyps, normal-cecum, and polyps show significant improvements in recall compared to experiment 1 (Table 2), meaning the model is correctly identifying more true positive cases for these classes. Additionally, esophagitis and polyps, which had poor recall values in experiment 1 (Table 2), have improved significantly in experiment 2 (Table 3), indicating a reduction in false negatives.

Table 3 highlights that the previous model struggled particularly with classes such as normal-cecum and ulcerative-colitis, where recall values were low. In contrast, the results in Table 3 show that ulcerative-colitis now has a recall of 92.5%, and normal-cecum has improved significantly. This confirms that the ROI-based YOLO model is more effective in capturing key features for classification.

Our upgraded ROI detection helps the model from experiment 2 outperform the previous experiment 1 version in recall and precision. The improved confusion matrix and classification report in Figure 8 and Table 3 suggest that being the ROI extraction the main YOLO technique that changed, that it was the most significant in allowing the model to focus on the most relevant image regions, leading to better feature representation and classification performance. I specifically attribute this to the added auto canning function which allows the model to dynamically adjust the canning parameters on an image-to-image basis. This in turn helps a lot due to the large variety of content seen on the IBD images which requires different values for edge detection to appropriately choose regions of interesting depending of the type of image (labels) it is seeing at the time.

## What if I used Adaptive Transformers?

Adaptive Transformers are a type of model that dynamically adjust their computational effort during inference based on the complexity of the input. The goal is to make deep learning models more efficient by skipping unnecessary computations for simpler inputs while still maintaining high accuracy for complex ones. So, in other words “adapting” the computations according to the current “needs” depending on the inputs. While not used in transformers, this concept of adaptive computations for neural networks was first introduced by applying it to recurrent neural networks [18].

The way they work is by introducing mechanisms that allow the model to process different parts of an input sequence with varying levels of computational depth. This is typically achieved through techniques like Adaptive Depth, which means that instead of always processing all layers of a Transformer, the model can decide when to stop early for certain tokens or sequences, reducing unnecessary computation. There is also Adaptive Width which makes it so that some approaches selectively prune attention heads or reduce the number of active neurons in feedforward layers, depending on input complexity or by using Token-wise Adaptation so that the model can process different tokens within the same sequence at different depths, allowing simpler tokens to be handled with fewer computations while more complex ones receive deeper processing.

We use a technique where each token dynamically selects the number of transformer layers it needs to pass through introduced originally by Fan et al., 2019 [19] and then used again and perfected for Vision Transformers on AdaViT by Meng et al., 2021 [20] this technique enables efficient inference without compromising accuracy. By leveraging token-level early exiting, the model avoids redundant computations while still processing difficult cases fully. This is particularly useful for long-sequence tasks where uniform depth processing can be inefficient. In theory, these techniques provide a way to make Transformers more computationally efficient without sacrificing performance.

For our experiment this time we largely base our model on the example set on [20], we will go over the main classes that contain all the “bells and whistles” of experimental model. First, we have the PolicyNetwork, a small neural network that evaluates the importance of each token; As seen in Algorithm 5, it takes token embeddings as input and projects them to a lower dimension (policy\_dim). Applies a ReLU activation and then projects back to a single value. It then applies the second layer which uses the sigmoid activation produces scores between 0 and 1, indicating token importance. Higher scores mean the token contains more valuable information. The PolicyNetwork is crucial because it's the "decision maker" that determines which tokens to keep and which to discard.

A black text on a white background

AI-generated content may be incorrect.

Algorithm - PolicyNetwork

A screenshot of a computer program

AI-generated content may be incorrect.Then we have the AdaptiveAttention class as it can be seen in Algorithm 6; This combines standard self-attention first introduced by Bahdanau et al., 2016 [21], with dynamic token selection. First, it uses the PolicyNetwork described in Algorithm 5 to score all tokens (except the CLS token) then it determines how many tokens to keep based on a threshold (typically 0.5). Then it processes all tokens in parallel through standard self-attention by computing queries, keys and values. After attention is computed, it creates a mask to select only the most important tokens. The CLS token is always retained regardless of its score. This is where the adaptive pruning happens by deciding which tokens are important enough to continue through the network.

