UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Active Learning strategies for WCE images classification

Author:
Sara BARDAJÍ SERRA

Supervisor:
Dr. Santi SEGUÍ MESQUIDA
Pere GILABERT ROCA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

January 16, 2024

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Active Learning strategies for WCE images classification

by Sara BARDAJÍ SERRA

Medical imaging, particularly through Wireless Capsule Endoscopy (WCE), has revolutionized gastrointestinal health by capturing intricate details of the digestive tract, with a focus on identifying potential precursors like polyps. However, labeling vast and continuous WCE videos for machine learning poses significant challenges due to its resource-intensive nature. This research explores the realm of active learning (AL) to optimize WCE image classification, aiming to enhance model performance with minimal labeled data. Utilizing WCE videos, we established an AL framework that at every cycles selects a video to query for labels. Our study implemented various sampling strategies, categorized into uncertainty-based and diversity-based approaches. Initial outcomes with uncertainty-based methods aligned closely with random sampling, prompting a shift towards diversity-based strategies. Notably, the cover strategy, especially with its autoencoder variant, and the clustering strategies, both diversity-based, exhibited promising results. Despite these advancements, discerning the superior strategy between cover with autoencoder and clustering necessitates further exploration. This study shows the potential of AL in WCE image classification while highlighting areas for future investigation.

Acknowledgements

I want to thank my tutors, Santi and Pere, for guiding and supporting me throughout this project. Their advice and encouragement have been invaluable.

I would also like to thank my family for their constant support.

Contents

Chapter 1

Introduction

1.1 Motivation and Goals

Medical imaging plays a crucial role in the early detection and diagnosis of various diseases, contributing significantly to improved patient outcomes. With the growing efforts in deep learning research, more concretely in computer vision tasks, deep learning algorithms have quickly become a methodology of choice in the analysis of medical images (Litjens et al., 2017). In the realm of gastrointestinal health, Wireless Capsule Endoscopy (WCE) (Iddan et al., 2000) has emerged as a non-invasive and patient-friendly technique, capturing high-resolution images and videos of the digestive tract. One of the challenging tasks within this domain is the identification and classification of polyps (Jin et al., 2020) (Mamonov et al., 2014), which are potential precursors to gastrointestinal malignancies.

To effectively train supervised machine learning algorithms, access to expansive labeled datasets is necessary—a task that can sometimes be challenging. Labeling data is an expensive and time-consuming process, often requiring an expert to complete it. While this reliance is commonly assumed, it takes on particular significance in the context of Wireless Capsule Endoscopy (WCE). In WCE, the acquired data comprises lengthy videos, often spanning hours and capturing intricate details of the gastrointestinal tract. The sheer volume and continuous nature of these recordings amplify the complexity of the data labeling process, rendering it a highly resource-intensive and costly endeavor.

Active learning (AL) (Settles, 2009a) is a machine learning paradigm where the algorithm is actively involved in selecting the most informative data points for training. The fundamental premise of AL is rooted in the notion of information gain. Rather than passively accepting a predefined set of labeled instances, the algorithm actively seeks out samples that present a higher degree of uncertainty or complexity. By doing so, AL aims to reduce the overall volume of labeled data required for effective model training.

This project focuses on the development of an AL technique tailored for the classification of images obtained from WCE videos. The primary objective is to design an approach that prioritizes the labeling of videos, optimizing the model's learning process. The underlying hypothesis is that judiciously chosen samples, guided by a well-defined metric, will lead to enhanced model performance with minimal labeled data.

1.2 Document Structure

This project follows a structure that goes bottom-up. It starts by introducing some background knowledge on WCE and AL and then continues to present the strategies

attempted to accomplish the objectives that have been previously mentioned. It is organized as follows:

1. **Background:** This chapter aims to present the basic concepts that are necessary to develop this project. We start by explaining what WCE is and why it's important for looking inside the digestive tract using videos. Then, we talk about AL, a smart way for computers to learn, especially when there's a lot of information to process or the cost of labeling data is very high. We show how these concepts set the stage for the rest of the project.

2. **Method:** Building upon the foundational knowledge established in the previous chapter, this chapter explains the strategies deployed to achieve the predefined objectives. It provides an in-depth exploration of the strategies that have been proposed, the reasoning behind them and their implementation.

3. **Results:** Contains the work environment and the frameworks used during the whole development of this project. It also contains a discussion on the obtained results. The proposed strategies are tested and a clear visualization of the results is shown.
   This chapter presents the experiments we have performed, explaining the different strategies that have been implemented and offering visualizations that help understand which strategies have provided better results.

4. **Conclusions and future research:** Brief summary of the project. It comments on whether the objectives have been met or not and attempts to reach some conclusions based on the experiments that have been performed.
   It also contains a discussion on possible improvements and mentions some ideas to further expand the project in the future.

Chapter 2

Background

This chapter serves as a foundational exploration, providing the background knowledge required to comprehend the motivation and implementation behind the strategies that will be explained in the following chapters.

2.1 Wireless Capsule Endoscopy

The exploration and diagnosis of GI disorders have traditionally relied on a series of specialized procedures, each designed to inspect specific segments of the GI tract. Gastroscopies primarily focus on examining the upper digestive system, including the esophagus and stomach, providing insights into conditions such as ulcers, gastritis, and tumors within this region. Small-bowel endoscopies, on the other hand, extend the examination to the small intestine, enabling the detection of abnormalities such as tumors, ulcers, and signs of inflammatory bowel diseases like Crohn's disease. Meanwhile, colonoscopies serve as the gold standard for evaluating the large intestine and rectum, facilitating the detection of polyps, colorectal cancer, and other colorectal conditions. These procedures are all endoscopic, meaning that they requires the use of an endoscope; a long tube with a camera attached to it, making them invasive procedures that often cause discomfort to the patient. Furthermore, the need for multiple procedures to inspect different parts of the GI tract complicates the diagnostic process, potentially delaying treatment and increasing patient inconvenience.

Wireless Capsule Endoscopy, introduced by Iddan et al., 2000, serves as a procedure for comprehensive imaging of the entire gastrointestinal (GI) tract. It consists of a swallowable pill-sized capsule that contains a camera, light source, lens, radio transmitter and battery that allows for it to be propelled along the whole GI tract while capturing images and then transmitting them to an external receiver for their posterior analysis. Through this procedure, WCE offers a patient-friendly alternative to conventional methods and has become the standard technique for the diagnosis of GI abnormalities such as bleeding, detection of polyps and lesions, small bowel obstructions, Crohn's disease and other critical conditions.

While WCE represents a significant leap forward in GI diagnostics, offering numerous advantages over traditional methods at the time of the procedures, it is not without its own limitations. The videos recorded by the capsule can have a duration of up to 12h (Vasilakakis et al., 2019) since this is not a targeted exploration, but a complete recording of the GI tract, therefore requiring additional post-analysis. Although these videos usually have a low frame rate, if we suppose that a 2 frames-per-second capsule is used, this means that a single video can contain up to 86400 images, and this number can get even larger with higher frame-rate capsules. The analysis of the videos is also complicated by the inability to control the direction of

the camera while in the GI tract. This causes frames to exhibit a significant variability in camera orientation and perspective (Seguí et al., 2016). This complexity of the variability in frames combined with the duration of a single WCE study, renders video analysis by physicians a challenging and labor-intensive task.

For this reason, the application of AI techniques has garnered increasing attention for detecting abnormal images in WCE, thereby aiding in the analysis of these videos. Deep learning, in particular, has significantly enhanced the performance of WCE. To assist physicians in diagnosis, deep learning models have been employed to develop Computer-Aided Detection (CAD) systems. These systems automatically identify gastrointestinal abnormalities, including bleeding (Hajabdollahi et al., 2018) (Aoki et al., 2020) (Saraiva et al., 2021), polyps or tumors (Gilabert et al., 2022) (Falin, Haihua, and Ning, 2022) (Saito et al., 2020), ulcers (Aoki et al., 2019) (Klang et al., 2020) (Ribeiro et al., 2022), and other pathologies (Ciaccio et al., 2010) (Malagelada et al., 2012). Still, to enhance the effectiveness of CAD systems, a vast amount of labeled data is essential for training deep learning models. Consequently, the laborious task of data annotation by experts remains integral.

2.2 Active Learning

In the realm of deep learning, the efficacy and accuracy of models largely depend on the quality and quantity of labeled data used for training. Labeled data forms the foundation upon which neural networks and other advanced algorithms learn to recognize patterns, make predictions, and execute complex tasks. However, annotating data is a labor-intensive and time-consuming process that often necessitates domain expertise to ensure both accuracy and relevance. Recognizing these challenges, Active Learning (AL) emerges as a methodology wherein the most informative unlabeled data points are selected for a human to review and label, aiming to maximize model improvement efficiently. AL is based on the idea that a model will make better predictions if it can chose which samples to learn from (Settles, 2009b).

2.2.1 AL Framework

An AL application begins with a small set of labeled data and a larger set of unlabeled data. Initially, the model is trained using the labeled data and subsequently employed infer on the set of unlabeled data. The Learner then assigns priority scores to the data points within this unlabeled set, subsequently selecting the data point with the highest priority score for labeling. An expert then labels this selected data point, and augments the labeled dataset. This iterative process continues until a predefined performance threshold is achieved. A visual representation of this framework is depicted in Figure 2.1. Although this is the general idea of how AL works, there are different frameworks to actively learn in.

