Politecnico di Torino

Politecnico di Torino

Master in Physics of Complex Systems

A.a. 2025/2026

Graduation Session March 2026

Multiple Instance Learning for Differential Diagnosis of Ovarian Tumors in Ultrasound Imaging

Supervisors:

Andrea Pagnani

Yernur Kushaliyev

Candidate:

Silvia Collicelli

Abstract

Ovarian cancer affects women worldwide, and early and accurate detection is essential for appropriate treatment planning. The gold standard for diagnosis is histological biopsy, a costly and invasive procedure that could potentially be avoided in many cases through reliable imaging-based assessment. Ultrasound imaging represents a widely available and cost-effective alternative; however, its interpretation is inherently subjective and depends on the clinician's expertise.
In recent years, deep learning approaches have been applied to ultrasound images for automatic differential diagnosis, often achieving performance comparable to expert clinicians. Nevertheless there are some important limitations related to the application of standard supervised machine learning in this setting. In clinical practice different images and/or videos are collected for each patient, and then a unique label is assigned at the clinical case level. In machine learning terms, this means that labels are not available at the item-level, but only at the clinical case-level, leading to a weakly supervised learning problem that can be naturally addressed by the Multiple Instance Learning (MIL) paradigm.
The objective of this thesis is to investigate whether MIL-based models are better suited than standard supervised approaches for case-level diagnosis of ovarian tumors from ultrasound data. A baseline model based on standard supervised learning was first implemented. Subsequently, four MIL approaches with different assumptions and structural complexity were evaluated: Attention-based MIL (ABMIL), Dual-Stream MIL (DSMIL), Transformer-based MIL (TransMIL), and a Graph Neural Network (GNN)-based MIL model. To the best of our knowledge, this represents the first comparative study of these MIL architectures on an ultrasound dataset for ovarian tumor diagnosis.
Several metrics were computed, but F1-score was used to compare models and select the best one, as it is the most suitable for imbalanced datasets. All MIL models outperformed the supervised baseline, proving the suitability of the MIL framework for weakly labeled ultrasound data. Among the evaluated approaches, DSMIL achieved the highest performance. These results provide evidence that MIL constitutes a more appropriate learning strategy than standard supervised methods when only case-level annotations are available.

Table of Contents

Introduction

Ovarian cancer is one of the most common malignancies affecting women worldwide. A timely and accurate diagnosis is essential for appropriate treatment planning and for preventing disease progression. Ultrasound imaging represents the most widely used diagnostic modality for the assessment of ovarian tumors, as it is non-invasive, relatively inexpensive, and associated with minimal risk and discomfort for the patient [1]. However, ultrasound imaging is highly operator-dependent: image quality and diagnostic interpretation can vary depending on the clinician performing the acquisition, the equipment used, and patient-specific factors that may facilitate or hinder the examination. For these reasons, computational approaches capable of providing a more objective and reproducible assessment based on ultrasound data could significantly support clinical decision-making.

With the rapid development of deep learning techniques, increasing attention has been devoted to their application in the automatic analysis of ovarian tumors [2]. Numerous studies have proposed deep learning models for the differential diagnosis of ovarian masses, typically aiming to distinguish between benign and malignant lesions. Most of these approaches rely on fully supervised learning, which assumes that detailed annotations are available for each individual image or video frame in the dataset. In clinical practice, however, this assumption is often unrealistic. A more common scenario involves multiple ultrasound images or video sequences acquired for a single clinical case, with only one overall diagnosis or risk classification provided at the patient level.

In this context, Multiple Instance Learning (MIL) offers a suitable weakly supervised framework. MIL assumes that data are organized into groups of instances, called bags, and that labels are available only at the bag level rather than for each individual instance. This paradigm is particularly appropriate for medical imaging scenarios where fine-grained annotations are difficult, time-consuming, or impractical to obtain.

In this project, we investigate and compare several deep learning models based on the MIL paradigm for the binary classification of ovarian tumors into benign or malignant from ultrasound data. As a reference baseline, we first implement a conventional supervised model that performs binary classification at the instance

level by propagating the bag label to all instances within the bag. Subsequently, four established MIL-based models are implemented and evaluated on a dataset of ultrasound images and videos of ovarian tumors. The models considered in this study are:

1. **Attention-Based MIL (ABMIL)**: assigns an attention weight to each instance in order to identify those that contribute the most to the bag-level prediction;

2. **Dual-Stream MIL (DSMIL)**: identifies the most relevant instance, referred to as critical, and models relationships among each other instance and that critical one;

3. **Transformer-based MIL (TransMIL)**: employs self-attention mechanisms to capture pairwise interactions among instances within the bag;

4. **Graph Neural Network (GNN)-based MIL**: represents bags as graphs in which instances correspond to nodes, enabling the modeling of structural relationships through edges.

All these models aggregate instance-level information to obtain a single bag representation, which is then used to perform the final binary classification.

This thesis is organized as follows. **Chapter 1** introduces the clinical background, describing the main characteristics of ovarian tumors and motivating the application of machine learning techniques in this domain. **Chapter 2** provides the theoretical foundations necessary for this work: the first part reviews deep learning models and their typical workflows, while the second part focuses on the Multiple Instance Learning paradigm, presenting its formal definitions, assumptions, and the theoretical description of the models considered in this project. **Chapter 3** details the experimental methodology, including dataset preparation, evaluation metrics, and the implementation details of each model, together with their architecture-specific hyperparameters. Finally, **Chapter 4** presents and discusses the experimental results, providing a comprehensive comparison between the different MIL approaches and the baseline model.

The results obtained in this work demonstrate that MIL-based approaches can outperform the baseline supervised model in the task of binary classification for ovarian tumor differential diagnosis using ultrasound data, particularly in scenarios where only case-level labels are available.

Chapter 1

Medical motivation: overview of ovarian tumors

In this introductory chapter, first it is provided an overview of ovarian tumors, describing their main characteristics and the different types, stressing the high diffusivity, the often late diagnosis and the associated risks. Then some examples of ultrasound images of ovarian cysts are presented and the standard diagnostic rules used to predict the type of tumor based on the image features are described, highlighting the inherit subjectivity and variability related to these rules, introducing then how the usage of machine learning models could help overcome these limitations. Finally, the role of SynDiag is introduced, the company where this thesis project was developed, which aims at pursuing machine learning based alternatives to support and improve ovarian tumor diagnosis.

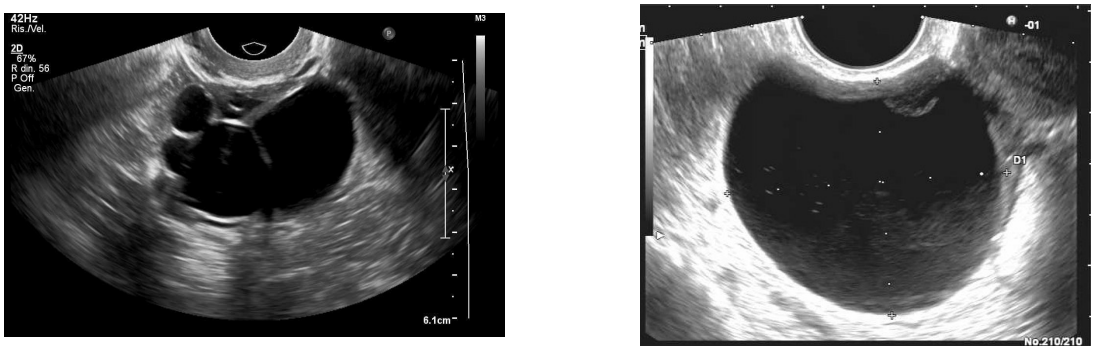
1.1 General aspects of ovarian tumors

According to data published by the World Cancer Research Fund in [3], in 2022 ovarian cancer was the eighth most common cancer among women worldwide, with more than 300.000 new cases reported. This fact highlights the importance of an early detection and an accurate classification. Unfortunately, due to its mild symptoms and to the absence of an effective population-wide screening program, diagnosis frequently occurs at advanced stages, for which the survival rate is much lower than for detection at earlier stages [4].

Analyzing more deeply ovarian tumors, they can be of many different types and can be broadly grouped into three categories: benign, malignant and borderline. Malignant tumors correspond to what is commonly referred to as ovarian cancer and are associated with the most severe clinical outcomes, while borderline cases represent an intermediate group with features between benign and malignant lesions.

Although all tumor types require appropriate clinical management, prognosis and treatment strategies differ substantially depending on the tumor classification. Consequently, a correct and early diagnosis is crucial to ensure the most suitable therapeutic approach.

A further difficulty in ovarian tumor management is the ability to correctly identify both the presence and the type of tumor using non-invasive methods. Histological analysis obtained through biopsy remains the most reliable diagnostic tool; however, it is invasive, costly, and not always feasible as a first-line examination. For this reason, ultrasonography represents the most widely used imaging modality for the evaluation of adnexal masses [5], owing to its accessibility, low cost, and safety. To better understand how the data we are discussing presents itself, Figure 1.1 shows two examples of ultrasound images of ovarian tumors, where the black circular area correspond to the cyst.



(a) First example

(b) Second example

Figure 1.1: Examples of ultrasound images of ovarian tumours

To support clinicians in the interpretation of ultrasound images, a set of standardized criteria known as the International Ovarian Tumor Analysis (IOTA) rules were developed [6]. These rules aim to classify ovarian tumors based on specific morphological and structural features observed in ultrasound images. Some of these rules are summarized in Figure 1.2, where red labels correspond to features related to malignant tumors, while green labels correspond to features of benign tumors.

Although the IOTA rules represent an important step toward standardizing ultrasound-based diagnosis of ovarian tumors, this process still involves a significant degree of subjectivity. Variability may arise both during image acquisition, as clinicians operating the ultrasound probe may emphasize certain anatomical regions while overlooking others, and during image interpretation, since several diagnostic criteria rely on qualitative assessments, such as the evaluation of cyst irregularity. As a result, inter-observer variability remains a well-documented issue, potentially

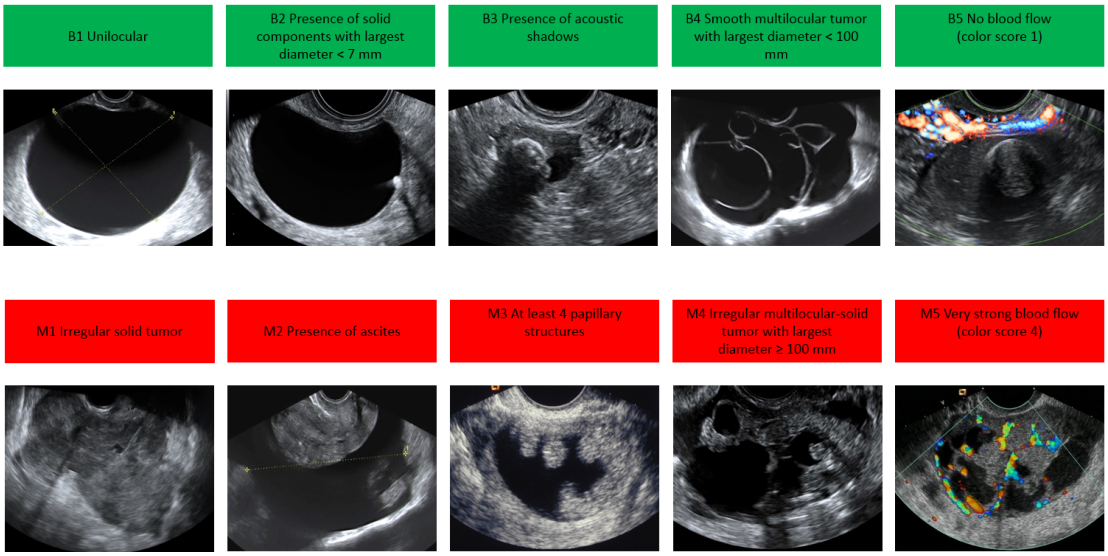


Figure 1.2: Summary of IOTA rules

leading to inconsistent diagnostic outcomes across operators and clinical centers.

1.2 Machine learning for ovarian tumors

In this medical setting, machine learning techniques offer a promising approach to reduce subjectivity and improve diagnostic consistency by learning predictive patterns directly from imaging data. Recent studies have shown that data-driven models can achieve performance comparable to expert-based assessment in ovarian tumor classification tasks [2]. These results motivate the investigation of automated methods as decision support tools in clinical practice and form the basis of the approach explored in this thesis.

Despite their potential, the application of machine learning methods in clinical workflows presents some practical challenges. In particular, most supervised learning techniques require large datasets with detailed annotations, including image-level labels or explicit delineation of regions of interest. Such annotation procedures are rarely compatible with routine clinical practice, as they are time-consuming and require additional effort from trained clinicians. In standard diagnostic workflows, labels are typically assigned at the patient or examination level, regardless of the number of images or videos acquired, and without explicit localization of diagnostically relevant regions.

This mismatch between the structure of clinical data and the requirements of conventional supervised learning approaches highlights the need for alternative

machine learning paradigms. Methods capable of learning from weakly labelled data, where only coarse-grained annotations are available, are particularly well suited to this setting [7]. By leveraging patient-level labels while implicitly identifying informative patterns at the image level, such approaches enable the development of scalable and clinically compatible decision support systems for ovarian tumor diagnosis and corresponding gold standard provided by histological biopsy.

1.3 Context of the thesis: SynDiag

This thesis project was developed in collaboration with SynDiag, an enterprise focused on the development of artificial intelligence–based decision support tools for gynaecological imaging. The company aims to assist clinicians in the analysis of ultrasound examinations of ovarian lesions, with the goal of supporting tumor characterization and clinical decision-making during routine diagnostic workflows.

SynDiag's research activities are primarily centered on two complementary tasks. The first involves the segmentation of irregular anatomical structures within adnexal masses, with the objective of identifying potentially pathological regions. The second focuses on differential diagnosis, which aims to predict the type of ovarian tumor based on imaging features extracted from ultrasound images and video sequences. To support these activities, SynDiag collaborates with clinicians from multiple hospitals in Italy and abroad and has access to a heterogeneous dataset of retrospectively collected ultrasound images and videos, each associated with expert-level clinical diagnoses.

This thesis focuses on the differential diagnosis task. In particular, it investigates the application of weakly supervised machine learning approaches to address the annotations granularity limitations discussed in the previous section. Since clinical labels are typically available only at the patient level, the adoption of learning paradigms capable of exploiting such coarse annotations is essential. For this reason, the methods explored in this work build upon Multiple Instance Learning (MIL) frameworks, which are specifically designed to learn from collections of instances associated with a single global label. A detailed overview of the machine learning background and multiple instance learning techniques relevant to this study is provided in next chapter.

Chapter 2

Foundations of Machine Learning and Multiple Instance Learning

In this chapter, the background knowledge necessary for understanding this thesis project is introduced. First, general concepts of machine learning are recalled, focusing on the typical workflow of a learning-based approach and on key aspects such as model training and generalization. Subsequently, the discussion focuses on deep learning methods, describing neural networks and the role of convolutional layers in extracting meaningful feature representations from images.

The second part of the chapter focuses on the Multiple Instance Learning (MIL) paradigm, presenting its formal definition, underlying assumptions, and main methodological categories. Finally, an overview of deep learning approaches applied to ultrasound images of ovarian tumors is provided as a reference for baseline models, followed by a detailed description of the original implementations of the MIL-based models considered in this work.

2.1 Machine learning background

In this section, a brief overview of the general machine learning workflow is provided in order to clearly define the main procedural steps adopted throughout this project. Subsequently, a more detailed explanation of neural networks is given, with particular attention to their application to image data through convolutional layers and to the extraction of feature representations, commonly referred to as embeddings.

2.1.1 Machine learning basics

The objective of a machine learning model is to first automatically detect and learn specific patterns in data and second to use them to make inference and decisions about new data. These patterns are learned by optimizing the parameters of a function that should be able to model the input data. This purpose is obtained through different steps, following a specific workflow that is common for every type of ML problem and that is summarized in Figure 2.1.

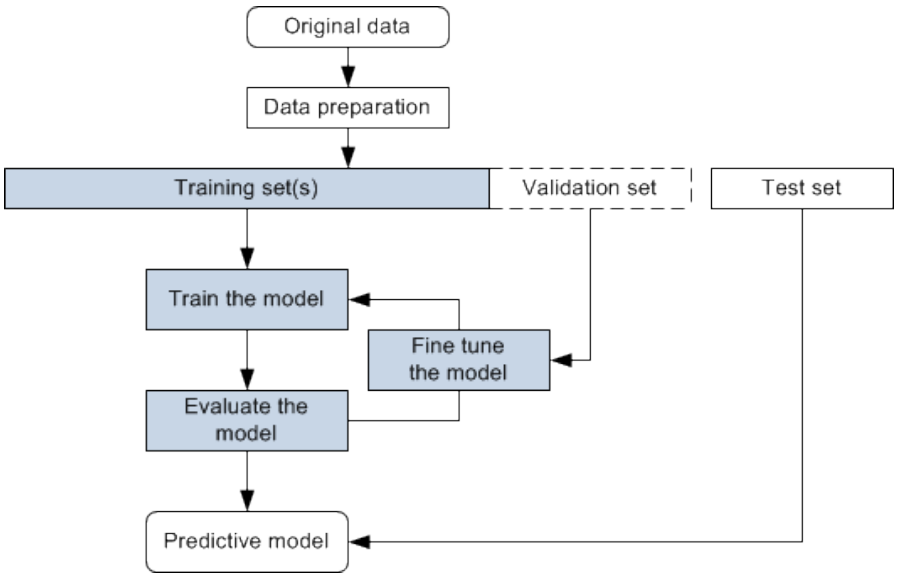


Figure 2.1: Schematic Machine Learning Workflow

General workflow

The first step of the workflow is *data acquisition*. In many practical scenarios, including the present thesis, this step is performed by actors outside the machine learning pipeline. In the considered case, ultrasound images and videos are acquired by clinicians during patient examinations, while collaborators at SynDiag collect and organize the data into a centralized database. The second step is *data preprocessing*. This phase aims at prepare data efficiently and in a suitable format for the learning algorithm. This preparation could include operations such as cropping, resizing, normalization or standardization, depending on the nature of the data. After preprocessing, the dataset is typically split into *training*, *validation*, and *test* sets. The training and validation sets are used to train and tune the model under different hyperparameter configurations. Once the optimal configuration is selected based on validation performance, the final model is evaluated on the test set, which provides

a final estimate of its generalization performance. Algorithm 1 summarizes the main steps of the learning process.