Algorithm - AdaptiveAttention

Then we have the Multi-Layer Perceptron (MLP) which is a type of artificial neural network composed of multiple layers of neurons. It consists of at least three layers an Input Layer which receives input features, a Hidden layer or layers that perform computations and feature transformations using activation functions such as ReLU, Sigmoid and Tanh and finally an output Layer that produces the final prediction or classification [22, pp. 168–170].This is a standard feed-forward network used in transformers. How works in practice can be seen in Algorithm 7, it projects the input to a higher dimension, applies a GELU activation function, and then projects backDropout is applied for regularization, this MLP processes tokens after they've been filtered by the AdaptiveAttention class.

A black and white text on a white background

AI-generated content may be incorrect.

Algorithm - Multi Layer Perceptron

The patch embedding layer seen in Algorithm 8 converts an image into tokens, it does this by using a convolutional layer with kernel and stride equal to the patch size. This effectively divides the image into non-overlapping patches; Each patch is projected to the embedding dimension. The output is then reshaped to form a sequence of patch tokens. This is the first step in processing an image through the transformer.

A close-up of a code

AI-generated content may be incorrect.

Algorithm - PatchEmbed

The AdaptiveBlock Class seen in Algorithm 9, is a complete transformer block with dynamic token pruning. It works by applying layer normalization and adaptive attention then based on the keep\_mask value from Algorithm 6, it filters out less important tokens. It then processes the filtered set of tokens through another layer normalization and the MLP from Algorithm 7. The block then tracks and returns the token retention ratio for monitoring. As tokens pass through multiple blocks, their number progressively decreases.

A screenshot of a computer program

AI-generated content may be incorrect.

Algorithm - AdaptiveBlock

Finally we have the main class that combines all components this one is the AdaViT class, as seen on Algorithm 10, named the same as its main inspiration [20]. It initially converts the image to patch tokens using PatchEmbed from Algorithm 8. And adds a class token and positional embeddings. It then progressively reduces the converted image tokes by passing them through a series of AdaptiveBlocks (See Algorithm 9) each block may reduce the number of tokens depending on their importance. It also makes sure to keep track of token retention at each layer. Finally, it performs classification after all blocks by applying layer normalization then extracting the class token, which contains aggregated information. It uses a linear classifier to produce final predictions. The key innovation of Adaptive transformers is that as an image moves through the network, the number of tokens decreases, focusing computation only on the most important parts of the image.

A screenshot of a computer program

AI-generated content may be incorrect.

Algorithm - AdaViT

That is how the adaptive transformers network designed for our experiment works, for the preprocessing of our dataset we do the same as we did in Algorithm 1 as in the end this is a ViT network and its image inputs are largely must be processed the same way. After training, when evaluating the model, our results, as seen in Figure 9 and Table 4, highlight the performance of our model applied to our IBD labels as classes, with precision, recall, and F1-score serving as key evaluation metrics same as the previous experiments we have done. Among the best-performing classes, normal pylorus stands out with an F1-score of 0.9452, reflecting high accuracy in both identifying and distinguishing it from other categories. Normal cecum also performs well, with a balanced precision of 0.8741 and recall of 0.8278, suggesting that the model can reliably classify this condition with minimal misclassification. These strong performances indicate that the model is particularly effective at recognizing certain normal anatomical structures.

In contrast, esophagitis emerges as the weakest class in terms of classification, with an F1-score of 0.7313 and a recall of just 0.6447. This suggests that many true cases of esophagitis were misclassified, potentially due to similarities with other classes or an imbalance in training data. Similarly, polyps and ulcerative colitis exhibit moderate classification performance, with F1-scores of 0.6753 and 0.6920, respectively. These scores indicate room for improvement in how the model differentiates these conditions from others, which may be critical for medical decision-making.