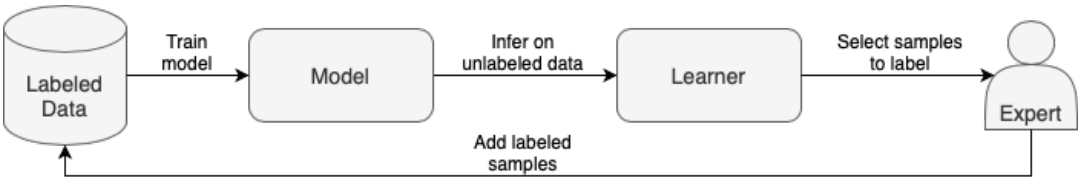


FIGURE 2.1: Graphical representation of the basic AL framework.

Streaming framework

In stream-based AL, the learner receives one sample at a time and determines if the instance should be labeled by an expert or not. This approach is cost-effective as it evaluates only individual data points sequentially, making it well-suited for real-time scenarios. However, a significant limitation of this framework is its potentially constrained performance, given that data points are evaluated sequentially and decisions are made independently for each one for them, without considering broader context among instances.

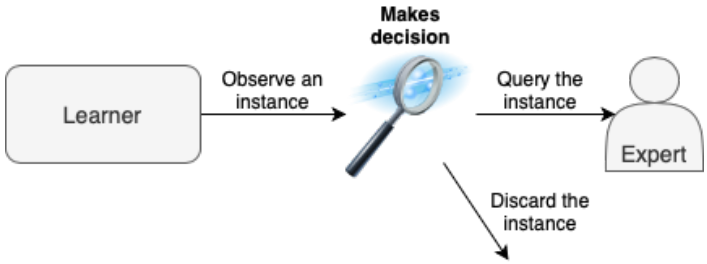


FIGURE 2.2: Stream-based AL framework.

Pooling framework

In pooling-based AL, the learner receives a batch of N instances from the unlabeled data set and identifies the most informative instance within the batch for expert labeling. Unlike stream-based AL, where decisions are made on individual data points in a sequential manner, pooling-based AL evaluates the entire set of unlabeled data points during each iteration. While this approach demands more memory and may be less cost-effective due to increased computational requirements, it effectively addresses some of the performance limitations associated with stream-based AL by considering a broader context and facilitating more informed decisions.

FIGURE 2.3: Pool-based AL framework.

Due to the extended duration associated with model training and the relatively minimal impact of individual data points on model behavior, pooling-based AL is frequently employed in scenarios where batches of data are submitted for labeling instead of just one. This batch-based approach allows for more efficient utilization of computational resources and expert annotation efforts, as it enables the model to leverage collective insights from multiple data points simultaneously.

2.2.2 AL Sampling Strategies

In AL frameworks, one of the most crucial components is the learner. As previously discussed, the learner is responsible for assigning priority scores to unlabeled data points and determining whether expert review and annotation are necessary.

Various strategies can be employed when designing the learner, with two primary categories being uncertainty sampling and diversity sampling strategies.

Uncertainty sampling

Uncertainty sampling centers on selecting samples for which the current model exhibits the highest level of uncertainty. Typically, when a supervised machine learning model makes predictions, it provides confidence metrics or probabilities associated with those predictions. The underlying principle of uncertainty sampling is to seek expert annotation for instances where the model's uncertainty is greatest, with the belief that obtaining feedback on these challenging samples will enhance the model's performance in such cases.

In the case of classification problems, the predictions of a model are a probability distribution, so every prediction is between 0 and 1, and the individual predictions for each class add up to 1. Let $Y = \{y_1, \ldots, y_n\}$ be the set of available classes in an multi-class clasification problem with n classes. Let U denote the collection of unlabeled instances of data. Then, some common uncertainty measures are:

- Least Confidence sampling: measures the distance from the most confident prediction to 100% confidence. Then, this sampling strategy selects the next instance to query for labels following:

$$x^* = \arg\max_{x \in U} \left(1 - \max_{y \in Y} P(y|x)\right),$$

  where $P(y|x)$ denotes the probability that instance x has label y. Consequently, this strategy measures uncertainty as the difference between 1, 100% confidence, and the most confidently predicted label for each item.

- Margin of Confidence sampling: measures the difference between the top two most confident predictions. The margin of confidence sampling strategy selects the next data point in an AL algorithm by solving the following problem:

$$x^* = \arg\max_{x \in U} \Big(1 - \big(P(y_m|x) - P(y_p|x)\big)\Big),$$

  where $y_m = \arg\max_{y \in Y} P(y|x)$ and $y_p = \arg\max_{y \in Y \setminus y_m} P(y|x)$.

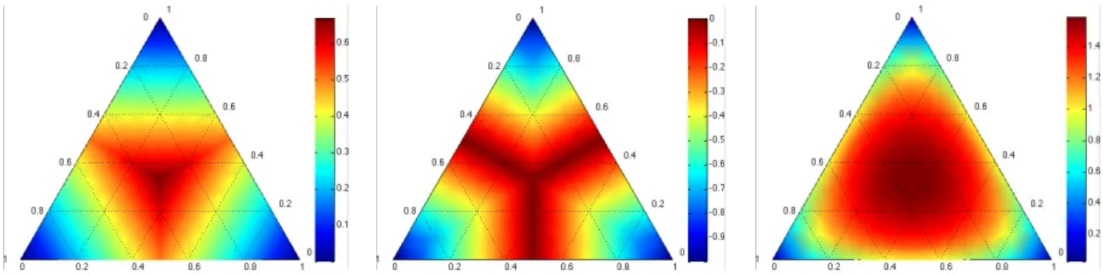
- Ratio of Confidence sampling: measures the ratio between the top two most confident predictions. The ratio of confidence, following the same notation as the margin of confidence strategy, solves the following problem:

$$x^* = \arg\max_{x \in U} \frac{P(y_p|x)}{P(y_m|x)}.$$

- Entropy sampling: measures the difference between all predictions. Entropy is a concept borrowed from information theory. It is frequently used as a measurement of impurity, disorder or randomness. In this case, it measures the disorder of a probability distribution. Entropy sampling selects the next point to query for labels by solving:

$$x^* = \arg\max_{x \in U} \frac{-\sum_{y \in Y} P(y|x) \log(P(y|x))}{\log_2(n)}.$$

Figure 2.4 shows three heatmaps that illustrate the differences between least confident, margin of confidence and entropy sampling strategies in a three-class classification problem. Notice that in all three cases, the most informative data points are found in the center of the triangles, that is where the posterior label distribution is most uniform. Correspondingly, the least informative data points are at the three corners, since this is where one of the classes has a very high probability assigned. The main differences between these three strategies lie in the rest of the probability space. Looking at the heatmap of entropy sampling, it shows that all the outer side edges are coloured as yellow or green, indicating that data points where only on of the classes is highly unlikely are not considered informative. On the other hand, the least confident and margin confidence strategies consider these data points to be informative for the model since it is not able to distinguish between the remaining two classes.



(A) Least confident (B) Margin confidence (C) Entropy

FIGURE 2.4: Comparison of uncertainty-based sampling functions in a three-class classification setting. Taken from (Settles, 2009b).

Diversity sampling

Diversity sampling selects data-points in order to maximize diversity, so it targets gaps in the current model's knowledge. There are four different strategies for diversity sampling:

- Model-based outliers: this method involves identifying samples that produce low activations or responses in the model's logits and hidden layers. Low activation implies that the model finds these samples confusing or ambiguous because they might represent data points on which the model lacks sufficient information or has not been trained adequately.
- Cluster-based Sampling: unsupervised machine learning techniques like clustering (e.g., k-means clustering) are employed to partition the feature space of the data into distinct clusters. Once clusters are formed, samples are selected from each cluster to ensure diversity.
- Representative Sampling: focuses on selecting samples that are most indicative or representative of the target domain you aim to model, relative to the existing training data. This method considers the distribution and characteristics of the target data when selecting samples.
- Real-world diversity: emphasizes sampling strategies that prioritize fairness, inclusivity, and representation of various demographic, cultural, or socio-economic groups in the data, trying to support real-world diversity.

In contrast to uncertainty sampling strategies, for which there exist some specific metrics that clearly measure uncertainty in predictions, diversity sampling approaches are more exploratory and diverse in nature. While the strategies mentioned above suggest different applications of diversity sampling, they do not prescribe specific methods due to the focus of diversity sampling on addressing "unknown unknowns."

Chapter 3

Method

3.1 Dataset

For this research, we had access to 45,851 Wireless Capsule Endoscopy (WCE) images. These images are derived from 23 distinct WCE videos, with each video containing approximately 2,000 images. All these images were labeled and categorized into one of six classes: bubbles, high turbidity, large blob, wall, wrinkles, and undefined. Sample images representing each class can be viewed in Figure 3.1.