Algorithm 1 Typical Machine Learning Workflow

Require: Dataset $\mathcal{D}$, learning algorithm $\mathcal{A}$, hyperparameters Θ
Ensure: Trained model M^* and performance metrics
1) Data acquisition
2) Preprocessing of dataset $\mathcal{D}$
3) Split $\mathcal{D}$ into training, validation, and test sets
for all $\theta \in \Theta$ **do**
    4) Train model $M \leftarrow \mathcal{A}(\mathcal{D}_{train}, \theta)$
    5) Evaluate M on $\mathcal{D}_{val}$
end for
6) Select best model M^* based on validation performance
7) Evaluate M^* on $\mathcal{D}_{test}$
return M^* and evaluation metrics

Training and validation

The training phase consists in iteratively updating the model parameters in order to minimize a loss function, which quantifies the discrepancy between the model predictions and the corresponding ground-truth labels. This optimization process is typically performed using gradient-based methods, which adjust the parameters in the direction that reduces the loss. Validation is performed alongside training to monitor the model's performance on unseen data and to guide decisions regarding the hyperparameter selection. The alternation between training and validation steps allows one to assess whether improvements in training performance translate into better generalization.

Generalization and overfitting

A fundamental requirement of a machine learning model is the ability to generalize to unseen data. A model that is excessively complex may memorize the training data instead of learning the underlying patterns, while an overly simple model may fail to capture relevant structures in the data. An index of the generalization level is usually obtained by comparing the training loss with the validation loss. While the training loss typically decreases monotonically during optimization, the validation loss usually reaches a minimum and may start increasing afterwards. This behavior indicates the beginning of the overfitting phase, where the model begins to adapt too closely to the training data, memorizing them and losing its ability to generalize.

In classical machine learning models, the inputs of the learning model are feature vectors that summarizes relevant characteristics of the input data. An evolution of these models are the deep learning ones, in which feature extraction and prediction are jointly learned from data. The next section focuses on neural networks, which are the basis of deep learning models, with particular attention to convolutional architectures for image analysis and to the construction of learned feature representations used as inputs for Multiple Instance Learning models.

2.1.2 Basics on Neural Networks

In classical machine learning approaches, features are manually extracted from the raw data and subsequently provided as input to a learning algorithm, which is then trained to produce a prediction based on these fixed representations. In contrast, in deep learning models the feature extraction is performed jointly with the prediction, learning hierarchical representations directly from data through a class of models known as **neural networks**.

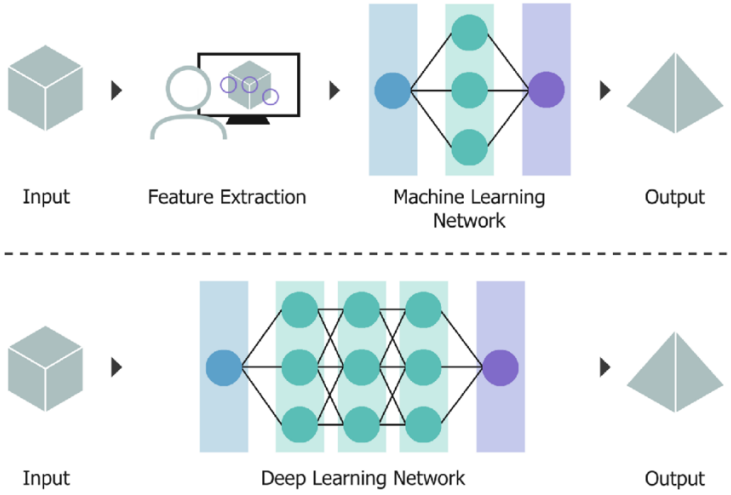


Figure 2.2: Conceptual difference between classical Machine Learning and Deep Learning pipelines

A neural network (NN) is a model composed of interconnected nodes called neurons, organized in layers. Each neuron performs a linear combination of its inputs followed by a non-linear transformation, known as the *activation function*. Formally, the output of a neuron can be expressed as:

$$y = f(\boldsymbol{w}^T \boldsymbol{x} + b) \tag{2.1}$$

where $\boldsymbol{x}$ denotes the input vector, $\boldsymbol{w}$ the node weights, b a bias term, and $f(\cdot)$ a nonlinear activation function. Nonlinearity is a fundamental component of

neural networks, as it enables the model to approximate complex, non-linear relationships in data. Common activation functions include the sigmoid function and the Rectified Linear Unit (ReLU), defined as $f(x) = \max(0, x)$. Neurons are arranged in multiple layers, starting from an input layer, passing to one or more hidden layers and finishing with the output layer, which produces the final data representation or prediction of the model. By stacking multiple layers, neural networks are able to learn increasingly abstract representations of the input data. The main characteristics of a neural network can be observed in Figure 2.3, where the three-layers example is a fully connected neural network, as every neuron receives inputs from all neurons of the precedent layer.

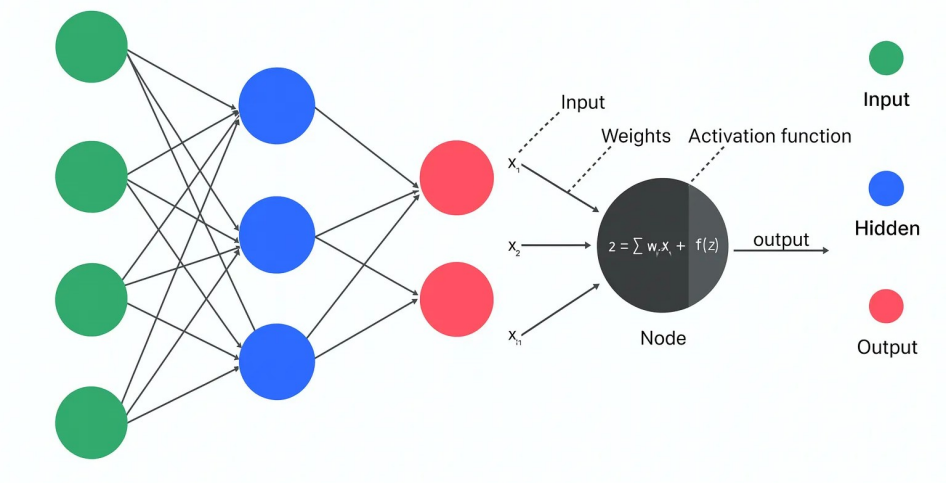


Figure 2.3: Neural Network architecture

Training a neural network consists in optimizing the parameters of all neurons in order to minimize a loss function that measures the discrepancy between the network output and the ground-truth target. This optimization is commonly performed using backpropagation, an efficient algorithm that computes gradients of the loss function with respect to all parameters by applying the chain rule of derivatives. Once the gradients are computed, the node parameters are properly updated. A key hyperparameter in this process is the **learning rate**, which controls the magnitude of parameter updates. A learning rate that is too large may lead to unstable training dynamics or divergence, while a learning rate that is too small can result in slow convergence and increased computational cost.

Although the first ideas related to neural networks were already developing in the mid-20th century [8], the spread of these models is more recent, because of the increased computational resources and the availability of large-scale datasets, which are strictly necessary with highly parametrized models as these kinds.

While neural networks are powerful function approximators capable of capturing

complex patterns, this expressive capacity often results in a large number of trainable parameters. When dealing with high-dimensional and structured data, such as images, this motivates the use of specialized neural network architectures, like convolutional neural networks, which reduce significantly the number of trainable parameters.

2.1.3 Convolutional Neural Networks and feature extraction

Convolutional neural networks (CNNs) are a specific class of neural network architectures that make use of convolutional filters (or kernels) to exploit the structure of the input data, while reducing the number of trainable parameters. They are particularly suitable for data with a locality property, such as time series, texts and images.

The main advantage of CNNs lies in their use of local connectivity and weight sharing. Each filter is applied only to a local region of the input and has a fixed set of weights that is shared across the entire input. By sliding over the input data, the filter produces an output feature map, significantly reducing the number of parameters compared to fully connected architectures. As a result, convolutional neural networks are able to recognize visual patterns independently of their spatial location. This property is particularly relevant for ultrasound imaging, where anatomical structures may appear at different positions across samples. In the case of images, convolutional filters are two-dimensional; an example of a 3×3 filter applied to an image is shown in Figure 2.4.

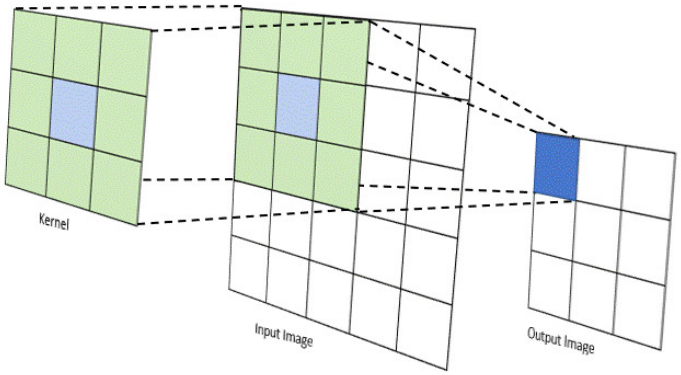


Figure 2.4: Example of two dimensional convolution

Each convolutional filter is designed to capture a specific type of local feature. For this reason, multiple filters are usually applied within the same convolutional layer, allowing the network to extract different features in parallel and produce

multiple output feature maps. A typical convolutional neural network is composed of several convolutional layers that progressively extract higher-level features from the input data, together with other types of layers. Pooling layers are often used to downsample and aggregate information, while normalization layers can improve training stability. Fully connected layers are usually placed at the end of the network to produce the final classification. An example of a CNN architecture is shown in Figure 2.5, where the model is used to classify the content of an image.

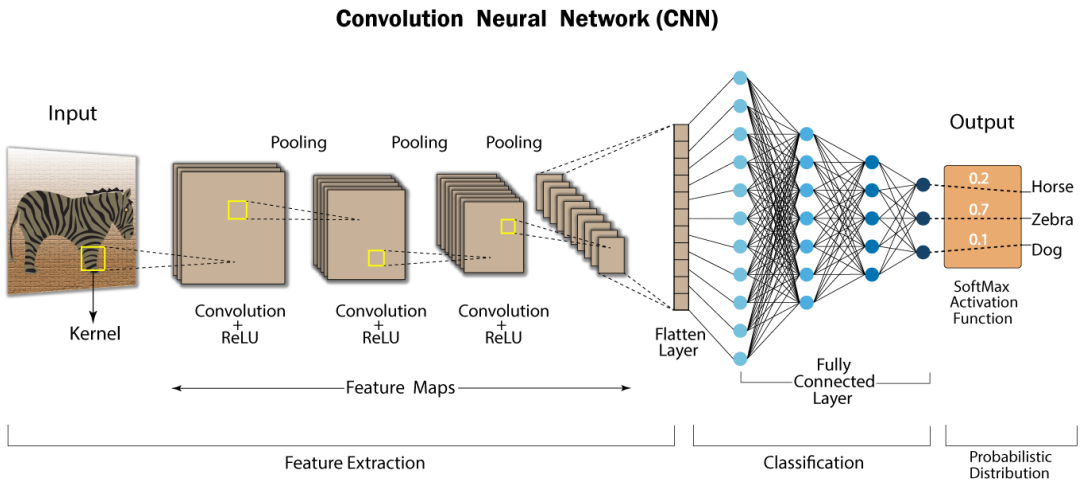


Figure 2.5: Example of the architecture of a convolutional neural network

Convolutional neural networks are therefore particularly effective for feature extraction from images. In this project, CNNs are extensively used to generate embeddings, defined as compact feature vectors representing each image, which are subsequently provided as input to the classification models. These embeddings constitute the instance-level representations used in the Multiple Instance Learning framework described in the next section.

2.2 Multiple Instance Learning (MIL)

In order to effectively exploit the characteristics of the considered ultrasound dataset, it is necessary to adopt a weakly supervised learning paradigm that takes into account the reduced availability of annotations. In standard supervised learning, the dataset consists of pairs of input instances and their corresponding labels, meaning that each individual sample is fully annotated. In contrast, weakly supervised learning refers to settings in which complete and precise annotations are not available, and the model must learn from partially labeled, imprecise, or

coarse-grained supervision.

This scenario is particularly common in medical applications: obtaining detailed annotations often requires expert knowledge and considerable time, making large-scale instance-level labeling impractical. As a consequence, many medical datasets are available only with patient-level labels, while fine-grained annotations at the level of individual images or regions are missing.

Multiple Instance Learning (MIL) addresses this type of problem by considering training data organized into sets of instances, called bags, for which only bag labels are provided, whereas individual instance labels remain unknown. In the following sections, the MIL paradigm is formally introduced, and its definitions, assumptions, and main methodological approaches are presented, as it constitutes the core framework of this thesis.

2.2.1 Problem formulation

In a Multiple Instance Learning problem instances are grouped into bags and only bag labels are available. Bags can contain different numbers of instances and single item labels are unknown. The goal of the model is to learn a mapping function that assigns labels to the corresponding test bags or instances. Figure 2.6 shows in a schematic way the difference between standard supervised learning, which is the Single Instance Single Label (SISL) case, and Multiple Instance Learning. The color of each instance represents its label, and as the example is a binary classification problem, it can either be positive or negative. In the SISL case each instance has its own label, whereas in MIL there is one single label per bag.

This learning paradigm was first applied to drug activity prediction [10], where the objective was to predict a molecular binding to a receptor. The molecular structure of the problem can have several conformations and if it contains at least one conformation that is low-energy then it can bind to the receptor and be considered as active. This problem configuration immediately reflects a MIL problem, as each structure conformation that the molecule assumes corresponds to an instance whose label is unknown; instead at the bag level the label is known, as one can observe whether the molecule is active or not. MIL paradigm can be applied also in many other contexts, like in the medical one. In this field, it is particularly used for pathology slides, also called Whole Slide Images (WSI), which are high-resolution slides of tissue samples from virtual microscope, often used in digital pathology. As they are highly detailed, a fine-grained annotation is often not available and a MIL-based approach could address this issue.

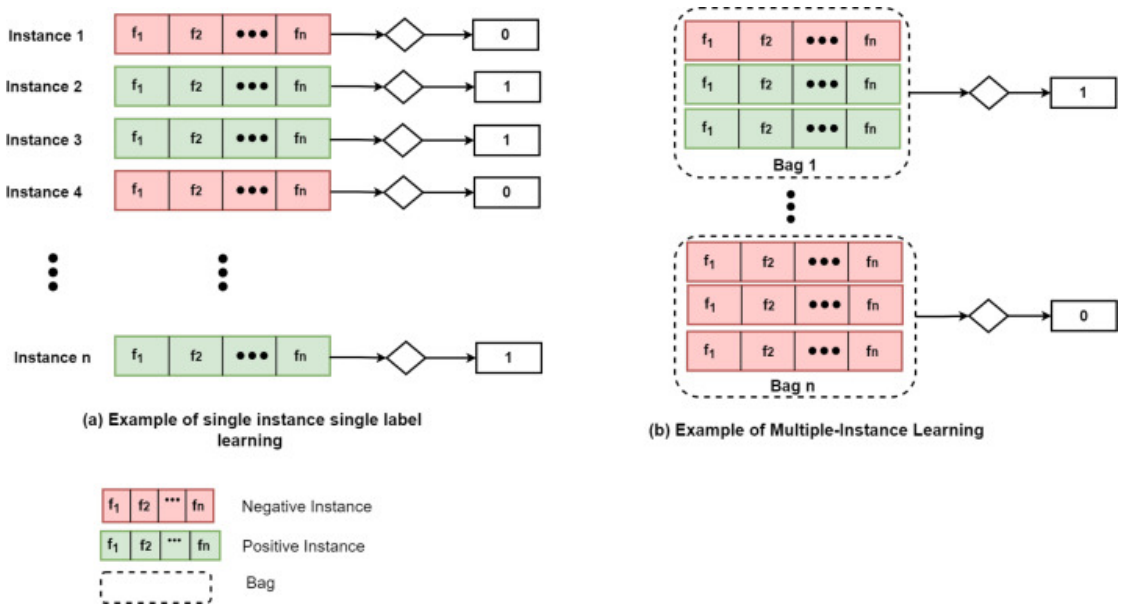


Figure 2.6: Difference between a SISL problem and a MIL problem from [9]

MIL assumptions for bag-level classification

Considering a general MIL problem of binary classification: the dataset consists of n bags and corresponding labels: $B = \{(\mathcal{B}_1, \mathcal{Y}_1), (\mathcal{B}_2, \mathcal{Y}_2), ..., (\mathcal{B}_n, \mathcal{Y}_n)\}$, where each bag is composed of a set of instances: $\mathcal{B}_i = \{x_{i,1}, x_{i,2}, ..., x_{i,n_i}\}$ with unknown instance labels $Y_i = \{y_{i,1}, y_{i,2}, ..., y_{i,n_i}\}$, each of them can be either equal to 1, indicating a positive instance or 0 as a negative one. The **Standard Assumption** states that a bag is labeled as positive if it contains at least one positive instance, otherwise, if it only contains negative instances, it is labeled as negative:

$$\mathcal{Y}_i = \begin{cases} 1, & \text{if } \exists y_{i,j} \in Y_i : y_{i,j} = 1 \\ 0, & \text{if } \forall y_{i,j} \in Y_i : y_{i,j} = 0 \end{cases} \tag{2.2}$$

The majority of MIL problems, like the already seen of drug activity prediction, are based on this assumption.