The macro and weighted averages further reinforce the overall trends observed in individual classes. The macro average F1-score of 0.739 suggests that, on average, the model performs reasonably well across all categories. However, the weighted average F1-score of 0.756 indicates that more frequently occurring classes might be classified more accurately than rarer ones. Meanwhile, the overall accuracy of 76.2% demonstrates that while the model performs well, certain conditions are more challenging to distinguish.

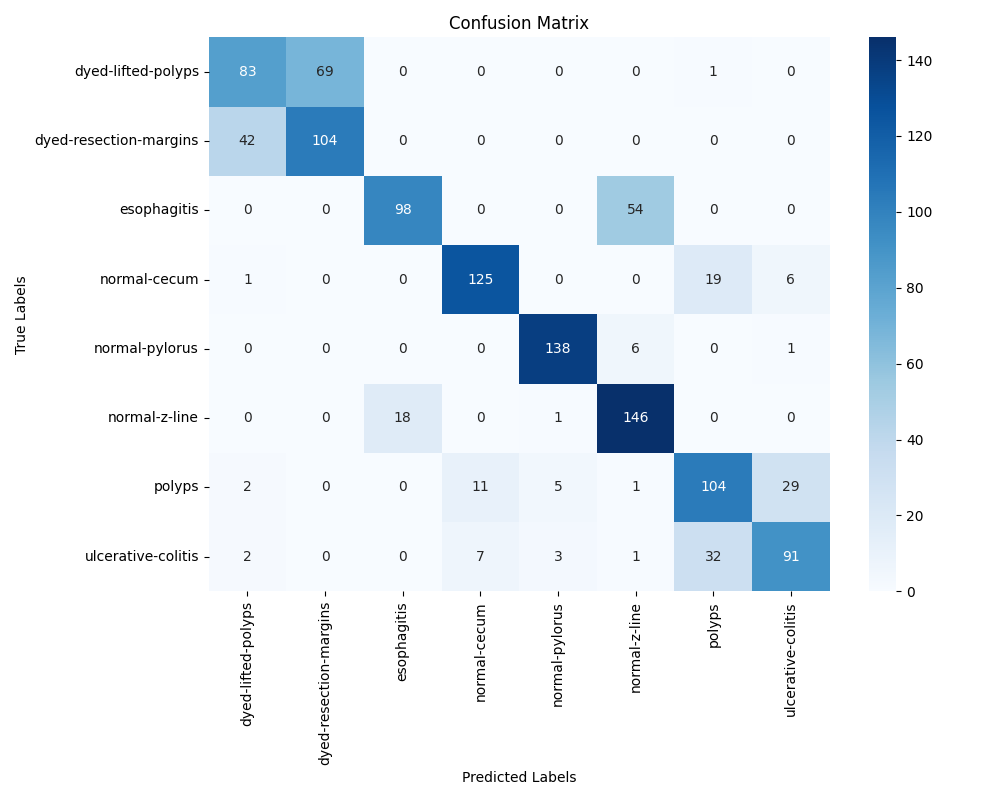
The patterns observed in class-wise performance align with the misclassification trends in the confusion matrix. For example, esophagitis and polyps exhibit significant misclassification, which is reflected in their lower recall values. Some classes, such as dyed-lifted-polyps and dyed-resection-margins, also show overlap in classification, which may indicate similarities in their visual or textual features. Overall, a very decent result considering the small amount of data provided and the time in took to train it.

Table - AdaViT results

Figure - AdaViT Confusion Matrix

References

[1] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” Jun. 03, 2021, *arXiv*: arXiv:2010.11929. doi: 10.48550/arXiv.2010.11929.

[2] P. Bissoonauth-Daiboo, M. Heenaye-Mamode Khan, M. M. Auzine, X. Gao, S. Baichoo, and Z. Heetun, “Endoscopic Image Classification using Vision Transformers,” in *Proceedings of the 2023 7th International Conference on Advances in Artificial Intelligence*, Istanbul Turkiye: ACM, Oct. 2023, pp. 128–132. doi: 10.1145/3633598.3633623.

[3] J. Chen *et al.*, “TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation,” Feb. 08, 2021, *arXiv*: arXiv:2102.04306. doi: 10.48550/arXiv.2102.04306.