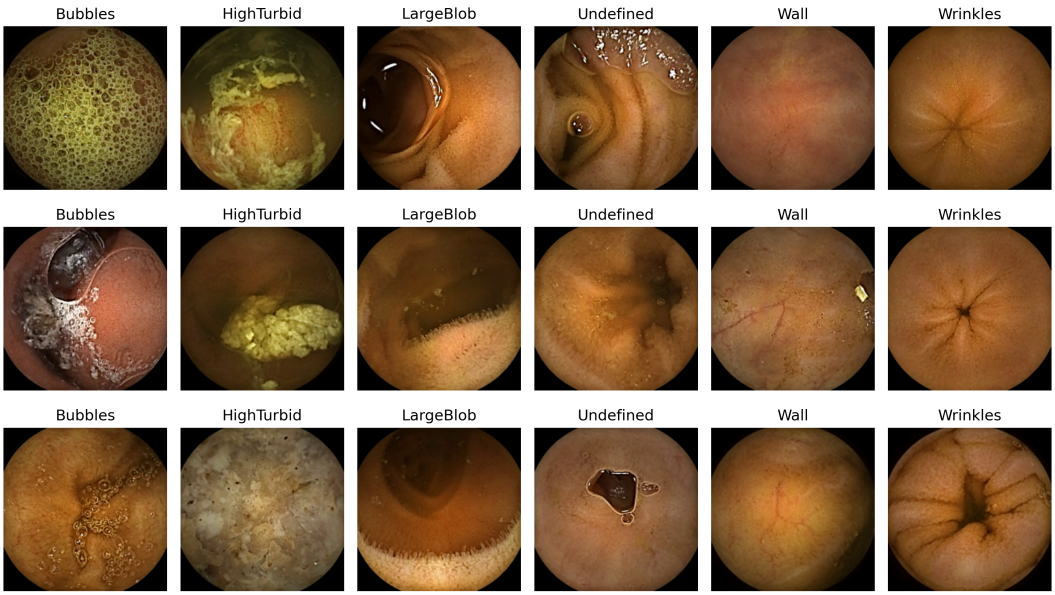


FIGURE 3.1: WCE samples images and their corresponding labels.

Analysing the number of images of each of these classes for every patient allows to detect bowel motility problems. Bowel motility refers to the synchronized movements of the stomach and intestine. Functional gastrointestinal disorders, such as functional dyspepsia and gastritis, are often associated with poor bowel motility, so the study of these classes can help in detecting these health issues.

3.1.1 Train-test Split

Given the availability of 23 videos, the data was partitioned as follows:

- Training set: This set consists of 13 videos. Initially, one video serves as the labeled dataset, while the remaining 12 videos are treated as unlabeled. These

unlabeled videos are the candidates for querying to obtain their respective labels.

- Validation set: This set encompasses 5 videos. Its primary purpose is to assess the efficacy of the model and validate whether the proposed strategies yield favorable outcomes in terms of performance enhancement.
- Test set: This set also consists of 5 videos. The videos in this set are not used by the model in any way and are separated only to evaluate the selected sampling strategy.

3.2 Problem Statement

As it was previously mentioned in Chapter 1, our project focuses on establishing an AL strategy tailored for WCE image classification. To achieve this goal, we execute multiple training cycles. Initially, we begin with a small subset of the data labeled (or, in our context, considered labeled). Subsequently, in each cycle, we incrementally incorporate additional data into the labeled dataset, enabling us to retrain the model from the ground up. Our primary emphasis lies in the selection of new data during this iterative process, as we explore various strategies to determine which approach optimizes model performance more efficiently. Figure 3.2 shows a diagram for the basic flowchart of our AL algorithm. This diagram shows in orange the step in which we use different strategies to test our AL method.

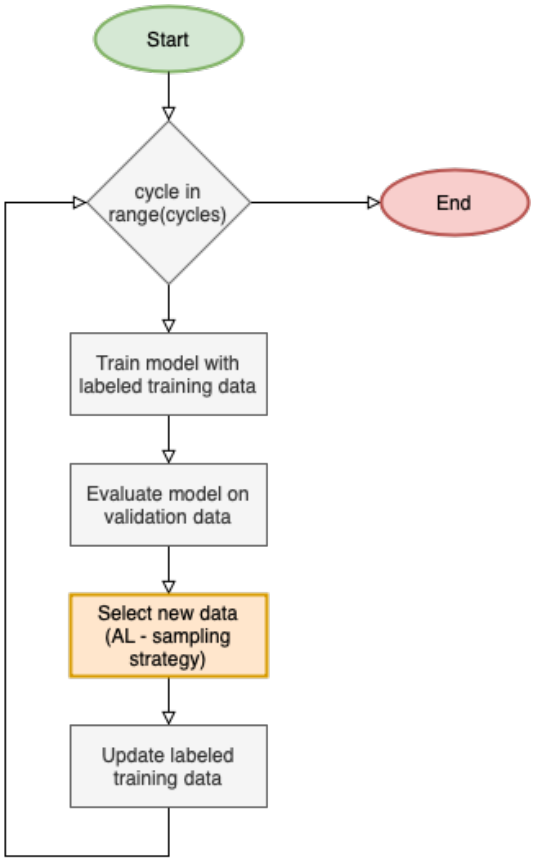


FIGURE 3.2: AL algorithm flowchart.

As discussed in Chapter 2, when operating within an AL framework, utilizing batches of data for labeling is frequently favored over selecting individual data points. In this research, our focus is on devising strategies that guide us in identifying the most appropriate video to query for labeling, optimizing the efficiency and effectiveness of the AL process. Therefore, whenever a new video is selected, about 2000 images are added to the labeled training images set.

To assess the effectiveness of a strategy, we are benchmarking it against a random sampling approach. In this comparison, during each cycle where a new video selection is required, a video is randomly chosen from the available list. When evaluating alternative strategies, we analyze the rate of enhancement in the model's performance using various performance metrics. Figure 3.3 presents three distinct plots, each illustrating the progression of a specific metric throughout the cycles of the AL algorithm. Consequently, our objective is to identify a strategy that produces more favorable outcomes than the random sampling approach. Specifically, for the loss metric, a superior strategy would yield a curve below that of the random strategy, whereas for accuracy and AUC metrics, an effective strategy would result in a curve surpassing that of the random strategy.

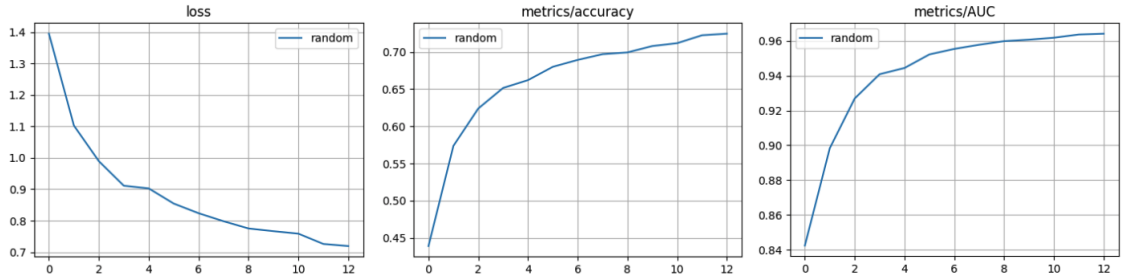


FIGURE 3.3: Plots showing the evolution of the model's performance through the different cycles of the AL algorithm.

3.3 Base Model

In testing all the proposed strategies, we have kept the foundational model within the AL algorithm consistent. Specifically, we utilized the EfficientNet architecture (Tan and Le, 2019), a convolutional neural network design introduced in 2019 aimed at enhancing both model accuracy and computational efficiency. The original publication presented a series of eight EfficientNet models, ranging from B0 to B7, distinguished by varying parameter counts. For this project, we have used the EfficientNetB3 model for our analyses and evaluations.

Also, the EfficientNetB3 model used is pre-trained with the ImageNet dataset instead of using randomly initialized weights.

3.4 Greedy Strategy

We begin with a straightforward greedy strategy. In each cycle, we test all available videos to see which one improves the most the model's performance. Once we pinpoint the most impactful video, we add it to our labeled dataset, aiming to boost the model's accuracy and effectiveness. Figure 3.4 illustrates the flowchart of this strategy.

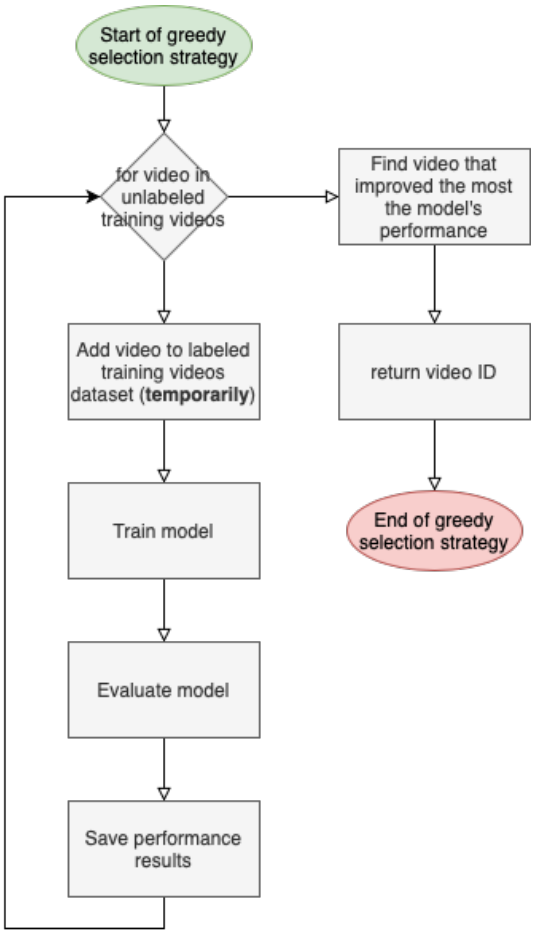


FIGURE 3.4: Flowchart of the greedy strategy.