However, there exist also other more generic collective non-standard assumptions that for example can consider the case of needing more than one positive instance to classify a bag as positive. An even more general collective non-standard assumption involves the assignment of concepts to each instance. Given a set of concepts $\mathcal{C}$, where a subset of them belong to the positive class ($\mathcal{C}_+ \subset \mathcal{C}$), a bag is classified as positive if its instances contain at least a specific number of times a positive concept. A simple example that clarifies this kind of assumption is provided in [11]. The problem considered is classifying images with desert, sea or beach. Desert images

contain sand segments, sea images contain water segments, and beach images contain both sand and water segments. To classify an image as a beach, the model must verify the presence of both segments. In this case the standard assumption couldn't work, instead a collective non-standard one that includes concepts is the optimal way to deal with this type of problem.

Classification level

Classification in MIL can be performed at two different levels: at the bag or at the instance level. Even though bag classification is the most common case, one could be interested also in classifying single instances individually. As instance labels are not available during training, different methods and algorithms are suited for the two classification levels. Importantly, also the loss function is different and misclassifications could have different impact. In fact, when the objective is classifying bags, if a bag is positive and some positive instances are misclassified as negative, this doesn't lead to any change in the bag label, if there is still at least one positive instance, under the standard assumption. On the contrary, if this happens in the instance-level classification case, each misclassification affects the loss function. For this reason usually accuracy for instance-level classification is lower.

2.2.2 Types of MIL methods

There are different types of MIL methods which can be grouped in three categories: instance-space methods, bag-space methods and embedding-space methods, based on how they treat the information collected from the dataset, as described in MIL review [12].

- In **instance-space methods** learning happens at the instance level. First the model learns how to predict instance labels and then those labels are aggregated according to the multiple instance assumption to give a prediction for the whole bag.

- **Bag-space methods** consider the bag as a whole and train the model for bag-level classification through traditional supervised learning techniques, usually evaluating a distance or a dissimilarity measure in the bag-space.

- **Embedding-space methods** consider instance embeddings, the feature vector representation of the bag items. The model aggregates them into a unique bag embedding, that captures the bag-level information which is then classified through traditional supervised algorithms.

2.2.3 MIL procedure steps applied to this project

The selection of Multiple Instance Learning (MIL) models to be adapted and applied in this thesis was guided by the specific characteristics of the addressed problem, including the structure of the dataset and the clinical objective of the task. In this project, the task is formulated as a binary classification problem, where each clinical case must be classified as either benign (negative class) or malignant (positive class). The ultimate goal of the proposed models is to support the differential diagnosis at the patient level.

From a MIL perspective, this corresponds to predicting bag-level labels, where each bag represents a clinical case composed of multiple instances, such as ultrasound images or video frames. Instance-level ground-truth annotations are not available. Instead, only a global label is provided for each bag, assigned by expert clinicians. In particular, a clinical case is labelled as malignant if at least one instance within the bag contains a malignant tumor, whereas it is labelled as benign if no malignant tumor is observed in any instance. This setting directly follows the Standard MIL assumption, which states that a bag is positive if it contains at least one positive instance and negative otherwise.

Another important characteristic of the dataset is that it is composed of images and videos. As discussed in Section 2.1.3, Convolutional Neural Networks (CNNs) represent an effective solution for extracting discriminative visual features from such data. In the adopted MIL framework, each instance is first processed by a CNN-based feature extractor, producing a corresponding embedding. These instance embeddings are then aggregated through a MIL pooling mechanism based on embedding-space methods, which combine the information from all instances into a single bag-level representation. This global embedding is subsequently provided as input to a supervised classifier that outputs the final diagnostic prediction. A schematic overview of this pipeline is illustrated in Figure 2.7.

Since the application domain of this work is medical decision support, an additional desirable property of the considered models is interpretability. In particular, it is beneficial for clinical support that the model provides a measure of the contribution of each instance to the final prediction. Such information can help clinicians identify the most diagnostically relevant frames or images within a clinical case, thereby increasing trust in the automated system and facilitating its integration into clinical workflows.

In summary, the main requirements for a MIL model suitable for this project can be outlined as follows:

- the capability to perform bag-level binary classification;
- compatibility with the Standard MIL assumption;
- the adoption of a deep learning framework combining feature extraction and

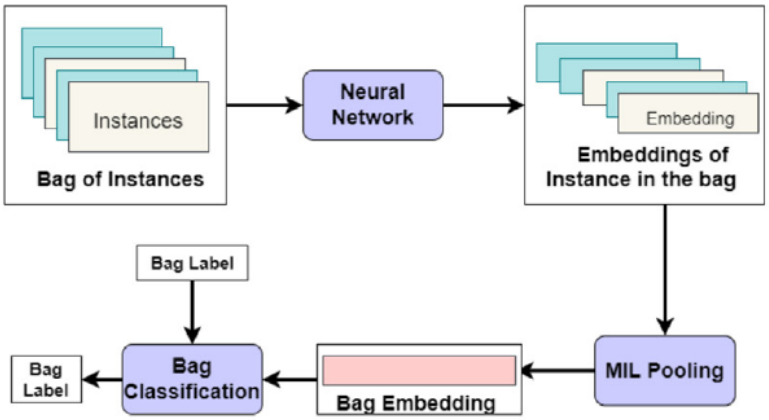


Figure 2.7: Scheme of the procedure steps of a MIL problem from [9]

aggregation;

- the use of embedding-space MIL pooling strategies;
- the provision, when possible, of interpretable outputs describing instance relevance.

Before presenting in detail the MIL architectures selected for this study, the next section briefly introduces the baseline model, which was chosen based on the current state of the art in deep learning approaches for the analysis of ultrasound datasets in ovarian tumor diagnosis.

2.3 Related work and baseline models

As ultrasound examination represents the most common, low–cost and non-invasive imaging modality for the diagnosis of ovarian tumors, several studies have already investigated the application of machine learning and deep learning methods for the automatic classification of ovarian lesions from ultrasound images. A recent meta-analysis [2] reviews multiple approaches differing in model architecture and experimental design but sharing the common objective of predicting a diagnosis.

To identify suitable baseline approaches for this project, we focused on studies employing deep convolutional neural networks. A key aspect emerging from the analysis of these works concerns the granularity of the available annotations. In clinical practice, the ground-truth diagnosis is established at the level of the clinical case (i.e., patient or lesion). However, deep learning models are commonly trained at the level of single ultrasound images. Therefore, particular attention was given

in understanding how each of these studies, that implement standard supervised models, addressed this mismatch.

Christiansen et al. [13] proposed an ensemble of pretrained convolutional neural networks (VGG16, MobileNet and ResNet50) to classify ovarian tumors. Each lesion was represented by a small set of manually selected representative ultrasound images. During training, the clinical case label was assigned to each image, and predictions obtained at image level were subsequently combined to produce a single patient-level probability of malignancy.

A similar supervision strategy was adopted by Gao et al. [14], who trained a DenseNet121 model on a very large dataset. In this case, each clinical case was associated with a larger number of images, and the case label was again propagated to all instances during training. Because of the higher number of items belonging to each patient not selected by experts, in this case noisier item labels were expected. The final patient-level prediction was obtained by aggregating the image-level probabilities through a weighted average, assigning greater importance to images considered more suspicious by the model.

Wang et al. [15] investigated both binary and three-class classification settings considering also borderline cases and using several standard convolutional architectures. In this case they used a smaller dataset with one or at most two images per clinical case and a label was provided for each image. Classification was performed at the item-level, which in the majority of cases corresponded also to the patient level, avoiding the mismatch problem. Jung et al. [16] extended the classification task to five tumor categories and explored the use of convolutional autoencoders to reduce visual and irrelevant annotations in the images, like the presence of calipers. Also in this case labels were available at the image level and there was no problem related to different label granularity.

Overall, these studies do not explicitly address the mismatch between the availability of annotations at clinical-case level and the instance-level supervision required by standard deep learning models. Instead, the first two cases rely on a label propagation strategy, whereby each image inherits the diagnosis of the corresponding patient. Image-level predictions are subsequently aggregated, typically through simple or weighted averaging, to obtain a final case-level prediction. In many datasets, this approach is facilitated by the prior selection of diagnostically relevant images by expert clinicians, which partially mitigates the noise introduced by label propagation. In the last two cases instead, this problem of granularity mismatch does not even apply, as single item labels are provided and image classification is performed.

In the next section, we introduce four Multiple Instance Learning based models specifically designed to handle this type of weak supervision. In particular, they provide a way to learn directly from case-level labels without requiring explicit instance-level annotations, making them particularly suitable for the clinical

scenario considered in this thesis.

2.4 MIL models

In recent years many deep learning models have been specifically developed to address MIL problems, providing aggregation methods compatible to the neural network environment and more complex than the basic average or maximum selection. The majority of these models have been developed independently and then tested on datasets of Whole Slide Images, which is the most common context where MIL models are applied. In [17] first developed deep MIL models as long as most recent ones are briefly reviewed. Starting from this analysis, four models have been selected and adapted to the dataset and the goal this project is pursuing.

This section provides a theoretical presentation of these four Multiple Instance Learning based models that have increasing complexity and different internal assumptions, starting from the simpler attention based MIL, which provides the attention layer as aggregation method, going on to the dual-stream based MIL as a middle way between ABMIL and next architecture, TransMIL, which uses the self-attention mechanism of transformers. Last model is based on Graph Neural Networks (GNNs) and presents consistently different architecture and assumptions with respect to the previous ones. In next chapter instead, it will be explained better how the original implementation of these models were adapted to our specific setting.

2.4.1 Attention Based MIL (ABMIL)

Ilse et al. in [18] provided what is now regarded as a sort of baseline for deep MIL problems. They presented an aggregation method, the attention mechanism, particularly suitable for deep learning models that contributes in increasing the model flexibility and interpretability.

Attention Based MIL pooling consists in a trainable weighted average of the instance embeddings, where the weights are given by a two-layered neural network, the attention mechanism. More in details, let $H = \{\mathbf{h}_1, ..., \mathbf{h}_k\}$ be a bag containing k instance embeddings, given as outputs by the feature extractor, each of them a vector $\mathbf{h}_i \in \mathbb{R}^M$. The aggregation to a single bag embedding is obtained through:

$$\mathbf{z} = \sum_{i=1}^{k} \alpha_i \mathbf{h}_i \qquad \text{with } \alpha_i > 0 \,\forall\, i, \quad \sum_{i=1}^{k} \alpha_i = 1 \tag{2.3}$$

The **attention weights** are obtained through the expression:

$$\alpha_i = \frac{\exp\{\mathbf{w}^T \tanh(\mathbf{V}\mathbf{h}_i^T)\}}{\sum_{j=1}^{k} \exp\{\mathbf{w}^T \tanh(\mathbf{V}\mathbf{h}_j^T)\}} \tag{2.4}$$

where $\mathbf{w} \in \mathbb{R}^{L\times 1}$ and $\mathbf{V} \in \mathbb{R}^{L\times M}$ are trainable parameters, L is the attention dimension and is an hyperparameter of the model and the hyperbolic tangent is used as activation function. They also proposed a variation to the attention pooling, the gated attention pooling, by adding another layer of non-linearity in order to improve gradient flow in the network. In this case the attention weights are computed through the formula:

$$\alpha_i = \frac{\exp\{\mathbf{w}^T(\tanh(\mathbf{V}\mathbf{h}_i^T) \odot \text{sigm}(\mathbf{U}\mathbf{h}_i^T))\}}{\sum_{j=1}^{k} \exp\{\mathbf{w}^T(\tanh(\mathbf{V}\mathbf{h}_j^T) \odot \text{sigm}(\mathbf{U}\mathbf{h}_j^T))\}} \tag{2.5}$$

where $\mathbf{U} \in \mathbb{R}^{L\times M}$ are other trainable parameters, sigm() is the sigmoid function and $\odot$ is the element-wise product.

Interpretability The attention pooling is a permutation-invariant transformation that assumes that instances in the same bag are independent. Through the trainable weights, also called attention scores, this aggregation method is able to provide an index of interpretability, as long as the actual label prediction. In fact, in each bag, instances with highest attention scores are the ones that give higher contribution to the bag prediction, thus they are the ones that should be the most relevant for the patient diagnosis. For example, in the case of a positive bag instances with highest weights are expected to be the positive ones that trigger the bag label to positive. This interpretation measure makes this model very appealing in the medical context, where it is very important to give clinicians a tool for interpretation together with the prediction. In the case of video frames for example, highest score instances could be interpreted as the interval of interest (IOI).

Flexibility Another important aspect of this approach is that attention pooling is differentiable and obtained through neural network layers. If also feature extraction and classification are obtained through $\mathcal{NN}$ modules, this implies that the whole procedure is differentiable and could be trained end-to-end by backpropagation with a unique neural network that combines the three tasks and make the whole approach very flexible. Figure 2.8 summarizes the procedure of the ABMIL approach, where instance embeddings are first obtained through a neural network that works as feature extractor; then attention pooling is performed, where attention weights are updated and multiplied by the corresponding instances to produce the bag embedding. Last a classifier is applied to obtain the final bag label.

2.4.2 Dual-stream MIL (DSMIL)

The second considered model was developed in paper [19] and consists in a particular MIL aggregator composed of a two-stream architecture that jointly learns both an

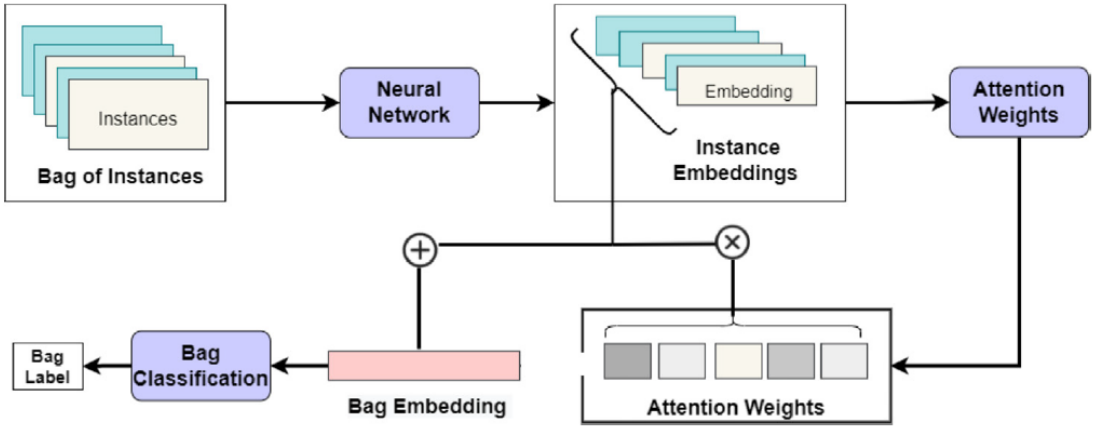


Figure 2.8: Schematic explanation of how ABMIL model works from [9]

instance and a bag classifier based on a distance measurement. The first stream apply a max-pooling aggregation to the instance scores, to identify the one with highest score, called the **critical instance**. The second stream attributes an attention score to each instance by evaluating the pairwise distance between every instance and the critical one and perform then a weighted sum. This provides a unique bag embedding, that is then given to a classifier and summed to the prediction of the critical instance to obtain the final model prediction.

DSMIL aggregator

More in details, the first stream applies an instance classifier to the instance embeddings and aggregate them through max-pooling. Let B be the bag we are considering, $H = \{\mathbf{h}_1, ..., \mathbf{h}_k\}$ its instance embeddings and $\mathbf{W}_0$ trainable parameters that produce a classification score. The final score of the first stream is given by:

$$c_m(B) = \max\{\mathbf{W}_0\mathbf{h}_1, ..., \mathbf{W}_0\mathbf{h}_k\} \tag{2.6}$$

where m is the index of the instance with highest score, the critical instance.

The second stream takes every instance embedding and transforms them into two vectors, $\mathbf{q} \in \mathbb{R}^M$ and $\mathbf{v} \in \mathbb{R}^M$ (referred to as query and information) through the trainable matrix weights $\mathbf{W}_q$ and $\mathbf{W}_v$:

$$\mathbf{q}_i = \mathbf{W}_q\mathbf{h}_i \qquad \mathbf{v}_i = \mathbf{W}_v\mathbf{h}_i \qquad \forall\, i = 1, ..., k \tag{2.7}$$

Distances are then evaluated between each instance query vector and the one from the critical instance, using the following distance measure:

$$U(\mathbf{h}_i, \mathbf{h}_m) = \frac{\exp(\langle \mathbf{q}_i, \mathbf{q}_m \rangle)}{\sum_{j=1}^{k} \exp(\langle \mathbf{q}_j, \mathbf{q}_m \rangle)} \tag{2.8}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product between two vectors. The bag embedding $\mathbf{b}$ is obtained through the sum of the information vector of each instance weighted by its distance to the critical instance:

$$\mathbf{b} = \sum_{i=1}^{k} U(\mathbf{h}_i, \mathbf{h}_m)\mathbf{v}_i \tag{2.9}$$

The distance U has actually the meaning of a similarity measure as the more the instance is similar to the critical one and the higher its value becomes. This measure can actually correspond to an attention weight, as it directly evaluate the amount of contribution that the instance provides to the bag embedding. The bag score coming from the second stream is:

$$c_b(B) = \mathbf{W}_b\mathbf{b} \tag{2.10}$$

where $\mathbf{W}_b$ is another trainable matrix of weights. The final bag score is given by the average of the scores coming from the two streams:

$$c(B) = \frac{1}{2}\Big(c_m(B) + c_b(B)\Big) \tag{2.11}$$

Figure 2.9 shows a schematic representation of MIL aggregator of DSMIL model.