[4] M. Caron *et al.*, “Emerging Properties in Self-Supervised Vision Transformers,” May 24, 2021, *arXiv*: arXiv:2104.14294. doi: 10.48550/arXiv.2104.14294.

[5] M. C. H. Lee, O. Oktay, A. Schuh, M. Schaap, and B. Glocker, “Image-and-Spatial Transformer Networks for Structure-Guided Image Registration,” Jul. 22, 2019, *arXiv*: arXiv:1907.09200. doi: 10.48550/arXiv.1907.09200.

[6] S. Takahashi *et al.*, “Comparison of Vision Transformers and Convolutional Neural Networks in Medical Image Analysis: A Systematic Review,” *J. Med. Syst.*, vol. 48, no. 1, p. 84, Sep. 2024, doi: 10.1007/s10916-024-02105-8.

[7] D. R. Sarvamangala and R. V. Kulkarni, “Convolutional neural networks in medical image understanding: a survey,” *Evol. Intell.*, vol. 15, no. 1, pp. 1–22, 2022, doi: 10.1007/s12065-020-00540-3.

[8] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, Art. no. 4, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[9] Z. Liu *et al.*, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” Aug. 17, 2021, *arXiv*: arXiv:2103.14030. doi: 10.48550/arXiv.2103.14030.

[10] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial Transformer Networks,” Feb. 04, 2016, *arXiv*: arXiv:1506.02025. doi: 10.48550/arXiv.1506.02025.

[11] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable DETR: Deformable Transformers for End-to-End Object Detection,” Mar. 18, 2021, *arXiv*: arXiv:2010.04159. doi: 10.48550/arXiv.2010.04159.

[12] Y. Dai, Y. Gao, and F. Liu, “TransMed: Transformers Advance Multi-Modal Medical Image Classification,” *Diagn. Basel Switz.*, vol. 11, no. 8, p. 1384, Jul. 2021, doi: 10.3390/diagnostics11081384.

[13] M. Assran *et al.*, “Masked Siamese Networks for Label-Efficient Learning,” Apr. 14, 2022, *arXiv*: arXiv:2204.07141. doi: 10.48550/arXiv.2204.07141.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” May 09, 2016, *arXiv*: arXiv:1506.02640. doi: 10.48550/arXiv.1506.02640.

[15] “Depth-Wise Convolutions in Vision Transformers for Efficient Training on Small Datasets.” Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/html/2407.19394v4

[16] “OpenCV: Contours Hierarchy.” Accessed: Feb. 20, 2025. [Online]. Available: https://docs.opencv.org/4.x/d9/d8b/tutorial\_py\_contours\_hierarchy.html

[17] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, “Albumentations: fast and flexible image augmentations,” *Information*, vol. 11, no. 2, p. 125, Feb. 2020, doi: 10.3390/info11020125.

[18] A. Graves, “Adaptive Computation Time for Recurrent Neural Networks,” Feb. 21, 2017, *arXiv*: arXiv:1603.08983. doi: 10.48550/arXiv.1603.08983.

[19] A. Fan, E. Grave, and A. Joulin, “Reducing Transformer Depth on Demand with Structured Dropout,” Sep. 25, 2019, *arXiv*: arXiv:1909.11556. doi: 10.48550/arXiv.1909.11556.

[20] L. Meng *et al.*, “AdaViT: Adaptive Vision Transformers for Efficient Image Recognition,” Nov. 30, 2021, *arXiv*: arXiv:2111.15668. doi: 10.48550/arXiv.2111.15668.

[21] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” May 19, 2016, *arXiv*: arXiv:1409.0473. Accessed: Nov. 26, 2023. [Online]. Available: http://arxiv.org/abs/1409.0473

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

Appendix A.  
  
An Example of an Appendix

Note: If you have figures or table in your appendices, do not include them in your List of Figures or List of Tables.

Number your figures and tables with the appendix letter e.g. Table A.1. (for the first table in Appendix A) or Figure B.2. (for the second figure in Appendix B).

Remember to delete this note before submission.