To determine the video that most significantly enhances the model's performance, we employ the Area Under the Curve (AUC) score, specifically derived from the Receiver Operating Characteristic (ROC) curve. The AUC score quantifies the model's ability to distinguish between the positive and negative classes across various classification thresholds. Essentially, the ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR), offering a graphical representation of the model's discriminatory power. A higher AUC score indicates superior model performance, with a perfect classifier yielding an AUC of 1, while a random classifier would yield an AUC of 0.5. The reasoning behind prioritizing the AUC metric over conventional accuracy metrics lies in its robustness, particularly for datasets characterized by significant class imbalances. Accuracy metrics may be misleading in such scenarios, whereas the AUC metric provides a more comprehensive evaluation, making it more suitable for datasets where class imbalances could potentially skew results and misrepresent actual model performance.

The reason we are using this greedy selection strategy is pretty straightforward: we want to see the best we can get out of our model using the videos we have. By picking the video that gives us the most improvement each time, we are aiming to push our model to its limits with the data we have got. This approach helps us figure out how well our model can perform under the best circumstances possible, given our current setup. In simpler terms, we're trying to see the highest performance we can achieve with our model and dataset before considering any other sampling

strategies, thus obtaining some sort of upper bound on the performance improvement that we are looking for with the strategies we devise.

Using the greedy selection approach is logical as it aims to maximize our model's performance with the available videos. However, this method is not practical for real-world applications due to its significant computational demands. Furthermore, it would not be possible since in a real-case scenario we would not have access to the labeled data immediately. Each time we select a video, the need to train an entirely new model consumes extensive time and resources. Therefore, while it offers insights into the model's potential, it becomes impractical for larger datasets or more intricate tasks. In essence, while effective for assessing peak performance, its computational requirements make it unsuitable for routine use.

3.5 Uncertainty Strategies

This section introduces various uncertainty-based strategies that have been implemented, offering insights into the motivation behind their adoption.

3.5.1 Logits Strategy

The logits sampling strategy aims to identify the video with the highest uncertainty by assessing the least confidence in the model's predictions across its frames. In this approach, each video's uncertainty is evaluated by computing the average least confidence (highest uncertainty) of its frames. Specifically, for every frame in a video, it looks at the label assigned to it and with what probability. Then, the average of these probabilities is done. Ultimately, the video exhibiting the highest average uncertainty, indicative of the model's least confidence in its predictions across frames, is selected. Following the same notation as in Chapter 2, let $Y = \{y_1, \ldots, y_n\}$ be the set of available classes in a multi-class classification problem with n classes. In our case $n = 6$. Let U denote the collection of unlabeled instances of data. To adapt the notation to the structure of our data, $U = \{U_1, \ldots, U_m\}$ where each U_i is a video and is defined as a set of frames $\{x_j^{(i)} | x_j^{(i)} \in U_i\}$. Then, the calculation used by this strategy to choose the next video to query for labels is given by:

$$U^* = \arg\min_{U_i \in U} \frac{1}{|U_i|} \sum_{x^{(i)} \in U_i} \left(\max_{y \in Y} P(y|x^{(i)}) \right)$$

This sampling strategy uses an uncertainty score inherently generated by the model itself. Therefore, the strategy works under the assumption that the model possesses a proficient capability for self-evaluation. The core hypothesis behind this approach is that prioritizing novelty is crucial. This implies that the model benefits significantly from encountering and grappling with more complex and challenging instances. Rather than reinforcing existing knowledge or patterns, the strategy prioritizes the continual incorporation of novel concepts and complexities.

3.5.2 Margin Strategy

The margin sampling strategy is based on discerning the uncertainty of a model's predictions by evaluating the gap between the probabilities of the most probable class and the next highest probable class for each data instance. In essence, this approach emphasizes the distinction or "margin" between the top two predicted classes, aiming to highlight instances where the model's decision is less definitive.

Following the same notation that was defined in the previous section, let $Y = \{y_1, \ldots, y_n\}$ be the set of available classes in a multi-class classification problem with n classes. Let U denote the collection of unlabeled instances of data. To adapt the notation to the structure of our data, $U = \{U_1, \ldots, U_m\}$ where each U_i is a video and is defined as a set of frames $\{x_j^{(i)} | x_j^{(i)} \in U_i\}$. Then, this sampling strategy determines the next video to query for labels by solving the following problem:

$$U^* = \arg\min_{U_i \in U} \frac{1}{|U_i|} \sum_{x^{(i)} \in U_i} \left(P(y_m|x^{(i)}) - P(y_p|x^{(i)}) \right),$$

where $y_m = \arg\max_{y \in Y} P(y|x^{(i)})$ and $y_p = \arg\max_{y \in Y \setminus y_m} P(y|x^{(i)})$.

Whereas the logits sampling strategy solely focuses on the predicted probability of the most likely class, the margin strategy extends its evaluation by also taking into account the predicted probability of the second most probable class. By incorporating this additional dimension, the margin strategy provides a more informative and comprehensive assessment of the model's confidence levels across its predictions, although it still doesn't consider the third to the nth confidences. This approach becomes particularly valuable when dealing with overconfident models. By scrutinizing the margin between the top two predicted classes, the strategy can effectively identify instances where the model's confidence might be misleadingly high, but with little difference to the confidence with which another class has been predicted.

This strategy is intuitive in its approach. It categorizes examples with larger margins as straightforward for the model, as it effectively distinguishes them from the next most probable class. Conversely, examples with narrower margins are deemed more ambiguous, providing the model with opportunities to refine its distinctions between two closely competing classes. Therefore, this strategy focuses in ambiguous regions of the feature space and aims to better separate these areas.

3.5.3 Entropy Strategy

The entropy sampling strategy uses the concept of entropy from information theory to determine the next video to query for labels. Entropy is a measure that represents the amount of information needed to "encode" a distribution (Settles, 2010). Again, we follow the same notation as in the previous sections, so this sampling strategy selects the next video by solving the following problem:

$$U^* = \arg\max_{U_i \in U} \frac{1}{|U_i|} \sum_{x^{(i)} \in U_i} \left(- \sum_{y \in Y} P(y|x^{(i)}) \log(P(y|x^{(i)})) \right),$$

Where the logits and margin strategies consider only one or two most probable classes for sampling, entropy sampling considers the confidence awarded to all classes available. Entropy provides a quantitative measure of the uniformity or spread of the predicted class probabilities. Instances with low entropy have predicted class probabilities that are concentrated on one or a few classes, indicating that the model is more confident in its prediction for these instances. Conversely, a higher entropy suggests a more uniform distribution of probabilities, meaning that predicted class probabilities are spread out across multiple classes, indicating uncertainty or ambiguity in the model's prediction.

3.6 Diversity Strategies

This section presents a range of diversity-based strategies that have been implemented, providing an understanding of the rationale for their incorporation.

3.6.1 Cover Strategy

Following the idea proposed in (Gilabert et al., 2023) the cover strategy employs embeddings of data points to quantify the "diversity" they introduce to the labeled dataset. To initiate this process, embeddings for the labeled data are computed, forming the foundation for constructing a KDTree based on these embeddings. Subsequently, for each video, embeddings are computed for all its frames. Utilizing the KDTree, the strategy determines the distances from these video frame embeddings to their 30 closest neighbors within the labeled dataset. By aggregating these distances, the average is calculated for each frame. To determine the overall diversity contribution of a video, the mean distance is derived from the averaged distances of its constituent frames. Ultimately, the strategy identifies the video that exhibits the highest mean distance as the one to be incorporated next, thereby emphasizing diversity in the labeling process.

The cover strategy is designed with a specific objective in mind: to pinpoint regions within the embedding space that remain unexplored or inadequately represented by the existing labeled dataset. By leveraging this approach, the strategy aims to augment the coverage of the labeled data. Essentially, it prioritizes the incorporation of data points that reside farthest from the current boundaries of the labeled space. In doing so, the strategy seeks to enhance the diversity and richness of the labeled dataset, ensuring a more comprehensive representation across the embedding landscape.

Embedding calculation

To derive embeddings for the data points, an intermediary model is used and plays an important role in the results of this strategy. Two distinct methodologies have been employed for this purpose. Firstly, the initial approach harnesses the layer immediately preceding the final dense layer equipped with the softmax activation function. This layer is extracted from the model trained on the most recent pool of labeled data.

Conversely, the second approach, uses an autoencoder model to generate embeddings for all available data points, encompassing both labeled and unlabeled instances. Autoencoder are an unsupervised learning technique that take advantage of neural networks for the task of representation learning. More concretely, autoencoders are a neural network that compress the knowledge from the initial output and then aim to reconstruct it based on its compressed version. As we have mentioned, autoencoders follow an unsupervised learning technique, since they are trained by minimizing the reconstruction error, which measures the differences between the original input and the consequent reconstruction. In short, autoencoders are known for their ability to compress and reconstruct data, they provide a holistic embedding that encapsulates the intrinsic characteristics of each data point.

There are major differences between these two approaches that may favor one resulting in a better performance than the other. The first strategy recalculates new embeddings in each cycle using the most recently trained model. Consequently, the

embedding space from which the cover strategy selects the subsequent video for labeling undergoes changes whenever fresh data integrates into the labeled dataset. This approach may exhibit some initial unreliability within the AL algorithm, particularly when the model operates with limited data. In contrast, the utilization of the autoencoder model ensures consistency in embeddings throughout the entirety of the AL algorithm. Since autoencoders are trained in an unsupervised way, their performance does not depend on the amount of labeled data that there is available. Therefore, the autoencoder model is trained independently from the AL framework and is used at the beginning of the AL algorithm to obtain the embeddings of all the data points, providing reliable and constant representations of the data points. This consistency facilitates the strategy in identifying areas to enhance the model's understanding, relying on a more dependable and steadfast latent space.