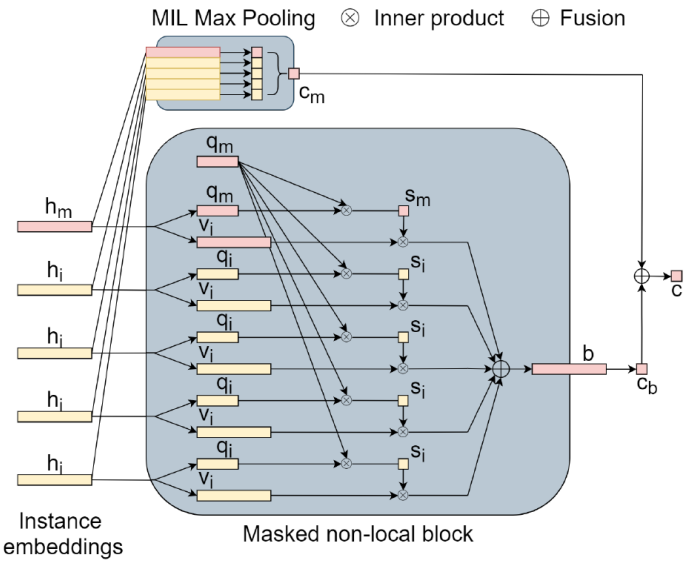


Figure 2.9: MIL aggregator of DSMIL from [19]

An important aspect that marks a difference from the ABMIL is that DSMIL model does not consider instances in the same bag to be independent; instead

it measures correlations of each instance with respect to the most relevant one. Furthermore, the additional layer of $\mathbf{v}_i$ allows to select intra-instance features, meaning the most discriminative features in an instance; while the distance measure applies an inter-instance selection, evaluating which are the most informative instances in the bag for the bag prediction.

2.4.3 Transformer-based MIL (TransMIL)

The third considered model was proposed by Shao et al. in [20] and consists of a Transformer-based MIL architecture, referred to as TransMIL. In many medical imaging scenarios, instances can be spatially and semantically correlated. In ultrasound imaging, for example, images belonging to the same clinical case usually show the same lesion or anatomical region, possibly from different viewpoints. TransMIL was originally designed with the aim of capturing both morphological information at instance level and spatial correlations across instances.

Differently from ABMIL and DSMIL, TransMIL employs a self-attention mechanism that explicitly models pairwise interactions between instances in the same bag, as schematically illustrated in Figure 2.10. This allows the aggregation process to be driven not only by the relevance of individual instances, but also by their mutual relationships.

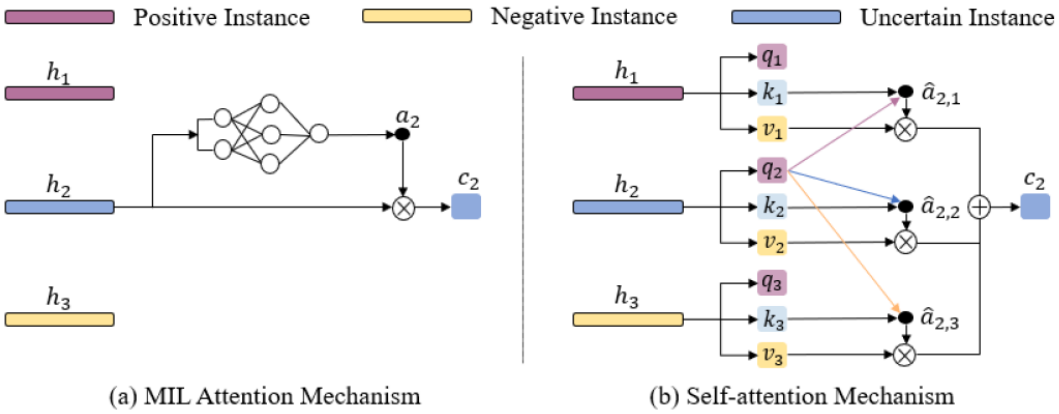


Figure 2.10: Difference between MIL attention mechanism and self-attention used in TransMIL [20]

TransMIL follows a two-stage MIL pipeline. First, instance-level features are extracted using a CNN backbone, similarly to the other considered architectures. In the original implementation a ResNet model was adopted, although in principle different feature extractors can be employed, provided that they produce a set of fixed-length instance embeddings.

The Transformer-based aggregation is performed through the so-called TPT module, composed of two Multi-Head Self-Attention layers separated by a Pyramid Position Encoding Generator (PPEG). The overall architecture is shown in Figure 2.11. The main processing steps can be summarized as follows:

1. **Sequence squaring:** Instance embeddings within each bag are padded and rearranged to form a squared 2D grid, enabling the application of spatial positional encoding.

2. **First self-attention layer:** Multi-head self-attention models global correlations among instances according to

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V.$$

3. **Pyramid Position Encoding Generator (PPEG):** Tokens are reshaped into a spatial layout and processed using convolutions with different kernel sizes. This step injects conditional spatial information and enriches each embedding with local contextual features.

4. **Second self-attention layer:** Further refines instance interactions after spatial information has been incorporated.

5. **Bag-level prediction:** The class token representation is fed to a Multi-Layer Perceptron (MLP) to obtain the final bag label.

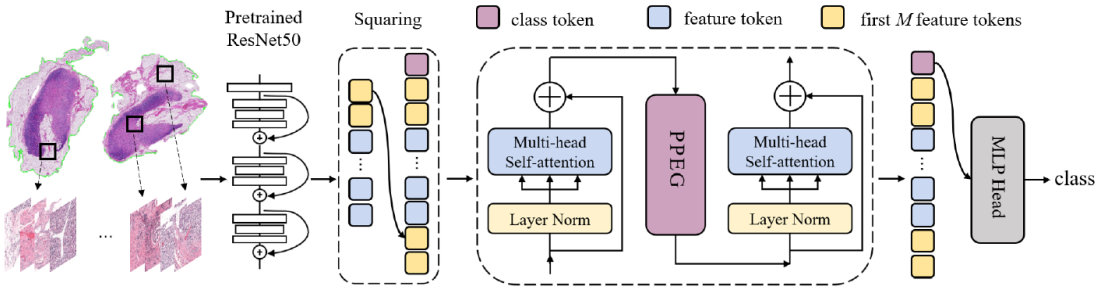


Figure 2.11: Overview of TransMIL model from [20]

TransMIL relies on assumptions that distinguish it from both ABMIL and DSMIL. In particular, it explicitly assumes that instances within a bag are not independent, but may exhibit spatial or semantic correlations. Modeling such dependencies can improve the representation of the overall clinical case. Compared

to standard attention-based MIL methods, TransMIL therefore shifts the focus from instance weighting to instance interaction modeling.

This characteristic is especially relevant when dealing with video data or multi-view image acquisitions, where frames are temporally or spatially related and diagnostic information may emerge from their combined interpretation rather than from isolated observations. In this sense, TransMIL can be considered a structured and correlation-aware extension of classical MIL aggregation strategies.

2.4.4 Graph neural networks for MIL

The last model considered in this project is based on a different neural architecture: the graph neural network (GNN) . Unlike standard feed-forward or convolutional networks, a GNN operates on graph-structured data. At each layer, node representations are updated by aggregating information from their neighbors, allowing the model to explicitly exploit relationships between entities. GNNs can be used for node-level tasks (e.g., node classification), edge-level tasks (e.g., link prediction), and graph-level tasks (e.g., graph classification). Their key strength lies in their ability to learn expressive representations that incorporate both node features and the relational structure encoded by the edges.

In a multiple instance learning (MIL) setting, each bag consists in a set of (correlated) instances. By converting each bag into a graph, instances become nodes and edges model pairwise relationships. This approach generalizes classical MIL pooling mechanisms by explicitly modeling dependencies among instances and use them to perform the aggregation. Before describing how GNNs are adapted to MIL, we briefly review their general formulation.

Graph Neural Networks (GNNs)

A graph is defined as $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. In the graph neural network setting, each node $i \in V$ is associated with a feature vector $\mathbf{h}_i \in \mathbb{R}^d$. Collecting all node features yields the matrix of all nodes feature vectors $\mathbf{X} \in \mathbb{R}^{k \times d}$, where $k = |V|$. The core mechanism of most modern GNNs is **message passing**, also known as neighborhood aggregation. At layer k, the representation of node i is denoted as $\mathbf{h}_i^{(k)}$ and updated according to:

$$\mathbf{h}_i^{(k)} = \gamma^{(k)}\left(\mathbf{h}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)}\left(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}, \mathbf{e}_{ji}\right)\right) \tag{2.12}$$

where:

- $\mathcal{N}(i)$ denotes the neighborhood of node i,
- $\bigoplus$ is a permutation-invariant aggregation operator (e.g., sum, mean, or max),

- $\phi^{(k)}$ is a function that computes messages from neighbors,
- $\gamma^{(k)}$ is a function that updates the node state.

The permutation invariance of $\bigoplus$ ensures that the model is invariant to node ordering, which is crucial since graphs are unordered structures. By stacking multiple layers that compute this operation, nodes gather information from progressively larger neighborhoods, learning both local and global graph structure. If the goal of the model is a graph-level prediction, then graph convolutional layers like 2.12 are combined with pooling and readout layers, that respectively coarsen the graph into a smaller substructure and aggregate node representation into a single graph embedding. Figure 2.12 shows in (a) the case of a GNN composed only of graph convolutional layers (referred to as Gconv) that has the goal of extracting the graph hidden representation; instead in (b) is the case where the goal of the GNN is graph classification, so the network combines Gconv layers, with readout and pooling layers and a final Multi-Layer Perceptron (MLP) that assigns class probabilities.

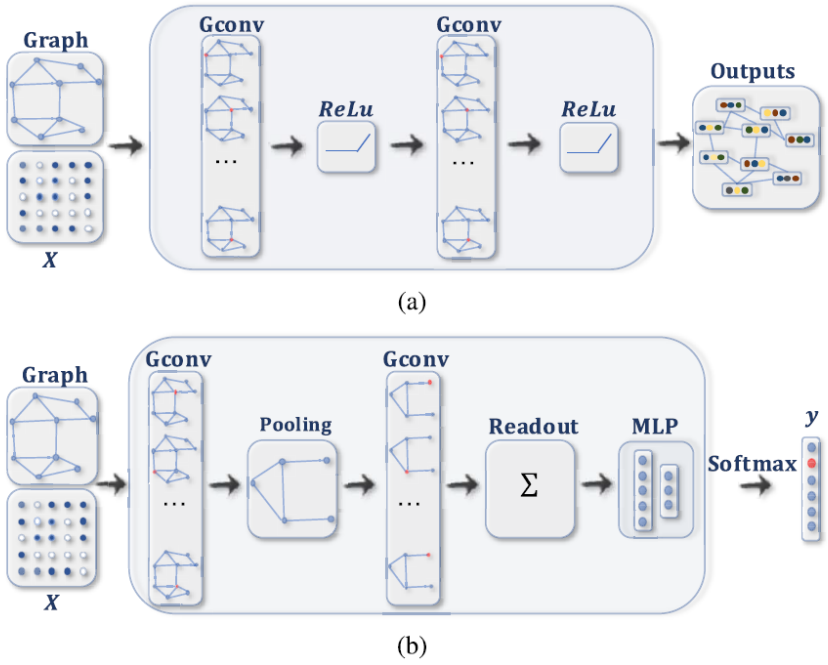


Figure 2.12: Two examples of GNNs with different goals from [21]

GNN for Multiple Instance Learning

In the GNN-based MIL framework proposed in [22], they designed a graph neural network to address a multiple instance learning problem by converting each bag into

a graph and capturing an expressive bag representation through graph convolutional layers. The overall pipeline consists of two main stages:

1. **Graph building:** Each instance in the bag is considered a graph node. To define the adjacency matrix and obtain the graph edges, pairwise Euclidean distances between instance embeddings are computed. The adjacency matrix is then constructed following:

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } d(\mathbf{h}_i, \mathbf{h}_j) < \eta \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

   where η is a threshold hyperparameter. Depending on the chosen value for η, graph is denser or sparser: $\eta = 0$ produces an edgeless graph (reducing the model to independent instance processing), whereas $\eta = +\infty$ yields a fully connected graph. This construction introduces an conceptual bias because instances that are close in feature space are assumed to be correlated. Alternative strategies (e.g., k-nearest neighbors graphs or learnable adjacency matrices) could also be considered and may affect performance and computational complexity.

2. **Graph representation learning:** Once the graph is constructed, a GNN embedding network, composed of one or more Graph Convolutional layers, computes updated node representations:

$$\mathbf{Z} = GNN_{embed}(\mathbf{X}, \mathbf{A}), \tag{2.14}$$

   where $\mathbf{Z} \in \mathbb{R}^{k \times d'}$, and d' could be different than d as the feature dimension could be changed. The number of nodes remains unchanged, but embeddings are enriched via message passing.

At this stage, each node representation encodes both its own features and contextual information from related instances within the bag, but the number of graph nodes is unchanged. To obtain a single bag-level embedding, the framework adopts the **differentiable pooling (DiffPool)** algorithm from [23]. DiffPool learns a hierarchical clustering of nodes and generates coarsened graphs in a fully differentiable way. This algorithm consists in two operations:

1. **Assignment matrix:** A second GNN, $GNN_{cluster}$, computes a soft assignment matrix:

$$\mathbf{S} = softmax(GNN_{cluster}(\mathbf{X}, \mathbf{A})) \tag{2.15}$$

   where $\mathbf{S} \in \mathbb{R}^{k \times C}$, and C is the fixed number of clusters. Each row of $\mathbf{S}$ represents a probability distribution over clusters for a node.

2. **Soft pooling:** From the assignment matrix, a new reduced graph is obtained, where the new node embeddings and adjacency matrix are computed as:

$$\mathbf{Z}^* = \mathbf{S}^T\mathbf{Z} \in \mathbb{R}^{C\times d} \tag{2.16}$$

$$\mathbf{A}^* = \mathbf{S}^T\mathbf{A}\mathbf{S} \in \mathbb{R}^{C\times C} \tag{2.17}$$

The new graph has C nodes, where C is a hyperparameter of the model. The matrix containing node embeddings $\mathbf{Z}^*$ is updated aggregating nodes in a way such that instances with higher probability of belonging to that cluster bring higher contribution to the embedding of that cluster node. If C is set to 1, then the coarsened graph has only one node and one feature embedding that corresponds to the final bag embedding. If C is set to 2 and our task is a binary classification, the probabilities of the assignment matrix could be interpreted as the probability of each instance to belong to either one of the two classes and an extra operation such as max pooling or concatenation is needed to obtain a single feature vector.

Differentiable pooling can also be repeated for more than one time, aggregating every time the graph in a smaller subgraph. Doing so allows a hierarchical learning of the correlations between instances of each bag. If C is not set to 1, than an additional layer of GNN_{embed} can be applied to have an additional step of information passing before the final classification.

The Graph Convolutional layer used in [22] is based on GraphSAGE layer from [24], which performs message passing with the mean as aggregator:

$$\mathbf{h}_i^{(k)} = act(\mathbf{W}\cdot \text{MEAN}(\mathbf{h}_j^{(k-1)}, \forall j \in \mathcal{N}(i)\cup\{i\}) \tag{2.18}$$

where $\mathbf{W}$ is a learnable weight matrix and the activation function *act* is LeakyReLU, a popular alternative to ReLU that learns also from negative inputs. GraphSAGE is inductive, meaning that it can generalize to unseen graphs or bags at test time, an important property in MIL settings where bags vary in size and structure.

Chapter Summary

This chapter provided the theoretical foundations of Multiple Instance Learning, reviewing both classical and deep learning-based MIL approaches, with particular focus on attention-based models, transformer-based architectures, and graph neural network-based formulations. Each paradigm models relationships among instances within a bag differently, ranging from independent attention weighting to explicit relational modeling through graph structures. Having established the theoretical background and architectural principles of these methods, the next chapter describes their practical implementation within this work.

Chapter 3

Materials and Methods

This chapter describes the methodological framework adopted to conduct the experimental comparison of the Multiple Instance Learning (MIL) models introduced in Chapter 2. While the previous chapter focused on the theoretical foundations and architectural principles underlying attention-based, transformer-based, and graph neural network-based MIL approaches, the present chapter details their practical implementation in the context of ovarian tumor differential diagnosis from ultrasound data.

The first part of the chapter focuses on the experimental setup. It begins with a description of the dataset curation process, including the filtering criteria applied to obtain the final cohort used in this study. The dataset splitting strategy is then presented, based on stratified k-fold cross-validation, together with the preprocessing steps applied to standardize the input images before feature extraction. Subsequently, the clinical problem is formalized within the Multiple Instance Learning framework by defining bags and instances in the specific context of the available ultrasound data. The training protocol adopted for all models is then described, including the loss function, the optimization strategy, and the main training parameters such as batch size and number of epochs. The evaluation metrics used to assess model performance are also introduced, with particular attention to the metric considered most appropriate for this setting.

The second part of the chapter focuses on the implementation of the considered architectures. For each model, the main design choices are described together with the set of hyperparameters explored during the experimental analysis.

Overall, the objective of this chapter is to provide a clear and rigorous description of the experimental design, ensuring reproducibility and enabling a fair comparison between the different MIL architectures considered in this work.

3.1 Dataset Curation

This section describes the composition of the original dataset and the filtering procedure applied to obtain the final set used in this study.

Original Dataset

The initial dataset consists of clinical cases, each identified by an anonymous code and associated with a variable number of ultrasound acquisitions, hereafter referred to as *items*. An item can be either a static image or a video sequence. The number of items per clinical case varies and ranges from one to four. For each clinical case, the following information is provided:

- a **risk class**, defined as *benign*, *malignant*, or *borderline*;
- a **histological diagnosis**, specifying the tumor subtype determined by histopathological examination.

Overall, the original dataset included 493 clinical cases corresponding to 828 items, subdivided into 598 images and 230 videos.

Filtering Procedure

To formulate a well-defined binary classification task and ensure sufficient representation of each class, a filtering procedure was applied at the clinical-case level. First, borderline cases were excluded. Only benign and malignant cases were retained, allowing the problem to be formulated as a binary classification task without introducing ambiguity related to intermediate-risk categories.