3.6.2 Cloud Strategy

The cloud strategy is another diversity-based sampling strategy that leverages data point embeddings. The strategy starts by computing the embeddings of the labeled data and uses a dimension reduction technique to condense these embeddings into a 2-dimensional space. Then, using these 2-dimensional embeddings the convex hull of all the labeled training data points is determined. These steps are repeated for every video: the embeddings of the frames of the video are obtained and the same dimension reduction technique is used to reduce them to 2-dimensional embeddings. Then, the convex hull of the data points of the video is computed. The convex hull of the video frame is compared to the one of the labeled data points by computing the intersection are between both convex hulls. The cloud strategy considers that the smaller the intersection area, the more informative the video will be for the model.

To refine the cloud strategy's performance and prevent it from being overly influenced by outliers, specific hyperparameters are incorporated:

- sample_size: This parameter determines the portion of frames from a video used to compute its convex hull. A value of 1 implies all frames are considered, while fractional values mean a proportionate random selection.

- num_samples_per_video: This parameter dictates how many times the convex hull for a video is computed. Each iteration uses a fresh set of sampled points. The intersection of the convex hull obtained for each sample is done to obtain the final convex hull of the video.

This iterative approach helps mitigate potential distortions caused by outlier frames. By focusing on consistent convex hulls across multiple samples, the strategy emphasizes areas where a video likely contains significant frames rather than being skewed by isolated outliers.

The objective of the cloud strategy aligns closely with that of the cover strategy but employs a slightly different approach. While the cover strategy focuses on maximizing distances to determine the most informative data points, the cloud strategy emphasizes the spatial area by leveraging convex hulls. Similar to the cover strategy's aim to expand the convex hull of the labeled data to enhance the model's understanding, the cloud strategy seeks to amplify this area to achieve a similar outcome.

Embedding calculation

To determine embeddings within the cloud strategy, it adopts the same techniques used in the cover strategy. Specifically, the first approach leverages the last dense layer prior to the final softmax activation function from the model trained on the most recent labeled data. Conversely, the second approach harnesses an autoencoder model. This autoencoder is instrumental in deriving embeddings for all available data points, encompassing both labeled and unlabeled data.

Dimensionality reduction step

In the dimension reduction phase, the cloud strategy employs two distinct techniques: Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018).

PCA operates on the principle of linear dimensionality reduction by identifying axes, termed principal components, that maximize the variance within the data. Its primary objective is to condense the data into a lower-dimensional space while retaining the maximum amount of variability present. On the other hand, UMAP delves into non-linear dimensionality reduction. Rather than merely focusing on variance, UMAP emphasizes preserving both the local and global structures inherent in the data manifold. This distinction makes UMAP particularly adept at capturing intricate relationships and patterns within data sets that might not be discernible through linear methods alone.

3.6.3 Clustering Strategy

The clustering strategy adopts a distinct methodology to categorize data points, leveraging the capabilities of an autoencoder model to generate embeddings for both labeled and unlabeled data. Once these embeddings are derived, an unsupervised clustering approach, specifically K-means or Gaussian Mixture Models (GMM), is employed to cluster these data points based on their similarities. This clustering enables a structured representation of the data, grouping together points that exhibit similar characteristics. Given the distribution of the labeled data in the clusters and the distribution of the frames of a video, a score is computed using a pre-defined metric that determines the next video to query for labels.

The core objective behind utilizing these clustering methods within the strategy is to test the hypothesis that videos with frames in a large number of clusters are more informative than videos with most of the frames in just a few clusters.

Clustering strategies

In order to cluster the embeddings, we have used two different clustering algorithms: KMeans and Gaussian Mixture Models.

The main difference between K-Means and Gaussian Mixture Models (GMMs) lies in their assumptions and resulting outcomes. Visually, Gaussian Mixtures have more flexible decision boundaries due to their ability to form elliptical shapes using covariance matrices, whereas K-Means strictly creates circular boundaries. Additionally, GMMs are probabilistic in nature. This means that for each data point, GMMs provide a quantifiable measure, indicating the likelihood or confidence of its association with a particular cluster. In contrast, K-Means offers definitive assignments, placing each data point squarely within one cluster without ambiguity. These

distinctions underscore that while both methods aim to segment data into clusters, they employ different foundational principles and methodologies.

By employing both strategies, we aim to discern which approach aligns more effectively with the inherent characteristics and structure of our data.

Scoring metrics

With regards to the scoring metrics employed in this strategy, three distinct metrics have been implemented to evaluate the distribution and diversity of videos across various clusters. The following notation will be used to explain the scoring metrics: let $Y = \{y_1, \dots, y_n\}$ be the set of available classes in a multi-class classification problem with n classes. Let U denote the collection of unlabeled instances of data. To adapt the notation to the structure of our data, $U = \{U_1, \dots, U_m\}$ where each U_i is a video and is defined as a set of frames $\{x_j^{(i)} | x_j^{(i)} \in U_i\}$. Let $C = \{C_1, \dots, C_s\}$ denote the set of clusters. We will denote by $U_{i|C_j}$ as the set of frames of U_i that have been assigned to cluster C_j.

- **Entropy Metric:** This metric is based on the same concept as the entropy sampling strategy explained in Section 3.5.3. It leverages the concept of entropy to measure the uniformity of the distribution of the frames of a video in the different cluster to determine if a video is scattered in several clusters or if most of its frames can be found in a few clusters. Following this scoring metric, the video to query for labels at each cycle is determined with the following formula:

$$U^* = \underset{U_i \in U}{\arg\max} \left(- \sum_{C_j \in C} \frac{|U_{i|C_j}|}{|U_i|} \log \left(\frac{|U_{i|C_j}|}{|U_i|} \right) \right)$$

- **Gini Metric:** This metric leverages the concept of the Gini impurity. The Gini impurity is a measure used to evaluate how pure a set of data is in terms of its class labels. A Gini impurity of 0 indicates perfect purity (all items in the set belong to the same class). Whereas, a Gini impurity of 1 indicates maximum impurity (the items are evenly distributed across all classes). Therefore in this case we are looking for videos with a high Gini impurity, since this means that frames are distributed across several clusters. Following this idea, this scoring metric uses the following problem to determine the next video to query for labels:

$$U^* = \underset{U_i \in U}{\arg\max} \left(1 - \sum_{C_j \in C} \left(\frac{|U_{i|C_j}|}{|U_i|} \right)^2 \right)$$

- **Log Metric:** This metric combines cluster weights with the logarithmic transformation of video frame distributions, aiming to prioritise videos with frames in many cluster, while emphasizing clusters with lower representation in the current pool of labeled data. The following formula shows the problem solved by this metric to choose the next video to query for labels:

$$U^* = \underset{U_i \in U}{\arg\max} \left(\sum_{C_j \in C} w_j \log \left(\frac{1 + |U_{i|C_j}|}{|C_j|} \right) \right), \text{ where } w_j = 1 - \frac{|C_{j_{|labeled}}|}{|C_j|}.$$

  By calculating the log transformation of the proportion of frames per cluster and summing them up, it prioritizes videos that have frames spread across

multiple clusters. Then, by adding the w_i weights, whose values are based on the proportion of labeled data in the corresponding cluster, it gives more importance to clusters with a lower ratio of labeled data points, Therefore, it aims to maximise the diversity included by the AL algorithm at every cycle.

Each of these metrics uses a different concept to score the videos, while still maintaining the same objective: to prioritise videos with frames belonging in a variety of clusters, and allowing us to see which concept is better suited for our problem in specific.

Chapter 4

Results

4.1 Working Environment

This project has been developed mostly in Jupyter notebooks. We did our coding in Python and used tools like TensorFlow, Keras, and NumPy for tasks and graphs. Since running an AL algorithm is highly computationally intensive, we have had access to a GPU, specifically an NVIDIA GeForce RTX 3090 to speed up our work and enhancing our ability to efficiently train and optimize deep learning models.

4.2 Experiments

This section outlines the experiments conducted in this research and their outcomes. We intend to showcase the results from implementing all the strategies introduced in Chapter 3 and provide comparisons between the outcomes of each strategy.

As it was explained in Section 3.1.1, our training set consists of 13 videos. Since one is used as the initial labeled dataset, there are 12 remaining videos, meaning that in our AL framework we can perform up to 12 cycles. In Figure 3.3 we showed the results obtained with the maximum of 12 cycles with the random strategy. Given that performance typically stabilizes after 8 cycles, we limited subsequent experiments to this number to optimize computational efficiency.

Furthermore, the performance curves presented in this section represent the average of three trials for each strategy. This means that the AL algorithm was executed three times for each strategy, and the graphs reflect the mean results from these trials.

4.2.1 Greedy Strategy

The initial strategy we implemented is the greedy approach. This method establishes an upper limit for the performance we aspire to achieve with a sampling strategy that will still remain feasible for real-world applications. Figure 4.1 illustrates the performance evolution curves of the model across the cycles of the AL algorithm. This plot shows that where the greedy strategy rapidly increases the model's performance in the first 3 cycles and then quickly stabilizes, the random strategy makes the model's performance improve more slowly. Therefore, for the rest of the experiments we will be using the curve obtained with this strategy as an upper bound of the performance evolution curve that can be reached with the data set available, and the curve obtained with the random sampling strategy as a lower bound of the performance evolution curve of a sampling strategy that we will consider successful.

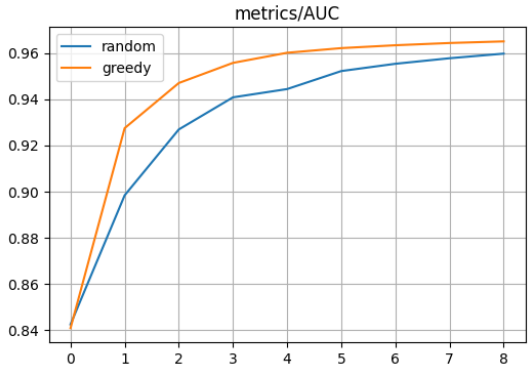


FIGURE 4.1: Plot of the performance evolution curves with respect to the AUC metric of the greedy and random strategies.

4.2.2 Uncertainty Strategies

Next, we implemented some uncertainty-based metrics. This section presents the results obtained with some of the most common uncertainty-based sampling strategies in AL and that have been explained in Section 3.5.

Figure 4.2 shows a comparison of the performance evolution curves obtained with the three different uncertainty-based sampling strategies that have been implemented: logits, entropy and margin sampling. As we can see, we can not express with confidence that one strategy results in a better performance than the other two since the three curves grow at a similar speed throughout the cycles in the AL algorithm. Still, we can see that the margin and logits sampling strategies have a slightly faster improvement in cycles 1 through 4. However, by cycle 4, the entropy sampling curve begins to closely align with the other two strategies.

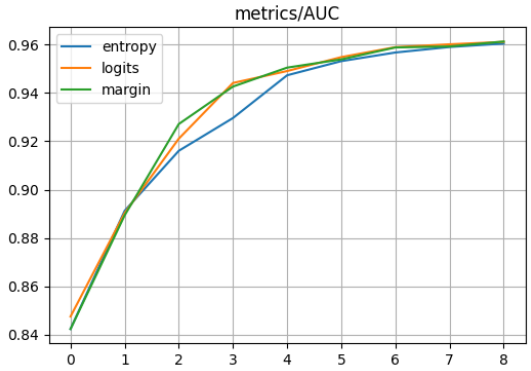
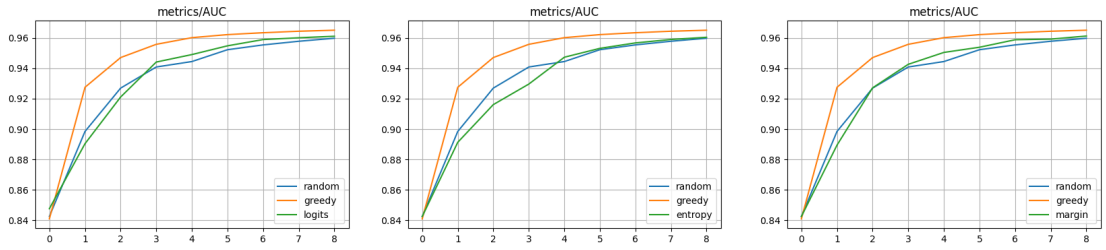


FIGURE 4.2: Plot of the performance evolution curves with respect to the AUC metric of the uncertainty-based strategies.

Given that we don't have a distinct strategy that stands out as the clear winner among the three implemented approaches, in Figure 4.3 compare each of them to the upper and lower bounds set by the greedy and random strategies respectively. In all three plots we can see that neither strategy gives a performance in between these bounds, all of them having a similar evolution to the one obtained by the random strategy.

One potential reason why the uncertainty-based sampling strategies might not exhibit a performance similar to the greedy or even superior the random sampling



(A) Logits (B) Entropy (C) Margin

FIGURE 4.3: Plots of the performance of each of the uncertainty-based strategies against the performance curves of the random and greedy strategies.

strategy is likely because of the videos themselves. While certain videos might register as highly uncertain to the model, their frames could present a consistent and limited variety. Given the design of our AL framework, where we select entire videos rather than individual frames, such videos could receive elevated uncertainty scores if their predominant frame type isn't represented in the current labeled dataset. Consequently, adding such videos to the labeled pool would not significantly augment the model's knowledge. Fundamentally, incorporating nearly identical frames from these videos offers minimal new insights, questioning the value of adding about 2000 similar frames to our labeled dataset.

4.2.3 Diversity Strategies

Given the conclusions drawn from the previous section, we then explored diversity-based sampling strategies. With this, we aimed to avoid the observed issue with uncertainty-based sampling strategies where the selected videos predominantly contained similar frames, thereby contributing minimal new information to the model's understanding.

To address this, we introduced three new sampling strategies: cover, cloud, and clustering techniques. Each strategy was developed with a distinct rationale and aims to address the previously highlighted issue in its own unique manner.

Cover strategy

The cover strategy attempts to solve the issue by using distance in the embedding space as a measure of diversity with respect to the data that can already be found in the labeled pool of data. As it has already been mentioned, this strategy leverages the information from the embeddings of the frames.

Initially, we used the model trained on the updated labeled data to obtain an intermediate model that predicts embeddings of the frames rather than the specific classes. However, this approach could produce unreliable embeddings, especially during the early stages of the AL algorithm when the model hasn't accumulated sufficient training data. To address this, we develop an alternative approach: instead of generating embeddings with the most recent model, we use an autoencoder model. This ensures consistency in the embeddings throughout the entire AL process. This variant of the cover sampling strategy is the cover autoencoder strategy.

In Figure 4.4, we present a comparison of the performance evolution curves between the cover strategy and the cover autoencoder strategy. Clearly, the cover autoencoder approach demonstrates superior results. Specifically, the cover strategy

employing the autoencoder rapidly improves the model's performance within the initial three cycles, followed by stabilization. In contrast, the standard cover strategy requires more cycles to achieve comparable efficiency and stability.

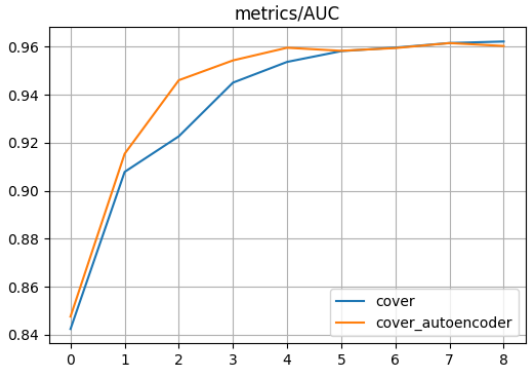
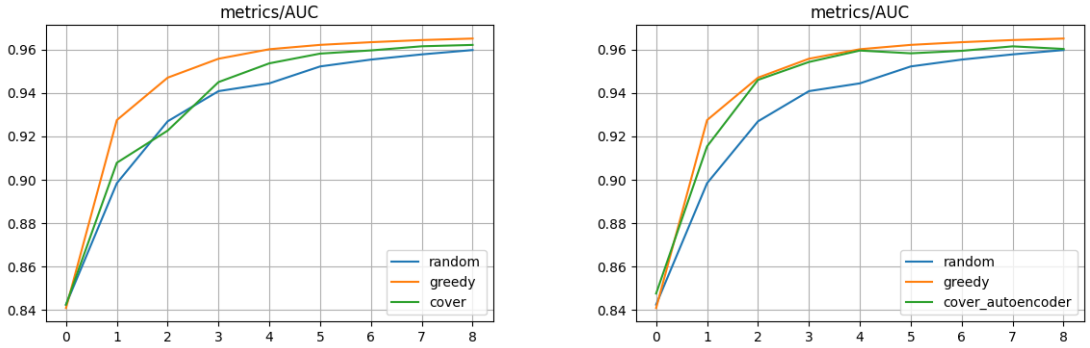


FIGURE 4.4: Plot of the performance evolution curves with respect to the AUC metric of all the cover strategy variants.

Figure 4.5(A) displays the performance evolution curve of the cover strategy, compared with the upper and lower bound curves established by the greedy and random strategies, respectively. It's evident that while the cover strategy outperforms the random approach marginally, the improvement isn't substantial, as their curves intersect and overlap over several cycles. In contrast, Figure 4.5(B) showcases the performance evolution curve of the cover autoencoder sampling strategy, again in comparison with the curves of the greedy and random strategies. Notably, the cover autoencoder strategy demonstrates significantly enhanced performance over the random strategy. Moreover, its curve closely aligns with the upper boundary set by the greedy strategy, reaching comparable performance levels at certain cycles.



(A) Cover

(B) Cover Autoencoder

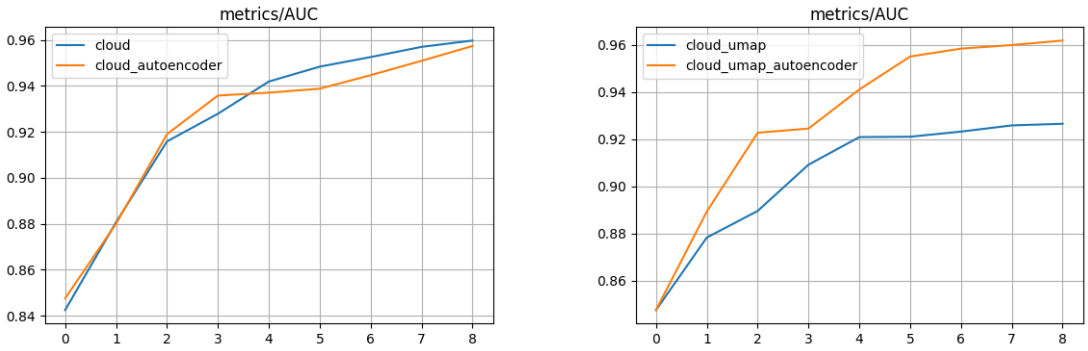
FIGURE 4.5: Plots of the performance of each of the cover strategy variants against the performance curves of the random and greedy strategies.