Second, a selection based on histological subtype was performed. Rather than including all tumor types, only sufficiently represented classes and with higher discriminative features were retained. This choice reduces class heterogeneity and mitigates extreme data sparsity, which could negatively affect the learning process. After filtering, the following four histological subtypes were preserved: endometrioma, cystadenoma fibroma, fibroma and epithelial invasive carcinoma.

Final Dataset Composition

The final curated dataset consists of 352 clinical cases and 567 items. Table 3.1 summarizes the distribution of clinical cases across diagnostic categories, while Table 3.2 reports the distribution of items. As shown in Table 3.1, the dataset presents a moderate class imbalance, with benign cases representing 58% of the cohort and malignant cases 42%. This imbalance is explicitly considered during dataset splitting and in the selection of evaluation metrics.

Class	Number of cases	Percentage
Benign (total)	204	58%
Endometrioma	24	6.8%
Cystadenoma fibroma	135	38.4%
Fibroma	45	12.8%
Malignant (Epithelial invasive)	148	42%
Total	**352**	**100%**

Table 3.1: Distribution of clinical cases in the final dataset with associated percentage

Item type	Number of items
Images	408
Videos	159
Total	**567**

Table 3.2: Distribution of items in the final dataset

The presented dataset was used for training and validation processes. No additional dataset was available for the testing phase, which for this reason was not performed. The performances reported in this thesis all refer to the results obtained on the validation set. The lack of a test set that is never seen during the optimization process, implies that no strong conclusions on the generalizability of different models architectures can be derived.

3.2 Dataset splitting and preprocessing

In this section it is reviewed the splitting procedure of the dataset into training and validation sets, explaining what is the principle of stratified k-fold cross validation. Then frame extraction and image preprocessing steps are explained.

3.2.1 Splitting Strategy

Model evaluation was performed using k-fold cross-validation to split the dataset into training and validation sets. In this strategy, the dataset is divided into k mutually exclusive subsets (folds) of approximately equal size. At each iteration, $k-1$ folds are combined to form the training set, while the remaining fold is used for validation. This procedure is repeated k times, each time selecting a different fold as the validation set. Final performance metrics are obtained by averaging the results across all folds. This method is frequently used when needing splitting

dataset into training and validation sets, because it provides a less biased estimate of model performance on unseen data compared to a single train/test split, and it utilizes all data for both training and validation.

Splitting was performed at the **clinical-case level**. All items (static images and extracted frames) associated with a given clinical case were kept within the same fold. This choice of procedure prevents data leakage, ensuring that no information from a patient included in the training set appears in the validation set.

Given the moderate class imbalance in the dataset (58% benign and 42% malignant cases), a **stratified k-fold cross-validation** strategy was adopted. In stratified splitting, each fold preserves approximately the same class distribution as the overall dataset. Consequently, both the training and validation sets in each fold maintain a similar proportion of benign and malignant cases, ensuring a fair and stable evaluation across folds. A schematic example of how stratified k-fold cv is performed in the case of a dataset with three unbalanced classes is shown in Figure 3.1. In the example k is equal to 4, and red portions correspond to data in validation set, while blue parts are the ones used for the training.

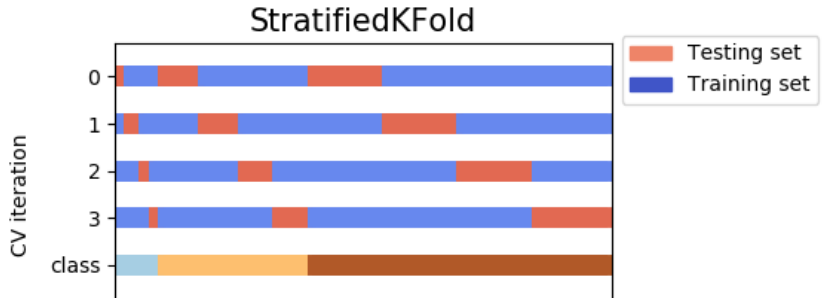


Figure 3.1: Example of stratified k-fold cross validation

3.2.2 Frame extraction from videos

Video sequences were converted into images by extracting a fixed number of frames from each video. Frames were sampled uniformly along the temporal axis to ensure coverage of the entire sequence while avoiding redundancy due to consecutive highly similar frames. The number of extracted frames per video is treated as a hyperparameter and is optimized during model development. This approach allows videos to be integrated in the dataset in the same way as static images within the same learning framework. In fact, once having extracted frames, no distinction is made between these two types of data and possible correlations among frames coming from the same video can be captured only when training the models.

3.2.3 Dataset preprocessing

After frame extraction, all items in the dataset correspond to static images and can therefore be processed using a unified pipeline. Since images originate from different hospitals and are acquired by different clinicians, they exhibit substantial variability in spatial resolution and dimensions. A simple preprocessing procedure was therefore adopted in order to standardize all inputs before feeding them to the learning models.

The preprocessing pipeline consists of the following steps:

- **Resizing.** Each image is resized to a square resolution of 256×256 pixels. This dimension is slightly larger than the input size required by the feature extractor and allows a subsequent cropping operation.
- **Center crop.** The resized image is centrally cropped to obtain the spatial resolution required by the feature extractor. In the case of DenseNet121, this corresponds to 224×224 pixels.
- **Tensor conversion.** Finally, images are converted into numerical tensor representations suitable for input to deep learning models.
- **Normalization.** Pixel values are normalized according to the statistics of the dataset used to pretrain the feature extractor.

This simple preprocessing pipeline does not completely remove all annotations or non-diagnostic regions that may be present in the images, such as textual overlays or areas outside the ultrasound acquisition region. Nevertheless, it provides a consistent and reproducible preparation of the data while keeping the focus of the study on the comparison of different MIL architectures rather than on advanced image preprocessing techniques. Future work could further improve this stage by introducing more specialized preprocessing procedures, such as explicitly isolating the fan-shaped ultrasound acquisition region or removing textual annotations.

After these operations, all items are represented as homogeneous tensors suitable for feature extraction. Having defined how raw ultrasound acquisitions are converted into model-ready inputs, the next step is to formalize how clinical cases and their associated images are structured within the Multiple Instance Learning framework.

3.3 MIL Problem Formulation

After preprocessing, each clinical case is represented as a set of numerical tensors representing ultrasound images. This structure can be naturally translated to a Multiple Instance Learning (MIL) formulation.

Let $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}$ denote the dataset composed of N clinical cases. Each clinical case corresponds to a bag X_i, defined as a set of instances:

$$X_i = \{x_{ij}\}_{j=1}^{n_i},$$

where $x_{ij} \in \mathbb{R}^{H \times W \times C}$ represents the j-th image associated with the i-th clinical case, and n_i denotes the number of images (instances) available for that case. Due to the variability in the number of acquisitions per patient, n_i is not constant across the dataset. Each bag B_i is associated with a binary label $Y_i \in \{0,1\}$, where $Y_i = 0$ denotes a benign tumor and $Y_i = 1$ denotes a malignant tumor. Importantly, labels are available only at the bag level, i.e., at the clinical case level. No instance-level annotations are provided.

Under the standard binary MIL assumption, as already seen in Section 2.2.1, a bag is labeled as positive, corresponding to a malignant tumor, if at least one of its instances is positive while it is labeled as negative, corresponding to a benign tumor, if all its instances are negative. Formally,

$$Y_i = \begin{cases} 1 & \text{if } \exists\, j \text{ such that } x_{ij} \text{ is malignant,} \\ 0 & \text{if } \forall\, j,\ x_{ij} \text{ is benign.} \end{cases}$$

In the present clinical setting, this assumption is consistent with the diagnostic process. A tumor is classified as malignant if malignant characteristics are observed in at least one acquisition. Conversely, a tumor is considered benign only if no malignant evidence is present across all available images.

3.4 Training Protocol

Having defined training and validation sets in a proper MIL formulation applied to our clinical setting, we can go more into details with the training steps to learn the predictive model. First of all, since instance-level labels are not accessible, the model must implicitly learn to identify discriminative images or frames that contribute to the overall diagnosis. Therefore, the learning objective consists in estimating a function

$$f : B_i \to [0,1],$$

that maps each bag to a probability of malignancy. This function depends on some fixed hyperparameters, such as the learning rate and the batch size, and some learnable parameters θ, that are the weights of the neural network layers, which are iteratively updated at each training step with the goal of minimizing a loss function.

Loss function

The loss function considered for this problem is the **Binary Cross Entropy (BCE)**, which is typically used for binary classification problems. It is formulated as following:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\Big(Y_i \cdot \log(p_i) + (1 - Y_i) \cdot \log(1 - p_i)\Big) \tag{3.1}$$

where Y_i is the ground truth label of bag i, p_i is the bag-level probability of the positive class (malignancy) predicted by the model f and N is the total number of bags in the training set. If the true label is positive, only the first term of 3.1 affects the loss function, whereas if $Y_i = 0$ only the second term gives a contribution. The more different the predicted probability is with respect to the true value, and the higher contribution that term gives to the whole summation.

Actually, the outputs of the implemented models are logits $\in \mathbb{R}$ and to convert them into probabilities, a sigmoid function must be applied to them. This operation is combined with the evaluation of the loss, by considering the BCE with logits loss function which takes as input the logits $\{x_1, ..., x_N\}$ given by the classifier and directly computes the sigmoid $\sigma(x_i)$ and the loss contribution with 3.1.

Optimizer and learning rate

To minimize the loss function, its gradient with respect to the model parameters is computed and used to iteratively update the parameters through an optimization algorithm. In its simplest form, this procedure corresponds to gradient descent, where parameters are updated by moving in the opposite direction of the gradient. However, plain gradient descent often suffers from slow convergence, sensitivity to the choice of learning rate, and instability when training deep neural networks. For this reason, more advanced optimization strategies are typically adopted.

In this project, the **Adam** (Adaptive Moment Estimation) optimizer [25] has been employed. Adam is a first-order optimization method, meaning that it relies solely on first derivatives of the loss function and does not require second-order information such as the Hessian matrix. Specifically, Adam maintains two exponentially decaying moving averages for each parameter: the first moment (the mean of past gradients) and the second moment (the uncentered variance of past gradients). The first moment estimate smooths the update direction and reduces oscillations. The second moment estimate rescales the update by accounting for the magnitude of past gradients. As a consequence, each parameter is updated with an individual effective learning rate. Parameters that consistently exhibit large gradients are automatically assigned smaller step sizes, while parameters with small or sparse gradients receive relatively larger updates. This adaptive rescaling

improves numerical stability, accelerates convergence, and reduces the need for extensive manual tuning of the learning rate.

Algorithm 2 summarizes the complete update procedure. The main hyperparameters of Adam are the learning rate γ, the exponential decay rates β_1 and β_2 controlling the first and second moment estimates, and the small constant ϵ introduced for numerical stability.

Algorithm 2 Adam algorithm from [25]

Require: γ learning rate
Require: $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective loss function with parameters θ
Require: : θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \frac{\gamma}{(\sqrt{\hat{v}_t} + \epsilon)} \cdot \hat{m}_t$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

In this project the decay rates and the ϵ constant are kept fixed to the default values: $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$, while the learning rate is tuned to some values $\gamma \in [10^{-5}, 10^{-3}]$ to find for each model the value that brings better performance. In next sections, when analyzing the implementation configuration of each model, the exact values of learning rates that have been tested for every case are also given.

Batch size

During training, the dataset is not processed as a whole but it is divided into smaller subsets called batches. A batch consists of a fixed number of training samples that are forwarded through the network before computing the loss and performing a parameter update. This strategy, commonly referred to as mini-batch training, provides a compromise between full-batch gradient descent, which is computationally expensive, and stochastic gradient descent, which updates

parameters after each individual sample and may introduce high variance in the updates.

The batch size represents the number of samples contained in each batch and directly influences memory usage, gradient stability, and convergence behavior. In this work, the batch size was treated as a tunable hyperparameter and empirically evaluated between the values of 32 and 64, selecting the one that yielded the best validation performance for each model separately.

Epochs

An epoch corresponds to one complete pass of the entire training dataset through the network. During a single epoch, all training samples are processed once, typically divided into multiple mini-batches, and the model parameters are updated after each batch. Training for multiple epochs allows the network to progressively update its parameters, as the optimization algorithm iteratively reduces the loss function across repeated exposures to the data.

The number of epochs is a crucial hyperparameter, as too few epochs may lead to underfitting, while too many may cause overfitting and unnecessary computational cost. In this work, a different number of epochs was selected for each model, depending on its convergence behavior and validation performance. The specific number adopted for each architecture is reported in the corresponding model implementation sections.

3.4.1 Overall Training Procedure

Having defined the loss function, optimizer, batch size and number of epochs, the complete training procedure can be formalized. The objective of the training loop is to iteratively update the model parameters θ so as to minimize the loss function over the training set.

Given the training dataset $\mathcal{D}_{\text{train}} = \{(X_i, Y_i)\}_{i=1}^{N_{\text{train}}}$, the data are divided into mini-batches of size n_b, for a total of N_b mini batches. For each epoch the model processes all batches sequentially and the loop produces a training loss for each step. For each batch, predictions are computed, the loss is evaluated, gradients are obtained through backpropagation, and the optimizer updates the parameters accordingly, given its learning rate. The training loss is accumulated over all batches and averaged at the end of each epoch to monitor convergence. The final outputs after all epochs are the updated and optimized parameters of the model and the training loss as a function of epochs. The overall training loop is summarized in Algorithm 3.

Algorithm 3 Training loop

Require: Training set $\mathcal{D}_{\text{train}} = \{(X_i, Y_i)\}_{i=1}^{N_{\text{train}}}$
Require: Batch size n_b, number of epochs n_e
Require: Model $f(X; \theta)$
Require: Loss function $\mathcal{L}(Y, \hat{Y})$
Require: Optimizer with learning rate γ
  for $e = 1$ to n_e **do**
    $\mathcal{L}_{\text{train}} \leftarrow 0$
    for each batch $(X_i, Y_i) \subset \mathcal{D}_{\text{train}}$ of size n_b **do**
      1) Forward pass: $\hat{Y}_i \leftarrow f(X_i; \theta)$
      2) Compute batch loss: $\ell_i \leftarrow \mathcal{L}(Y_i, \hat{Y}_i)$
      3) Accumulate loss: $\mathcal{L}_{\text{train}} \leftarrow \mathcal{L}_{\text{train}} + \ell_i$
      4) Backpropagation: compute $\nabla_\theta \ell_i$
      5) Optimizer step: update θ
    end for
    $\mathcal{L}_{\text{train}} \leftarrow \mathcal{L}_{\text{train}} / N_b$
  end for
  return Optimized parameters θ^* and training loss per epoch

3.5 Evaluation Metrics

The model performance is evaluated through several quantitative metrics, that measure how well the predicted labels match the true ones. The metrics computed for each model of this project are: accuracy, precision, recall (sensitivity), specificity, F1-score and area under the ROC curve (AUC). Each of them is a single value that can be computed at the end of each epoch. Therefore, during training, they can be tracked as functions of the epoch to observe how performance evolves as the model parameters are updated. In general, higher values indicate better performance, although the interpretation of each metric depends on the clinical context and on the class distribution. In this section a proper definition for each previously mentioned metric is provided, with later some considerations on which is the most informative one for the specific task of differential diagnosis treated in this project.

3.5.1 Metrics definitions

Let Y_i be the true label corresponding to bag X_i and $\hat{Y}_i$ the label predicted by the model, for $i \in \{1, ..., N\}$. In the binary classification setting considered in this

work (benign vs malignant), we define:

$$TP = \#\{(Y = 1) \wedge (\hat{Y} = 1)\}$$
$$TN = \#\{(Y = 0) \wedge (\hat{Y} = 0)\}$$
$$FP = \#\{(Y = 0) \wedge (\hat{Y} = 1)\}$$
$$FN = \#\{(Y = 1) \wedge (\hat{Y} = 0)\}$$

where the meaning is:

- **TP (True Positives):** number of malignant cases correctly predicted as malignant
- **TN (True Negatives):** number of benign cases correctly predicted as benign
- **FP (False Positives):** number of benign cases incorrectly predicted as malignant
- **FN (False Negatives):** number of malignant cases incorrectly predicted as benign

Starting from these quantities it is possible to define all the metrics evaluated for each model.

Accuracy

Accuracy computes the overall fraction of correctly predicted bags, considering both negative and positive classes:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

It provides a global view of model performance but does not distinguish between the two classes.

Precision

Precision computes the fraction of correctly predicted positive labels over all predicted positive labels.

$$Prec = \frac{TP}{TP + FP} \tag{3.3}$$

It reflects the ability of the classifier to avoid false positives. In the medical context, high precision means that when the model predicts a malignant lesion, it is likely to be truly malignant.

Recall (Sensitivity)

Recall, also called Sensitivity or True Positive Rate (TPR), measures how many of the actual positive cases are correctly identified:

$$Rec = \frac{TP}{TP + FN} \tag{3.4}$$

It represents the ability of the classifier to detect malignant cases. In clinical applications, recall is particularly important because missing a malignant case (false negative) may have serious consequences.

Specificity

Specificity, also called True Negative Rate (TNR), measures how many of the actual negative cases are correctly classified:

$$Specif = \frac{TN}{TN + FP} \tag{3.5}$$

It is the counterpart of recall for the negative class and quantifies the ability of the model to correctly identify benign cases.

F1-score

The F1-score is the harmonic mean of precision and recall:

$$F1 = \frac{2}{\frac{1}{Prec} + \frac{1}{Rec}} = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \tag{3.6}$$

It provides a single metric that balances false positives and false negatives. Because it combines precision and recall, it is particularly informative when dealing with imbalanced datasets, where one class is more represented than the other.

AUC

AUC refers to the Area Under the Receiver Operating Characteristic (ROC) Curve. The ROC curve is obtained by plotting:

- True Positive Rate (TPR) = Recall
- False Positive Rate (FPR) = $\frac{FP}{FP+TN} = 1 - Specificity$

for different decision thresholds. When applying the model to one sample, it outputs a probability p for the positive class. To obtain a binary prediction, this probability is compared to a decision threshold. In standard practice, the threshold

is set to 0.5: if $p \geq 0.5$, the sample is classified as positive otherwise, it is classified as negative. However, the threshold can be varied to trade off recall and specificity. The ROC curve visualizes this trade-off over all possible thresholds.