Cloud strategy

Although the cover sampling strategy does target videos that are diverse with respect to the data already in the pool of labeled data, it does not take into consideration the amount of diversity that is being added. For this reason, we devise the cloud sampling strategy. The idea behind this strategy is to use the intersection area

between the labeled data and the frames of the video, instead of using only the distance.

In Figure 4.7 we show the performance of the four variants of the cloud strategy. The plot to the left shows the performance evolution curves of the cloud variants that use PCA for the dimension reduction of the embeddings. We can observe that in this case, both sampling strategies, cloud and cloud autoencoder have a similar curve. Then, on the plot on the right we show the performance evolution curves of the cloud strategy variants that use UMAP for the dimension reduction. It's evident by observing this plot that the cloud strategy that uses UMAP with the autoencoder model results in an infinitely better performance improvement than the same strategy that does not use the autoencoder model. Where the cloud UMAP autoencoder sampling strategy improves the model's performance throughout all the cycles of the AL algorithm, the cloud UMAP strategy stops improving the model's performance in cycle 4.



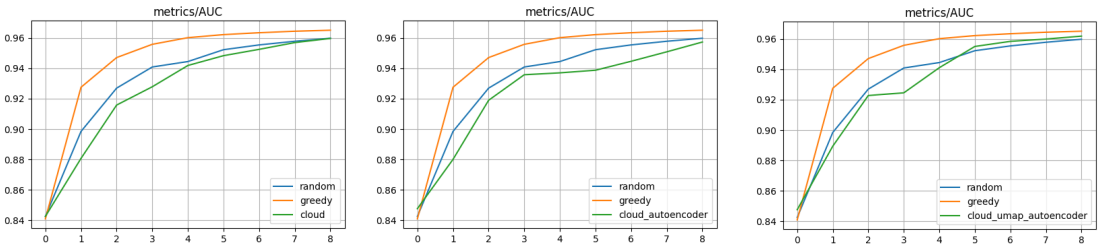
(A) With PCA (B) With UMAP

FIGURE 4.6: Plots of the performance evolution curves with respect to the AUC metric of all the cloud sampling strategy variants.

Following these results, Figure 4.7 compares the curves of the three best performing variants; cloud, cloud autoencoder and cloud UMAP autoencoder strategies, to the upper and lower bound performance evolution curves obtained with the greedy and random sampling strategies respectively. Regrettably, the outcomes we observed did not align with our initial expectations. The three plots show that neither the cloud strategy nor its variants, have an evolution better than the random sampling strategy. We notice however, that the cloud sampling strategy with the UMAP and autoencoder model has the best performance out of the four sampling strategies of this type, being the closest one to match the performance of the random strategy.

A possible explanation for the outcomes observed with the cloud strategy, and all its variants, could be attributed to the dimension reduction of the embeddings. Specifically, when employing the autoencoder model, the embeddings are condensed from a dimension of 256 down to just 2. In contrast, utilizing an intermediate layer from the last trained model reduces the dimensions from 64 to 2. This significant reduction in dimensions implies that a considerable amount of information is lost during this process. Such loss in critical information might be a leading factor contributing to the suboptimal performance of the cloud strategy.

However, despite the subpar results from the cloud strategy, when comparing the performance curves of the cover and cloud strategies, both of which incorporate variants that use the autoencoder model for obtaining data point embeddings, a notable trend emerges. Strategies employing the autoencoder model for embeddings



(A) Cloud (B) Cloud Autoencoder (C) Cloud UMAP Autoencoder

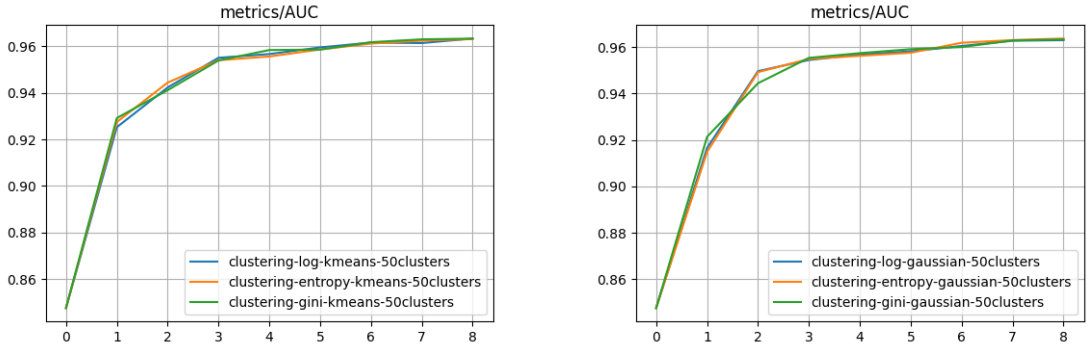
FIGURE 4.7: Plots of the performance of the cloud, cloud autoencoder and cloud UMAP autoencoder sampling strategies (from left to right) against the performance curves of the random and greedy strategies.

consistently match or outperform those utilizing the intermediate model from the last trained iteration.

Clustering strategy

The final strategy we implemented is the clustering strategy. This approach aims not only to introduce diversity into the labeled data pool but also to assess the degree of diversity being incorporated. The basis of this strategy is the clustering step, where all the embeddings, obtained from labeled and unlabeled data, is clustered. For these experiments, we have clustered the data into 50 clusters.

The clustering strategy has two main factors that can affect the strategy: the clustering strategy and the scoring metric. In Figure 4.8 the left plot shows the results obtained with the three different scoring metrics using KMeans as the clustering algorithm. The plot on the right shows the same results with a Gaussian Mixture Model for the clustering step of the sampling strategy. From these two plots, we can observe that using either clustering strategy, all three scoring metrics have an extremely similar performance, overlapping and crossing throughout all the cycles of the AL algorithm.



(A) With KMeans (B) With Gaussian Mixture model

FIGURE 4.8: Plot of the performance evolution curves with respect to the AUC metric of all the cover strategy variants.

Observing minimal differences in the performance evolution curves across the three distinct scoring metrics, Figure 4.9 compares the performance trends of the KMeans and Gaussian Mixture Model in the clustering step. This figure contrasts the evolution curves of the clustering strategy using the log scoring metric with both

clustering methods. Notably, the resulting curves closely resemble each other, showing nearly identical performance increments for both strategies in each cycle. Consequently, based on these findings, we cannot definitively conclude that one clustering method outperforms the other within this specific context and with our dataset.

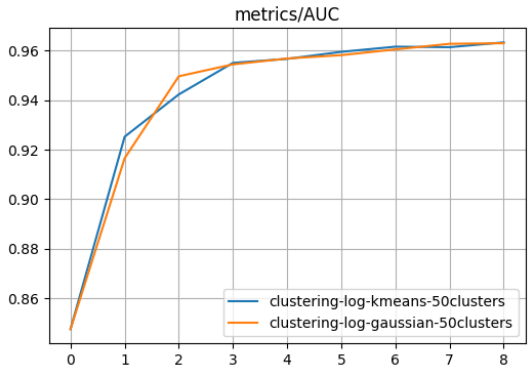
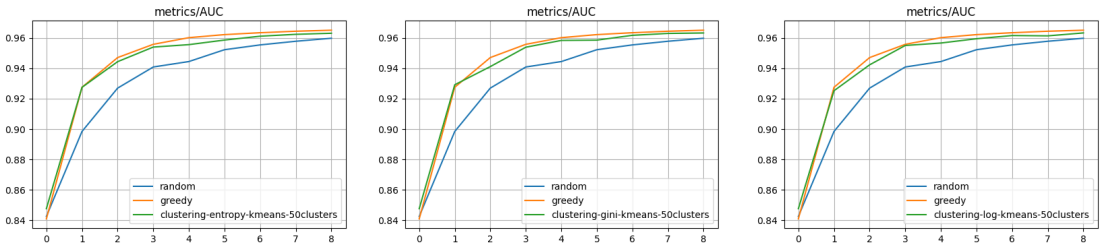


FIGURE 4.9: Plot of the performance evolution curves with respect to the AUC metric of the clustering strategy with the log scoring metric.

Ultimately, given the almost zero variance in the performance evolution curves between the two clustering algorithms, we use the results from the KMeans clustering algorithm to evaluate the success of the clustering sampling strategy. Our aim is to determine if this strategy can outperform the results achieved with the random sampling strategy and approach the performance curve of the greedy sampling strategy. For this purpose, Figure 4.10 compares the performance evolution curves of the clustering sampling strategy using KMeans and the three different scoring metrics; entropy, gini and log, to the upper and lower bounding curves given by the greedy and random sampling strategies respectively. These plots shows that the clustering sampling strategy, with either scoring metrics, clearly outperforms the random sampling strategy and almost matches the evolution obtained with the greedy sampling strategy.



(A) Entropy (B) Gini (C) Log

FIGURE 4.10: Plots of the performance of the clustering sampling strategy with the entropy, gini and log scoring metrics (from left to right) against the performance curves of the random and greedy strategies.

Best performing strategies comparison

Having explored various sampling strategies and dissected their outcomes, two sampling strategies have distinctly emerged with superior performance evolutions compared to others. To discern the most effective strategy for our objectives, we

compare these top-performing strategies — namely, the cover autoencoder and the clustering with KMeans using the log scoring metric — through a plot showcasing their performance evolution curves. While the various iterations of the clustering strategy showcase comparable results, we've opted for the KMeans variant with log scoring metric in our visualization to maintain clarity in the readability of the results for the reader.