The area under this curve (AUC) summarizes the overall discriminative ability of the model. The larger the area is, the higher also the ability is, with the two extremes as:

- AUC = 1 indicates perfect discrimination,
- AUC = 0.5 corresponds to random guessing.

Unlike accuracy or F1-score, AUC is threshold-independent and evaluates how well the model ranks positive samples higher than negative ones.

3.5.2 Choice of the Primary Metric

To compare different architectures and select the best-performing model, all the metrics are analyzed, but the F1-score is used as the primary metric. This choice is motivated by the class imbalance present in the dataset, where benign cases are more frequent than malignant ones. In such situations, accuracy alone can be misleading. In a general example, if 95% of the samples are benign and the model always predicts "benign", it would achieve 95% accuracy while completely failing to detect malignant cases. In medical diagnosis, correctly identifying the minority class (malignant lesions in our case) is crucial. The F1-score, by combining precision and recall, provides a more balanced and informative evaluation of performance in this scenario. It ensures that the model does not achieve good results simply by favoring the majority class.

Although the F1-score is used as the primary metric for model comparison, it is not considered alone. In practice, the validation loss is monitored together with the F1-score to select the best-performing model. A model may achieve a high F1-score at a certain epoch while already starting to overfit the training data, which is typically reflected by an increasing validation loss. For this reason, model selection is based on the joint observation of validation loss and F1-score trends across epochs, favoring configurations that achieve high F1-score without showing clear signs of overfitting.

3.5.3 Validation loop

Performance metrics are evaluated in the validation phase, which is performed at the end of each training epoch. More precisely, for each epoch first the model is trained over all batches in the training set; then it is evaluated on the validation set. For each validation batch, predicted probabilities are provided by the application of

the model to the validation batch, then the batch loss is computed and accumulated. At the end of the loop, the validation loss is averaged over all validation batches and all evaluation metrics (Accuracy, Precision, Recall, Specificity, F1-score, AUC) are computed over the entire validation set. Binary predictions for metric computation are obtained using a decision threshold equal to 0.5, while AUC is computed using the raw predicted probabilities. This procedure is summarized in Algorithm 4.

Algorithm 4 Validation loop

Require: Validation set $\mathcal{D}_{val} = \{(X_i, Y_i)\}_{i=1}^{N_{val}}$
Require: Batch size n_b
Require: Model $f(X;\theta)$ and parameters θ
Require: Loss function $\mathcal{L}(Y, \hat{Y})$
  $\mathcal{L}_{val} \leftarrow 0$
  for each batch $(X_i, Y_i) \subset \mathcal{D}_{val}$ of size n_b **do**
    1) Forward pass: $\hat{Y}_i \leftarrow f(X_i;\theta)$
    2) Compute batch loss: $\ell_i \leftarrow \mathcal{L}(Y_i, \hat{Y}_i)$
    3) Accumulate loss: $\mathcal{L}_{val} \leftarrow \mathcal{L}_{val} + \ell_i$
  end for
  4) $\mathcal{L}_{val} \leftarrow \mathcal{L}_{val}/N_b$
  5) Compute metrics: $Acc, Prec, Rec, Specif, f1-score, AUC$
  return $\mathcal{L}_{val}$ and metrics per epoch

3.6 Models Implementation

In this section the implementation of each model is analyzed, starting from supervised baseline and then going to the MIL-based architectures. For each model, the corresponding hyperparameters are listed, both fixed ones and the ones which are tuned in sweep configurations.

3.6.1 Baseline Model Implementation

Starting from the study on supervised models applied to the differential diagnosis of ovarian tumors from a dataset of ultrasound items discussed in Section 2.3, a baseline model has been devised for our case.

The model used is based on a DenseNet121 architecture, where the final classifier layer, originally designed for ImageNet classification into 1000 classes, is replaced by a Sequential module composed of:

1. One **linear** layer with output dimension equal to 256;

2. One **layer normalization**, to stabilize the learning procedure;
3. One **ReLU** activation function;
4. One **dropout** layer;
5. One final **linear** layer producing one single output value, corresponding to the logit for the positive class (malignancy).

This modification allows the network to perform binary classification.

The dropout layer is introduced to reduce overfitting by randomly deactivating a fraction of neurons during training. The percentage of deactivated neurons is determined by the dropout rate: if this value is set to 0, the layer has no effect. This hyperparameter is tuned and optimized during the training process.

Fine Tuning and Freeze Strategy

The initial weights of the model are set to the default pretrained DenseNet121 weights trained on the ImageNet dataset, except for the newly introduced classifier layers, whose parameters are randomly initialized. During training on the ultrasound dataset, only a subset of the model parameters is updated. To do so two different freeze strategies are considered:

- **Classifier fine-tuning:** all network parameters are frozen except for those of the classifier module. Only the final layers are updated during training.
- **Last block fine-tuning:** in addition to the classifier module, the parameters of the last dense block (the fourth one) are also updated.

As described in the original paper implementation of DenseNet architectures [26], each dense block is composed of multiple dense layers. In particular, the fourth block contains 16 dense layers, and each dense layer includes two convolutional layers with ReLU activations and two batch normalization layers. Updating this additional block significantly increases the number of trainable parameters and therefore the computational complexity. On the other hand, fine-tuning a larger portion of the network may allow better adaptation to the target dataset. This aspect is particularly relevant in our case, since ultrasound images differ substantially from the natural images contained in ImageNet. For this reason, both freeze strategies have been tested and compared.

The optimizations of the fourth dense block and of the classifier module are handled differently. The dense block starts from pretrained weights that have already undergone substantial optimization, although on a different dataset. In contrast, the classifier parameters are randomly initialized and require faster adaptation. Therefore, different learning rates are used: a smaller learning rate

for the dense block and a larger one for the classifier, in order to have larger (and quicker) updates. A fixed ratio between the two is adopted, setting the classifier learning rate to be ten times higher than the block learning rate:

$$lr_{class} = 10 \cdot lr_{block}.$$

In this way, tuning the block learning rate automatically determines the classifier learning rate and only the block learning rate is treated as a tunable hyperparameter.

Aggregation Method

A key limitation of applying a supervised model in this setting is that labels are available only at the bag level, while instance-level labels are not provided. Therefore, an aggregation strategy is required. In this baseline approach, each instance is assigned the same label as the corresponding bag. After this assignment, items are treated independently, without explicitly modeling their belonging to a specific bag. The model is then trained at the instance level by minimizing the loss computed on individual items. This approach mimics the one used in the supervised studies reviewed in chapter 2. It simplifies the problem by converting it into a standard supervised classification task, at the cost of ignoring the multiple-instance structure of the data.

Hyperparameters configuration

The baseline model has high computational complexity due to the large number of layers and parameters in DenseNet121. As a consequence, performing an extensive hyperparameter search was not feasible, since each single training run required several hours. For this reason, many hyperparameters were kept fixed. Additionally, the number of folds for cross-validation was set to 3 instead of 5 to reduce the total number of runs, and the number of training epochs was limited. Table 3.3 reports the hyperparameters sweep configuration, which summarizes both fixed hyperparameters and the ones that have been tuned.

The hyperparameter search was conducted using grid search, meaning that all possible combinations of the selected hyperparameters were evaluated by performing a complete training run for each configuration.

3.6.2 MIL Models Implementation

Each of the four MIL models described in Section 2.4 has been adapted to our dataset and classification task. Since all of them operate on instance-level embeddings rather than raw images, a preliminary feature extraction step is required. This step is common to all MIL models and has been performed once, prior to training

Baseline hyperparameters	
Number of folds	3
Number of epochs	50
Batch size	64
# of extracted frames	16
Freeze strategy	[only classifier, also last block]
Block learning rate	$[10^{-4}, 10^{-5}]$
Dropout rate	[0, 0.3]

Table 3.3: Hyperparamenter sweep configuration for Baseline model

the aggregation architectures. Another common aspect consists in the padding strategy that allows to treat bags with different sizes in the same batch and apply the model uniquely to the entire batch. Other aspects common to all architectures are the Adam optimizer and the binary classification entropy with logits loss. Some hyperparameters are common, but are tuned independently like the learning rate or are fixed differently because of the different model behaviors. In the cases of MIL models the number of folds for cross validation is kept fixed to 5 and the number of epochs per run could be very large, because simulations are very quick, as embeddings are precomputed and not extracted every time from scratch.

After reviewing the common choices regarding the feature extractor and the padding strategy, and after making a small observation about how we treated attention scores, each of the four models with its own hyperparameters is described independently.

Feature Extraction

To obtain instance embeddings, a single feature extractor has been used for all models. It consists of a DenseNet121 architecture pretrained on ImageNet, consistently with the baseline model. In this case, however, the final classifier layer is replaced by an Identity layer, so that the network outputs the feature vector produced by the last convolutional block.

The resulting embedding has dimension 1024, corresponding to the output size of the layer preceding the original classifier. The feature extractor is kept frozen and is not fine-tuned during MIL training. Feature extraction is performed offline, separately from the MIL aggregation models. For each bag, embeddings of all its instances are computed and saved locally as 1024-dimensional tensors. Embeddings are stored in separate folders depending on the number of frames extracted per video, with one file per bag. The embedding shape at the bag level is then $n_i \times 1024$ where n_i is the number of items contained in bag i. During training of the different MIL models, the precomputed embeddings are loaded and used as input to the

corresponding aggregation architecture.

Mini-batches and masking

For each model, both the training and validation datasets are processed in mini-batches. During the forward pass, the model operates on the entire batch simultaneously in order to exploit the computational advantages of batch processing. A challenge arises from the nature of Multiple Instance Learning: bags may contain a different number of instances. However, tensors within a batch must have consistent dimensions in order to be processed together.

To address this issue, a **masking** strategy is adopted. First, all bags in a batch are padded to match the size of the largest bag in that batch. Padding is performed by adding fictitious instances so that all bags share the same number of items. Clearly, these artificial elements must not influence the model's predictions. For this reason, each instance is associated with a mask value, which is a binary indicator specifying whether the instance is real or padded. Real instances are assigned a value of 1, while padded instances are assigned a value of 0. During the forward pass, this mask is used to ensure that padded elements are ignored by the model, preventing them from contributing to the aggregation or attention mechanisms.

Attention Scores

An important observation regarding the first three MIL architectures considered in this work, ABMIL, DSMIL, and TransMIL, is that they provide attention mechanisms capable of producing attention scores for individual instances within each bag. These scores can be interpreted as indicators of the relative importance of each instance in contributing to the final bag-level prediction, thus providing a potential source of model interpretability.

However, in the present dataset no labels or annotations are available at the item level that could be used to assess the reliability of these attention scores. As a consequence, it is not possible to quantitatively evaluate whether the instances receiving higher attention weights correspond to clinically relevant images or frames. For this reason, although the considered architectures internally compute attention scores, our implementation focuses exclusively on the bag-level predictions and does not explicitly extract or analyze these scores. Nevertheless, attention scores remain an interesting direction for future work, particularly if instance-level annotations become available.

ABMIL Adaptation

For Attention-Based MIL, an attention layer has been implemented following the formulation proposed in [18]. The main hyperparameter of this layer is the attention

dimension, which determines the hidden dimensionality of the trainable parameters $\mathbf{w}$ and $\mathbf{V}$. Additionally, the activation function used inside the attention mechanism is not restricted to the standard hyperbolic tangent proposed in the original work. Other commonly used activation functions have also been considered, in particular ReLU and GeLU activations were tested as long as tanh. GeLU activation function is defined as $\text{GeLU}(x) = x\Phi(x)$, where $\Phi(x)$ is the Gaussian cumulative distribution function, and can be more stable than ReLU because of its differentiability. Last hyperparameter related to ABMIL architecture is the attention mechanism that can be implemented either as the standard formulation in Equation 2.4 or as the gated version described in Equation 2.5.

All these hyperparameters have been tuned in order to identify the configuration that provides the best performance. In particular, first a sweep has been performed on the architecture parameters, keeping fixed the other training parameters. After selecting the best configuration of model parameters, another grid search has been performed keeping fixed this configuration, while tuning hyperparameters related to the training process, like batch size, the number of extracted frames per video and the learning rate. The full configuration explored during the sweep is summarized in Table 3.4.

ABMIL hyperparameters	
Number of epochs	1000
Batch size	[32, 64]
# of extracted frames	[8, 16, 32, 64, 128]
Learning rate	[10^{-3}, 10^{-4}, 10^{-5}]
Attention dimension	[64, 128, 256]
Attention activation	[Tanh, ReLU, GeLU]
Type of attention	[Standard, Gated]

Table 3.4: Hyperparameter sweep configuration for the ABMIL model

DSMIL Adaptation

The implementation of Dual-Stream MIL follows the original formulation proposed in [19], with the addition of several configurable components.

- **Attention dimension:** Similarly to the attention mechanism used in transformer architectures, instance features are projected into a lower-dimensional latent space to produce the query vector $\mathbf{q}$. The information vector $\mathbf{v}$, instead, remains in the original feature space so that the final bag representation preserves the same dimensionality as the instance embeddings. Consequently,

the attention dimension only affects the computation of attention weights, while the resulting bag embedding remains in the original feature space.

If M denotes the original feature dimension and d the hidden dimension of the projection, the transformation is obtained through the projection matrix $\mathbf{W}_q \in \mathbb{R}^{d \times M}$, producing $\mathbf{q} = \mathbf{W}_q \mathbf{h} \in \mathbb{R}^d$. The attention dimension d therefore becomes a tunable hyperparameter of the model.

- **Non-linear transformations:** Additionally to the original formulation proposed in [19], an optional introduction of non-linear transformations for both the query vector $\mathbf{q}$ and the information vector $\mathbf{v}$ is allowed. When non-linearity is enabled for $\mathbf{q}$, the vector is computed through a Sequential module composed of a Linear projection into the attention dimension, followed by a ReLU activation, a second Linear layer maintaining the same dimensionality, and a final Tanh activation. When non-linearity is enabled for $\mathbf{v}$, the vector is computed through a sequence composed of a Dropout layer, followed by a Linear transformation and a ReLU activation function.

  If these options are disabled, the vectors are computed using the original formulation: $\mathbf{q}$ is obtained through a single Linear projection, while $\mathbf{v}$ corresponds directly to the instance embedding (identity mapping).

- **Dropout:** As described above, the dropout layer is applied only when the non-linear transformation of vector $\mathbf{v}$ is enabled. In this case, the dropout rate becomes an additional hyperparameter that can be tuned.

The explored hyperparameter configuration is reported in Table 3.5. An initial grid search was performed by fixing the learning rate to 10^{-4} and training the model for 500 epochs. After identifying the most promising configuration, a second sweep was conducted using lower learning rates and increasing the number of training epochs to 1000. The batch size and the number of extracted frames were kept fixed to the values that provided the best performance in the ABMIL model.

TransMIL Adaptation

Also in the case of Transformer-based MIL the implementation is extended with respect to the original formulation proposed in [20], in order to provide a more flexible implementation.

A first important aspect to clarify is that no masking mechanism is provided to the model during the forward pass. In the original implementation, the model was designed for Whole Slide Image (WSI) datasets, where the number of instances in each bag corresponds to the number of patches extracted from a slide. In that setting, slides are typically divided into a fixed grid of patches, resulting in bags

DSMIL hyperparameters	
Number of epochs	500
Batch size	64
# of extracted frames	16
Learning rate	$[10^{-4}, 5 \cdot 10^{-5}, 10^{-5}]$
Attention dimension	[64, 128, 256]
Nonlinear q	[True, False]
Nonlinear v	[True, False]
Dropout rate	[0, 0.1, 0.2, 0.25, 0.5]

Table 3.5: Hyperparameter sweep configuration for the DSMIL model

with the same number of instances. Consequently, the original architecture does not need to explicitly handle bags of different sizes.

In contrast, in the dataset used in this work the number of instances per bag may vary, since each bag contains a variable number of items. For this reason, an adaptation of the architecture was required in order to handle bags with different sizes. To address this issue, a padding strategy has been introduced. Specifically, the size of each bag in a mini-batch is matched to the size of the largest bag in that batch by repeating already existing instances of the smaller bags. In this way, all bags within the batch reach the same size and can be processed jointly by the Transformer layers. This strategy allows all instances to contribute to the model learning process. However, this repetition may introduce a bias in the attention mechanism, as repeated instances could disproportionately influence the bag representation, potentially affecting the model's generalization.

This architecture introduces several hyperparameters specific to Transformer models, in addition to the standard training parameters shared with the other MIL approaches.

- **Attention dimension:** In the followed implementation of TransMIL, instance features are first projected into a latent embedding space before entering the Transformer layers. The dimensionality of this space, referred to as the attention dimension, determines the size of the queries, keys, and values used in the self-attention mechanism. It therefore corresponds to the internal embedding dimension of the Transformer and represents a tunable hyperparameter of the model.

- **Transformer layers:** The number of Transformer layers can be adjusted to control the depth of the model. In the architecture illustrated in Figure 2.11, the layers correspond to blocks composed of layer normalization and multi-head self-attention mechanisms. At least two layers are required: one applied before the positional encoding step and one after it. Additional layers can be added

after the positional encoding stage, increasing the representational capacity of the model. However, deeper configurations also increase the computational complexity and may lead to training instability.

- **Attention heads:** Multi-head attention consists of several parallel self-attention mechanisms operating on different learned projections of the input embeddings. Each head can capture different relationships between instances in the bag, and their outputs are concatenated to form the final representation. The number of attention heads is therefore another important hyperparameter explored during the hyperparameter sweep.

- **Additional Multi-Layer Perceptron:** Another extension to the original formulation of TransMIL is the option of adding a multi layer perceptron at the end of each self-attention layer. Enabling the addition of these final sequence increases the number of trainable parameters, bringing to higher risks of overfitting, but also constructing a more expressive model. When this final sequence is enabled also a dropout layer is used and dropout rate becomes another tunable hyperparameter.

The explored sweep configuration for the TransMIL architecture is reported in Table 3.6. The number of training epochs was initially set to 100 when using a learning rate of 10^{-4}, since the validation loss tended to overfit rapidly and longer training did not provide further improvements. When lower learning rates were considered, the number of epochs was slightly increased to allow the model sufficient time to converge.

TransMIL hyperparameters	
Number of epochs	100
Batch size	[32, 64]
# of extracted frames	[8, 16, 32, 64, 128]
Learning rate	$[10^{-4}, 5 \cdot 10^{-5}, 10^{-5}]$
Attention dimension	[128, 256, 512]
# of attention layers	[2, 3]
# of attention heads	[4, 8]
Enable MLP	[True, False]
Dropout Rate	[0.1, 0.3, 0.5]

Table 3.6: Hyperparameter sweep configuration for the TransMIL model

GNN-based MIL Adaptation

For the Graph Neural Network-based MIL approach, three different models of increasing complexity have been implemented. All models rely on GraphSAGE layers [24] as graph convolutional operators, while differing in the strategy used to aggregate node information and obtain the final bag representation. The most complex architecture among the three follows the clustering strategy proposed in paper [22] and already analized in Chapter 2.

All models share the same procedure for graph construction. For each bag, a graph is built by computing the Euclidean distance between the instance embeddings and connecting each node to its k nearest neighbors. The parameter k therefore controls the local connectivity of the graph and is treated as a tunable hyperparameter. If the value of k exceeds the number of instances in the bag, the resulting graph becomes fully connected.

The three implemented architectures are described below.

1. **Simple GNN model:** The first and simplest model consists of stacking multiple GraphSAGE layers in order to update the instance embeddings by aggregating information from their local neighborhoods. The number of graph convolutional layers is a tunable hyperparameter.

   Instance embeddings are projected into a lower-dimensional latent space, defined by the attention dimension, which represents the size of the node feature vectors used throughout the graph network. After the sequence of GraphSAGE layers, the graph still contains the same number of nodes as the input graph, with only the node feature representations being updated. To obtain a single bag-level embedding, a global aggregation function is applied over the node embeddings. Three different aggregation strategies have been explored: mean pooling, max pooling, and attention-based pooling using an attention layer. The aggregation method is treated as an additional hyperparameter.

2. **Top-k GNN model:** The second model introduces a node selection mechanism that exploits the learned importance of graph nodes. As in the previous model, a tunable number of GraphSAGE layers is first applied to update the node embeddings. Subsequently, an importance score is learned for each node, and only the most relevant nodes are retained. Specifically, the top K nodes with the highest scores are selected, where K is computed as

   $$K = \lceil ratio \cdot N \rceil$$

   with N representing the number of nodes in the graph and $ratio$ indicating the fraction of nodes to retain. The parameter $ratio$ is included in the hyperparameter search.

After this node selection step, a mean pooling operation is applied to the remaining nodes in order to obtain the final bag embedding. Mean pooling was chosen since it provided the best performance in the Simple GNN model. This architecture represents an intermediate level of complexity, introducing a learnable node selection mechanism without relying on explicit clustering.

3. **Cluster-based GNN model:** The third and most complex model follows the clustering strategy proposed in [22], which employs the differentiable pooling (DiffPool) algorithm. As described in Section 2.4.4, after the graph convolutional layers the model learns an assignment matrix that associates each node with a probability distribution over a set of clusters. Using these assignment probabilities, nodes are aggregated into a smaller set of cluster representations, effectively producing a coarsened graph. The key hyperparameter of this model is therefore the number of clusters C into which the graph is collapsed. Following the recommendation in [22], the number of clusters was tuned between 1 and 2.

   After the pooling step, an additional GraphSAGE layer is applied in order to further propagate information within the reduced graph. When $C = 1$, the resulting representation already corresponds to a single bag embedding, which is directly used for classification. When $C = 2$, two cluster embeddings are produced and the final bag representation is obtained by concatenating the two vectors.

The hyperparameter sweep configuration explored for the three GNN-based models is summarized in Table 3.7.

	Simple GNN	**Top-k GNN**	**Cluster GNN**
Epochs	150		
Batch size	64		
# extracted frames	16		
Learning rate	$5 \cdot 10^{-5}$		
Embedding dim	[128, 256, 512]		
k **(nearest n.)**	[3, 8]	[3, 8]	[3, 8, 15]
# GSAGE layers	[1, 2, 3]	[1, 2, 3]	[1, 2]
Aggreg. method	[mean, max, att.]	mean	–
Top-k ratio	–	[0.2, 0.3, 0.4]	–
# clusters	–	–	[1, 2]

Table 3.7: Hyperparameter sweep configuration for the GNN-based MIL models

3.7 Implementation and Computational Details

All experiments were implemented in Python using the PyTorch deep learning framework. The ABMIL, DSMIL, and TransMIL architectures were implemented using the TorchMIL library [27], a framework specifically designed for Multiple Instance Learning problems that provides modular implementations of several MIL models. Graph-based models were implemented using the PyTorch Geometric library [28], which provides efficient tools for constructing and training graph neural networks.

In order to ensure reproducibility of the results, a fixed seed for randomness was used across all experiments. Training and evaluation metrics were logged and monitored using the Weights & Biases (W&B) platform, which allowed systematic tracking of model performance and hyperparameter configurations throughout the experimental phase. All experiments were executed on a GPU-based computing environment to accelerate training. The specific versions of the software libraries used, together with the hardware configuration and GPU model, are reported below.

	Version
Python	3.11.14
Pytorch	2.9.1
Pytorch Geometric	2.7.0
TorchMIL	1.0.1
W&B	0.23.0
GPU	NVIDIA Quadro RTX 5000

Table 3.8: Computational Details

Chapter 4

Results

In this chapter, the quantitative results obtained for each model are presented. For every architecture, the evolution of the primary metric and the loss function during training is reported. A comparison between the models is then performed in order to identify the configuration that achieves the best performance. Particular attention is devoted to the comparison between Multiple Instance Learning (MIL) models and the baseline approach, with the aim of evaluating whether MIL can provide improvements over a standard supervised instance-level classification strategy. The chapter concludes with a discussion of the results and a final section outlining possible future research directions.

4.1 Performance on Validation Set

In this section, the evolution of the selected evaluation metrics as a function of the training epoch is reported for each model. Several hyperparameter configurations are explored and compared in order to identify the most effective setup. As discussed in Chapter 3, the F1-score is adopted as the primary evaluation metric. After identifying the epoch corresponding to the highest F1-score, a summary table is provided reporting all the metrics computed at that epoch.

4.1.1 Baseline Results

The baseline model consists of a DenseNet121 architecture that performs both feature extraction from the raw ultrasound images and binary classification at the instance level. Since labels are only available at the bag level, the bag label is propagated to all the instances belonging to the bag and each item is treated as an independent training sample.

As described in Chapter 3, the dataset splitting for the baseline model is performed using 3-fold stratified cross-validation. Each hyperparameter configuration is therefore trained three times, once for each fold. The reported metric curves correspond to the average values across the folds. The shaded regions represent the maximum–minimum range observed among the folds and provide an indication of the variability induced by different dataset splits. Lower variability indicates that the model is more robust to variations in the training and validation sets.

Baseline Results for three exemplifying configurations

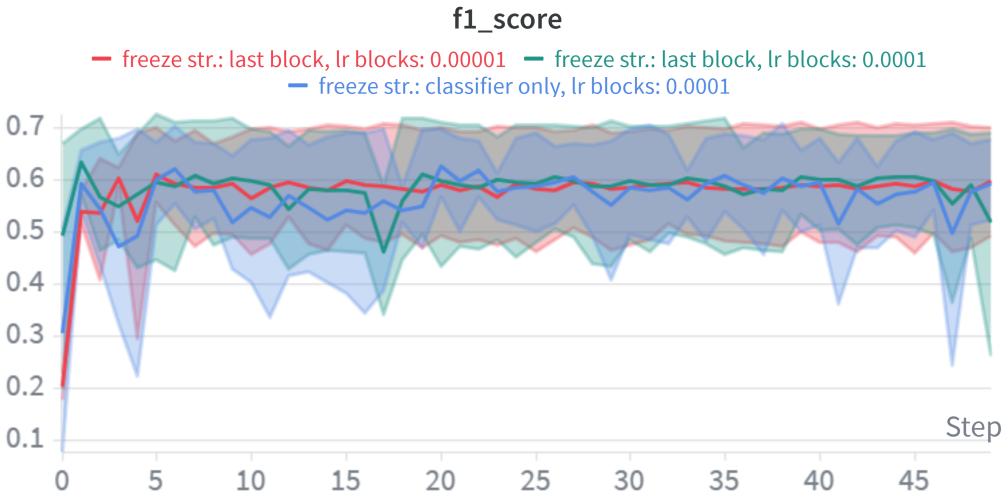
The sweep configuration for the baseline model is presented in Table 3.3. Figure 4.1.1 reports the evolution of the F1-score and validation loss for a reduced set of the most relevant hyperparameter configurations. Only a subset of configurations is shown, since plotting all of them would make the figure difficult to interpret. In particular, the dropout rate is fixed to 0.3 in the reported plots because it consistently produced the best results. The remaining configurations differ in the freeze strategy and learning rate values.

From Figure 4.1a, the highest F1-score corresponds to the configuration shown in green, which trains both the classifier and the last dense block using a block learning rate of 10^{-4} and a classifier learning rate of 10^{-3}. The best F1-score is reached at epoch 2, with a value of 0.63. However, the validation loss for this configuration rapidly increases and shows high variability across folds, as indicated by the large shaded region in Figure 4.1b. This behavior may suggest that the model starts to overfit very early during training, or that the supervision signal derived from propagating bag labels to individual instances introduces noise in the training process.

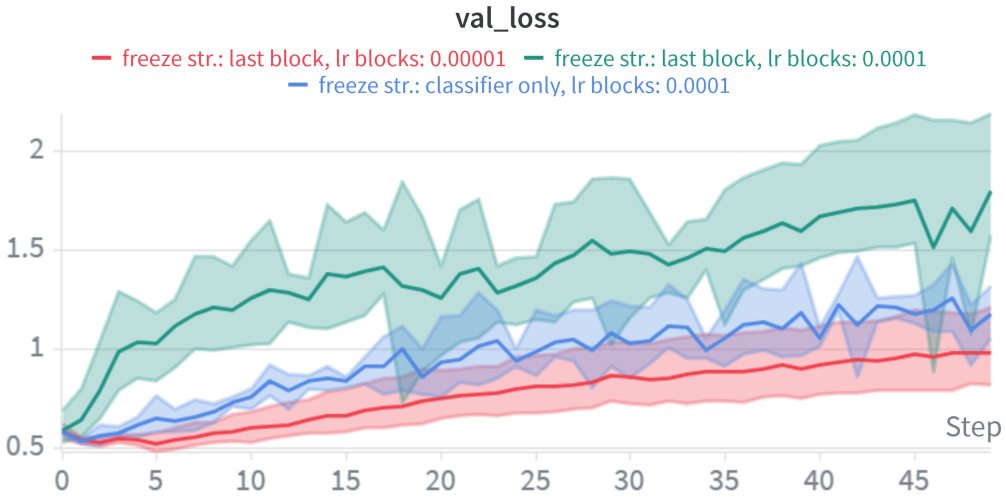
Fixing the learning rate and observing at differences between curves green and blue, the main difference regards the loss, where if training also the last dense block the resulting validation loss increases earlier, faster and has higher folds variability. This could be explained by the increasing complexity of the model, that needs to train a significantly larger amount of parameters. Fixing instead the train strategy and looking at different learning rates, lower learning rates typically imply more stable curves for both plots, but no increase in the values of F1-score, which stays around 0.6.

Another interesting observation regarding validation losses is that for all three configurations there is no clear decreasing trend, not even in the first few epochs. In typical supervised learning scenarios, the validation loss is expected to decrease at least during the early stages of training. The absence of such behavior may indicate that the model is not effectively extracting discriminative information from the input data under the current training setup.

Despite the generally limited performance, the configuration that trains the



(a) Baseline F1-score



(b) Baseline Validation Loss

Figure 4.1: Examples of F1-score and validation loss of Baseline model

last block together with the classifier achieves the highest F1-score and is therefore considered as the reference baseline. Its hyperparameter configuration and corresponding performance are summarized in Table 4.1.

Hyperparameter	Value
Batch size	64
Number of frames	16
Classifier learning rate	10^{-3}
Block learning rate	10^{-4}
Freeze strategy	Last block
Dropout rate	0.3

Metric (epoch 2)	Value
F1-score	0.63
Val accuracy	0.75
Train accuracy	0.93
Recall	0.64
Precision	0.65
Specificity	0.80
AUC	0.80
Validation loss	0.64
Training loss	0.19

Table 4.1: Summary of the best baseline hyperparameter configuration and corresponding performance.

It can be observed that some metrics, such as accuracy and specificity, appear relatively high. However, as discussed in Section 3.5.2, the F1-score is considered the primary evaluation metric since it provides a more reliable measure of performance in the presence of class imbalance.

Observations on baseline performance

The limited performance of the baseline model may be explained by the characteristics of the available supervision. In this approach, the bag label is propagated to all the instances belonging to the bag, meaning that each image or video frame is assigned the same label. However, in ultrasound acquisitions only a subset of frames may actually contain relevant diagnostic information, while many others may be non-informative or visually ambiguous. As a consequence, a significant portion of the training data may receive noisy labels, which can hinder the learning process of an instance-level classifier. Moreover, ultrasound videos often contain many temporally adjacent frames that are highly similar to each other. Treating them as independent training samples may therefore introduce redundancy in the dataset without providing additional discriminative information.

These limitations highlight the potential advantages of Multiple Instance Learning approaches, which are specifically designed to operate with weak supervision at the bag level and to automatically identify the most informative instances within each bag.

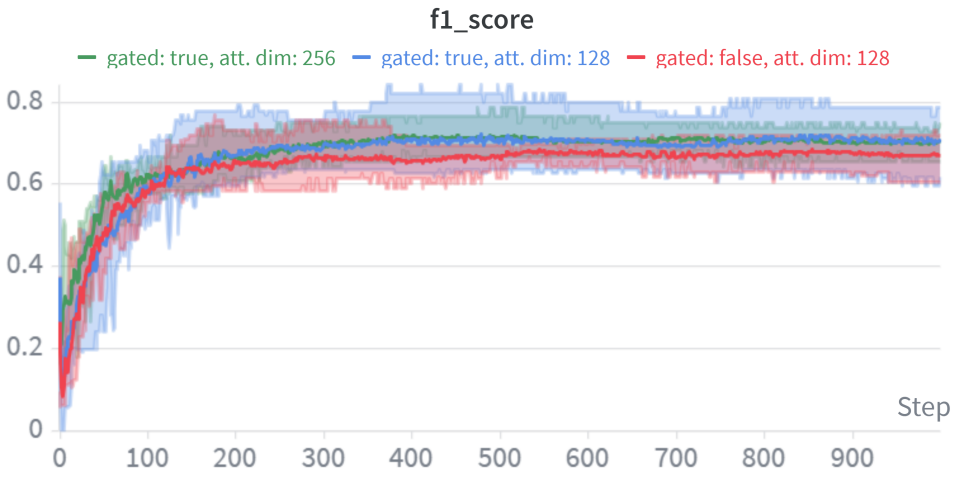
4.1.2 ABMIL Results

The Attention-Based Multiple Instance Learning (ABMIL) model consists of two main components. The first component is the feature extraction, which is performed offline and is shared across all the MIL models considered in this work. The second component is the MIL aggregation module, which is implemented through an attention mechanism. The attention layer assigns a score to each instance within a bag, allowing the model to weigh instances according to their relevance for the classification task. The resulting weighted aggregation produces a bag-level representation, which is then used to perform the final classification. This approach is consistent with the available supervision, since labels are provided only at the bag level. Compared to the baseline approach described in the previous section, the ABMIL model operates directly at the bag level. This allows the model to focus on the most informative instances within each bag rather than assuming that all instances contribute equally to the bag label.

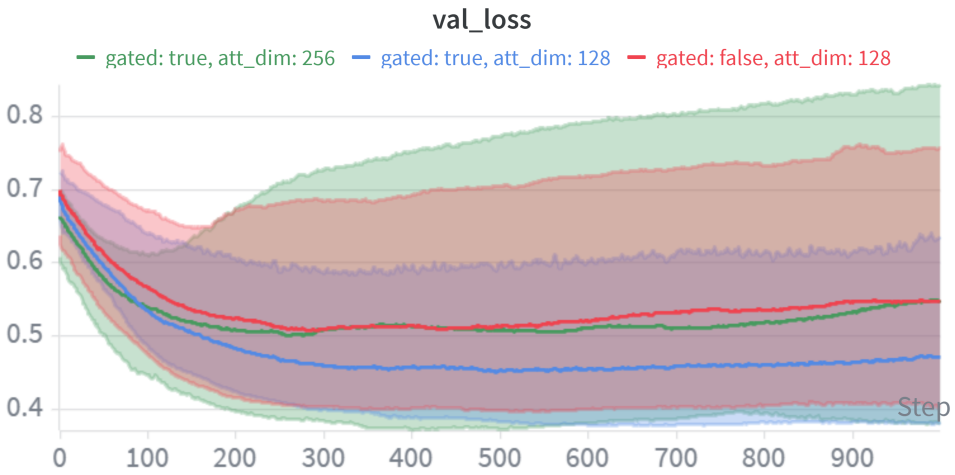
As discussed in the previous chapter, for ABMIL and all other MIL-based models the dataset splitting is performed using a 5-fold stratified cross-validation strategy. For each hyperparameter configuration the reported performance corresponds to the average across the folds.