In Figure 4.11, we present this comparison alongside the greedy and random sampling strategies. It's evident that the curves closely mirror the performance trajectory of the greedy strategy, showing consistent improvements with each cycle. This suggests that both strategies effectively serve as viable sampling methods for our AL algorithm. Yet, given the striking similarity in their curves, it's challenging to assert the superiority of one over the other; rather, both emerge as commendable sampling approaches.

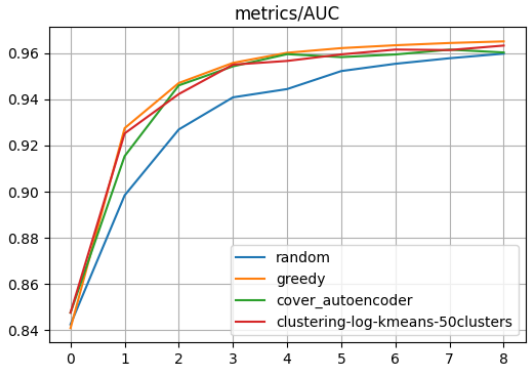


FIGURE 4.11: Plot of the performance evolution curves with respect to the AUC metric of the best performing sampling strategies: cover autoencoder and clustering with KMeans and log scoring metric against the performance curves of the random and greedy strategies.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this research, our main goal was to explore into different sampling strategies within AL, specifically within the realm of WCE image classification. Our aim was to establish an AL framework and explore the idea that by carefully selecting training samples, we could enhance the model's performance with minimal data.

To achieve these objectives, we utilized 23 WCE videos. Initially, we defined the operation of our AL algorithm. For this research, we opted to design an AL framework that consistently selects entire videos for querying, rather than individual frames that might span across different videos.

Next, we implemented a random sampling strategy and used it as our baseline to benchmark the progression and effectiveness of the AL algorithm. Additionally, to determine the peak performance enhancement achievable with the given data, we incorporated the greedy sampling strategy, setting it as our upper performance limit to strive for.

From the sampling strategies we introduced, we can categorize them into two primary types: uncertainty-based and diversity-based. Within the uncertainty-based category, we explored three distinct strategies: logits, entropy, and margin. As discussed in Chapter 4, the outcomes from these strategies fell short of our expectations, often only matching the performance improvements seen with the random sampling strategy.

Believing that the issue might be related to selecting videos with high uncertainty but minimal diversity, thus not significantly enhancing the model's understanding, we explored three diversity-based sampling strategies: cover, cloud, and clustering. Among these approaches, we observed improved results. While the cloud sampling strategy didn't outperform the random strategy, the cover and clustering strategies stood out. Specifically, the cover strategy, particularly with the autoencoder version, demonstrated notable results. Meanwhile, across the clustering strategy's various algorithm and scoring metric variants, we consistently observed successful outcomes, often rivaling the performance evolution seen with the greedy strategy.

While we've identified AL sampling strategies that meet our objectives, distinguishing between the efficacy of the cover with autoencoder and clustering strategies remains challenging. To definitively determine the superior strategy, additional research is necessary.

5.2 Future Work

Our AL algorithm selects whole videos for labeling in each cycle, reaching the upper performance limit with the greedy strategy. However, there's room to explore

a different approach. Instead of choosing entire videos, we can consider selecting individual frames from different videos in each cycle. Future studies can investigate into optimizing AL strategies for this frame-level approach, offering a chance to improve the performance evolution curves achieved by our video-level approach, with the potential to enhance the efficiency and effectiveness of AL in WCE images classification.

Appendix A

Source Code

A.1 Github Repository

All the code that was develop for this research can be found in the following GitHub repository

https://github.com/sarabase/ActiveLearning-WCE

The github repository contains the following files:

- **ActiveLearning-Benchmarks.ipynb**: executes the active learning framework using benchmarking strategies: random and greedy. It serves as a baseline comparison for other sampling strategies.
- **activelearning_utils.py**: contains auxiliary functions for the active learning algorithm's functionality. It includes functions for plotting results, as well as functions specific to the random and greedy sampling strategies.
- **ActiveLearning-Uncertainty.ipynb**: implements the active learning framework with uncertainty-based sampling strategies: least confident, margin and entropy.
- **activelearning_uncertainty.py**: contains the auxiliary functions for the uncertainty-based sampling strategies.
- **ActiveLearning-Cover.ipynb**: implements the active learning framework with the cover sampling strategy and its variants.
- **activelearning_cover.py**: provides the necessary functions for the cover sampling strategy and its variants.
- **ActiveLearning-Cloud.ipynb**: implements the active learning algorithm with the cloud sampling strategy and it variants.
- **activelearning_cloud.py**: contains the necessary functions for the cloud sampling strategy and its variants.
- **ActiveLearning-Clustering.ipynb**: executes the active learning algorithm with the clustering sampling strategy and test three different scoring metrics: entropy, gini and log.
- **activelearning_clustering.py**: contains the necessary functions for the clustering sampling strategy variants.

Note that due to the sensitive nature of the dataset, which contains clinical information of real patients, the data with which the experiments have been performed cannot be uploaded to this repository.

Also, since this is a non-publicable project, the repository is private. If you require access to the GitHub repository, please contact me at sbardase7@alumnes.ub.edu.

Bibliography

Aoki, Tomonori et al. (2019). "Automatic detection of erosions and ulcerations in wireless capsule endoscopy images based on a deep convolutional neural network". In: *Gastrointestinal endoscopy* 89.2, pp. 357–363.

Aoki, Tomonori et al. (2020). "Automatic detection of blood content in capsule endoscopy images based on a deep convolutional neural network". In: *Journal of gastroenterology and hepatology* 35.7, pp. 1196–1200.

Ciaccio, Edward J et al. (2010). "Distinguishing patients with celiac disease by quantitative analysis of videocapsule endoscopy images". In: *computer methods and programs in biomedicine* 100.1, pp. 39–48.

Falin, Zhuo, Liu Haihua, and Pan Ning (2022). "Gastrointestinal Polyps and Tumors Detection Based on Multi-scale Feature-fusion with WCE Sequences". In: *arXiv preprint arXiv:2204.01012*.

Gilabert, Pere et al. (2022). "Artificial intelligence to improve polyp detection and screening time in colon capsule endoscopy". In: *Frontiers in Medicine* 9, p. 1000726.

Gilabert, Pere et al. (2023). "Leveraging Embedding Information to Create Video Capsule Endoscopy Datasets". In: *2023 18th International Conference on Machine Vision and Applications (MVA)*, pp. 1–5. DOI: 10.23919/MVA57639.2023.10215919.

Hajabdollahi, Mohsen et al. (2018). "Segmentation of bleeding regions in wireless capsule endoscopy images an approach for inside capsule video summarization". In: *arXiv preprint arXiv:1802.07788*.

Iddan, Gavriel et al. (2000). "Wireless capsule endoscopy". In: *Nature* 405, p. 417. URL: https://doi.org/10.1038/35013140.

Jin, Eun Hyo et al. (2020). "Improved accuracy in optical diagnosis of colorectal polyps using convolutional neural networks with visual explanations". In: *Gastroenterology* 158.8, pp. 2169–2179.

Klang, Eyal et al. (2020). "Deep learning algorithms for automated detection of Crohn's disease ulcers by video capsule endoscopy". In: *Gastrointestinal endoscopy* 91.3, pp. 606–613.

Litjens, Geert et al. (2017). "A survey on deep learning in medical image analysis". In: *Medical Image Analysis* 42, pp. 60–88. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2017.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1361841517301135.

Malagelada, C et al. (2012). "Functional gut disorders or disordered gut function? Small bowel dysmotility evidenced by an original technique". In: *Neurogastroenterology & Motility* 24.3, 223–e105.

Mamonov, Alexander V. et al. (July 2014). "Automated Polyp Detection in Colon Capsule Endoscopy". In: *IEEE Transactions on Medical Imaging* 33.7, 1488–1502. URL: http://dx.doi.org/10.1109/TMI.2014.2314959.

McInnes, Leland et al. (2018). "UMAP: Uniform Manifold Approximation and Projection". In: *Journal of Open Source Software* 3.29, p. 861. DOI: 10.21105/joss.00861. URL: https://doi.org/10.21105/joss.00861.

Ribeiro, Tiago et al. (2022). "Artificial intelligence and colon capsule endoscopy: Automatic detection of ulcers and erosions using a convolutional neural network". In: *Journal of Gastroenterology and Hepatology* 37.12, pp. 2282–2288.

Saito, Hiroaki et al. (2020). "Automatic detection and classification of protruding lesions in wireless capsule endoscopy images based on a deep convolutional neural network". In: *Gastrointestinal endoscopy* 92.1, pp. 144–151.

Saraiva, Miguel Mascarenhas et al. (2021). "Artificial intelligence and colon capsule endoscopy: automatic detection of blood in colon capsule endoscopy using a convolutional neural network". In: *Endoscopy International Open* 9.08, E1264–E1268.

Seguí, Santi et al. (2016). "Generic feature learning for wireless capsule endoscopy analysis". In: *Computers in biology and medicine* 79, pp. 163–172.

Settles, Burr (2009a). "Active learning literature survey". In.

— (2009b). "Active learning literature survey". In.

— (July 2010). "Active Learning Literature Survey". In: *University of Wisconsin, Madison* 52.

Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946. arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

Vasilakakis, Michael et al. (2019). "Follow-up on: optimizing lesion detection in small bowel capsule endoscopy and beyond: from present problems to future solutions". In: *Expert review of gastroenterology & hepatology* 13.2, pp. 129–141.