The hyperparameter sweep configuration for ABMIL is reported in Table 3.4. The tunable hyperparameters can be divided into two groups. The first group includes architecture-related hyperparameters, which are specific to the ABMIL model and include the attention dimension, the attention activation function, and the type of attention mechanism used. The second group includes training-related hyperparameters, which are common to all models, including the baseline, and consist in the number of training epochs, batch size, number of extracted frames per bag and learning rate. The hyperparameter search was conducted in two stages. First, a sweep over the architecture-related parameters was performed while keeping the training-related parameters fixed. After identifying the best architectural configuration, a second sweep was carried out over the training-related hyperparameters.

Figure 4.1.2 shows the evolution of the F1-score and validation loss for three representative configurations which differ in the type of attention mechanism used and the attention dimension. In particular the three reported plots correspond to three configurations that differ for the attention dimension and the type of attention mechanism. The remaining hyperparameters are fixed to the values that produced the best results during the sweep and are summarized in Table 4.2, together with the corresponding performance metrics.



(a) ABMIL F1-score



(b) ABMIL Validation Loss

Figure 4.2: Examples of F1-score and validation loss of ABMIL model

Observations on ABMIL results

From the F1-score plot, it is evident that these three configurations do not bring significant changes in the evolution of this metric. The only separation, even if weak, is between the red line, which corresponds to the configuration with the standard attention mechanism, and the other two, that use gated attention and reach higher values. For this reason the gated attention is preferred among the two mechanisms. Looking instead at the blue and green lines, differing for the attention

Hyperparameter	Value
Batch size	64
Number of frames	16
Learning rate	10^{-4}
Attention dimension	128
Attention activation	Tanh
Attention mechanism	Gated

Metric (epoch 478)	Value
F1-score	0.72
Val Accuracy	0.77
Train Accuracy	0.95
Recall	0.70
Precision	0.74
Specificity	0.80
AUC	0.82
Validation loss	0.45
Training loss	0.20

Table 4.2: Summary of the best ABMIL hyperparameter configuration and corresponding performance.

dimension, F1-score is very similar and both their highest value correspond to 0.72. For this reason the best configuration was chosen looking at the validation loss. Between green and blue losses, the blue one is lower and has much lower folds variability, so it was chosen as reference for best hyperparameter configuration of ABMIL.

For ABMIL model it was also performed a hyperparameter sweep on the number of extracted frames for each video, in order to analyze if the number of instances in a bag can affect the final performance. Considering a high number of extracted frames, implies a high difference in bag dimensions, because the ones that contains videos will have a very high number of items, whereas the ones with only images still have a maximum of four instances per bag. The result of this search is that the configuration that brings highest F1-score is the one where 16 frames are extracted for each video. The second best performance is the one with 128 frames, whereas the ones with 8, 32 and 64 frames have all slightly lower and similar performance. Therefore there is no evident correlation between the number of frames and the final performance on classification. Nevertheless, when choosing a fixed number of extracted frames, this parameter was set to 16.

4.1.3 DSMIL Results

The Dual-Stream Multiple Instance Learning (DSMIL) model performs both instance-level and bag-level classification through two separate streams. One stream identifies the most relevant instance within each bag, referred to as the critical instance. The second stream computes the similarity between all instances and the critical one, and uses this information to assign attention weights and aggregate the instances into a single bag representation. Compared to ABMIL, DSMIL introduces an explicit mechanism for identifying the most informative

instance within each bag and computes instance correlations with that one as similarity measures.

The main architecture-specific hyperparameters of this model are the attention dimension and the optional application of nonlinear activation functions to the vectors $\mathbf{q}$ and $\mathbf{v}$. Additionally, when a nonlinear activation is applied to the information vector $\mathbf{v}$, a dropout layer is introduced, with the dropout rate acting as an additional tunable hyperparameter.

The hyperparameter search configuration is reported in Table 3.5. Similarly to the procedure adopted for ABMIL, the hyperparameter optimization is performed with a first sweep over the architecture-related parameters, keeping the training-related parameters fixed. After identifying the best architectural configuration, a second sweep is performed on the remaining training-related hyperparameters.

Figure 4.1.3 reports the evolution of the F1-score and validation loss for two representative configurations corresponding to the cases of enabling or disabling nonlinear activation function for the vector $\mathbf{v}$, while for vector $\mathbf{q}$ the nonlinearity is kept enabled. The remaining architecture-related parameters are fixed to the values that yielded the best performance during the sweep, namely an attention dimension of 256 and a dropout rate equal to 0.5 when nonlinear activation is applied to $\mathbf{v}$.

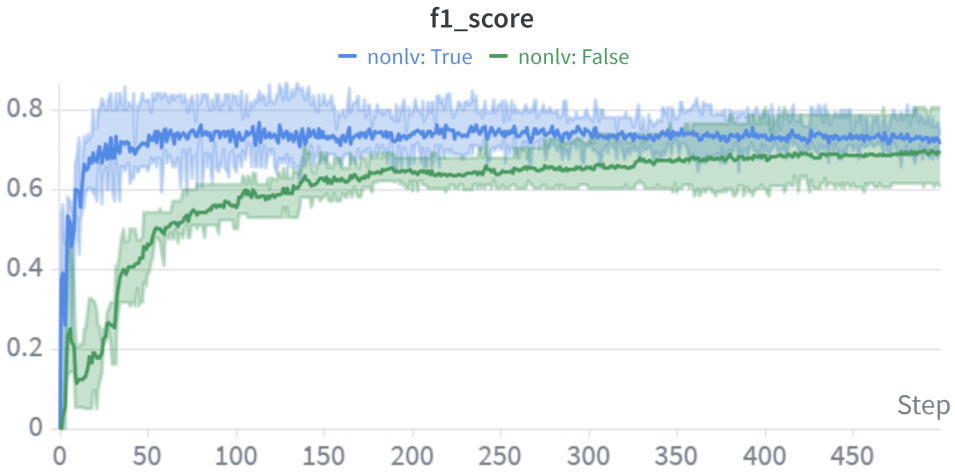
Observations on DSMIL results

From Figure 4.5b one can observe that when the nonlinear activation is enabled for information vector $\mathbf{v}$, corresponding to the blue curve, the validation loss quickly reaches a minimum value around epoch 60, after which the model begins to overfit. In the same configuration, the F1-score increases rapidly and stabilizes around values close to 0.7, reaching a maximum value of 0.77 at epoch 113.

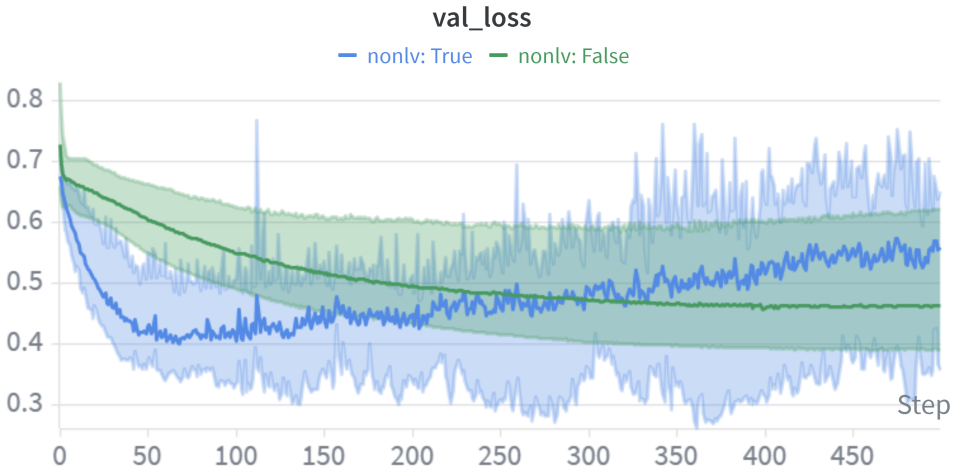
An interesting observation is that the configuration with nonlinear activation on $\mathbf{v}$ exhibit stronger fluctuations in both F1-score and validation loss across consecutive epochs. When the nonlinear activation on $\mathbf{v}$ is disabled instead, both metrics show smoother and more stable training curves. This difference is particularly visible in the validation loss behavior.

For the case with `nonlinear v = False`, the validation loss does not show clear signs of overfitting within the first 500 training epochs, suggesting that the model may still be improving. For this reason, additional simulations were performed extending the training to 1500 epochs. In these longer runs, the validation loss eventually reaches a minimum and then begins to increase, while the F1-score continues to improve until approximately epoch 536. Nevertheless, the maximum F1-score obtained in this setting remains lower than the one achieved when nonlinear activation on $\mathbf{v}$ is enabled in the blue curve.

Additional experiments were also conducted using lower learning rates ($5 \cdot 10^{-5}$



(a) DSMIL F1-score



(b) DSMIL Validation Loss

Figure 4.3: Examples of F1-score and validation loss of DSMIL model

and 10^{-5}), although the corresponding plots are not reported here. In these cases, the training process progresses more slowly, with the validation loss reaching its minimum around epoch 501 for the learning rate 10^{-5}. However, this slower convergence does not lead to significant improvements in F1-score, whose maximum value remains around 0.74, still below the best configuration discussed above.

The strong performance observed for the best DSMIL configuration may be related to the architectural design of the model. By explicitly identifying a critical instance and measuring the similarity of all other instances with respect to it, DSMIL is able to emphasize the most informative regions of the bag while still incorporating contextual information from the remaining instances. This mechanism may be particularly beneficial in ultrasound data, where only a subset of frames may contain diagnostically relevant information. The hyperparameters corresponding to the best-performing configuration and the related evaluation metrics are summarized in Table 4.3.

Hyperparameter	**Value**
Batch size	64
Number of frames	16
Learning rate	10^{-4}
Attention dimension	256
Nonlinear q	True
Nonlinear v	True
Dropout rate	0.5

Metric (epoch 113)	**Value**
F1-score	0.77
Val Accuracy	0.75
Train Accuracy	0.91
Recall	0.80
Precision	0.74
Specificity	0.79
AUC	0.84
Validation loss	0.44
Training loss	0.23

Table 4.3: Summary of the best DSMIL hyperparameter configuration and corresponding performance.

4.1.4 TransMIL Results

The transformer-based MIL (TransMIL) model assumes that instances within a bag are not independent and leverages a self-attention mechanism to capture pairwise correlations among them. In contrast to DSMIL, which focuses on the relationship between each instance and the single critical one, TransMIL models the interactions between all pairs of instances. This allows the model to learn more complex relationships within the bag before performing classification at the bag level. Compared to the previous MIL architectures, TransMIL introduces a transformer-based design that explicitly models global relationships among instances through self-attention.

The architecture of TransMIL is composed of transformer layers that apply the self-attention mechanism, together with a positional encoding module that provides information about the relative position of instances within the bag. This positional information is useful for preserving structural relationships among instances during the attention computation.

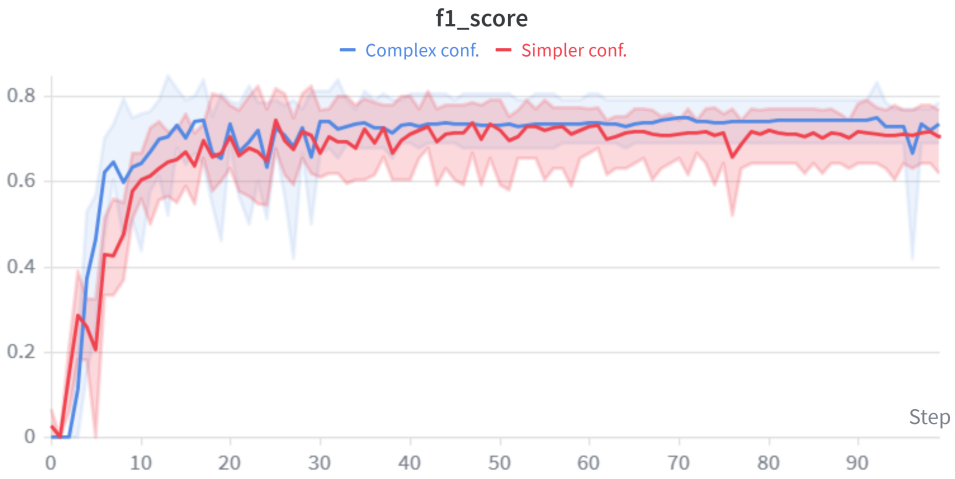
The architecture-specific hyperparameters of TransMIL include the attention dimension, the number of transformer layers and the number of attention heads in each layer. Additional parameters are related to the optional use of a multi-layer perceptron (MLP) applied at the end of each transformer layer; when this option is enabled, dropout layers are also introduced within the MLP, controlled by a dropout rate hyperparameter. The full set of explored hyperparameters is summarized in Table 3.6.

Figure 4.1.4 reports two representative configurations illustrating the evolution of the validation loss and F1-score during training. The blue curve corresponds to a more complex configuration containing a significantly larger number of trainable parameters. In this setting, the attention dimension is set to 512, two transformer layers are used, each with eight attention heads, but the MLP module is disabled. The red curve corresponds instead to a simpler configuration with fewer trainable parameters. In this case, the attention dimension is reduced to 64, the number of attention heads is set to two, and the MLP module is enabled with a dropout rate of 0.1. Additional differences concern the training setup: the complex configuration uses a batch size of 64 and a learning rate of 10^{-4}, whereas the simpler configuration uses a batch size of 32 and a learning rate of $5 \cdot 10^{-5}$.

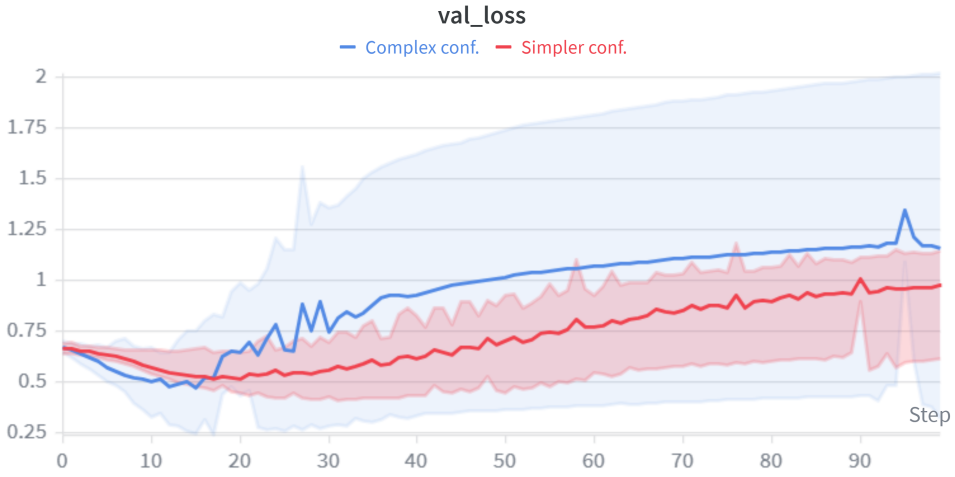
Observations on TransMIL results

The maximum F1-score achieved by both configurations is approximately 0.75, reached at epoch 17 for the complex configuration and at epoch 25 for the simpler one. Although the achieved F1-scores are similar, the validation loss exhibits markedly different behaviors in the two cases. The simpler configuration shows a smoother validation loss curve, reaching a minimum around epoch 20 and then increasing gradually with relatively low variability across folds. In contrast, the more complex configuration reaches its minimum validation loss earlier, around epoch 15, after which the loss increases rapidly and shows significantly larger variability across folds. Interestingly, despite the large increase in validation loss in the complex configuration, the corresponding F1-score remains relatively stable with limited variability across folds. This discrepancy between validation loss and F1-score may be related to the higher capacity of the model, which could make the optimization process more sensitive to small variations in the data.

Additional experiments were conducted using different learning rates and attention dimensions, but no significant improvements in performance were observed.



(a) TransMIL F1-score



(b) TransMIL Validation Loss

Figure 4.4: Examples of F1-score and validation loss of TransMIL model

Given that both configurations achieve similar F1-score values, the simpler configuration is selected as the reference configuration for TransMIL. Besides providing comparable classification performance, it also exhibits a more stable and interpretable validation loss behavior, which suggests better generalization properties. The hyperparameters corresponding to this configuration and the related evaluation metrics are summarized in Table 4.4.

Hyperparameter	Value
Batch size	32
Number of frames	16
Learning rate	$5 \cdot 10^{-5}$
Attention dimension	64
# att. heads	2
# trans. layers	2
Use MLP	True
Dropout rate	0.1

Metric (epoch 25)	Value
F1-score	0.75
Val Accuracy	0.76
Train Accuracy	0.87
Recall	0.81
Precision	0.69
Specificity	0.73
AUC	0.80
Validation loss	0.53
Training loss	0.30

Table 4.4: Summary of the best TransMIL hyperparameter configuration and corresponding performance.

4.1.5 GNN-based MIL Results

Several Graph Neural Network (GNN)-based architectures were implemented to explore an alternative approach for modeling relationships among instances within a bag. In this framework, each bag is represented as a graph in which nodes correspond to instances and edges represent relationships between them. The goal of these models is to capture correlations between instances by explicitly modeling them through graph connections.

Three different models were implemented using the same graph convolutional operator based on the GraphSAGE layer, which updates node representations by aggregating information from neighboring nodes. The three architectures, referred to as *Simple GNN*, *Top-k GNN*, and *Cluster-based GNN*, differ mainly in the strategy used to aggregate node embeddings into a bag-level representation and progressively increase in architectural complexity.

In the *Simple GNN* model, the node embeddings produced by the GraphSAGE layers are directly aggregated into a bag representation using either mean pooling, max pooling, or an attention layer equal to the one used in ABMIL. The *Top-k GNN* model introduces an additional node-selection step, where nodes are ranked according to a learned importance score and only the top-k fraction of nodes

is retained before performing the final aggregation. This mechanism allows the model to focus on the most informative instances within the bag. Finally, the *Cluster-based GNN* model employs a differentiable pooling mechanism that learns an assignment matrix mapping the original nodes to a fixed number of clusters. This operation collapses the graph into a smaller set of super-nodes, allowing the model to learn hierarchical representations of the bag.

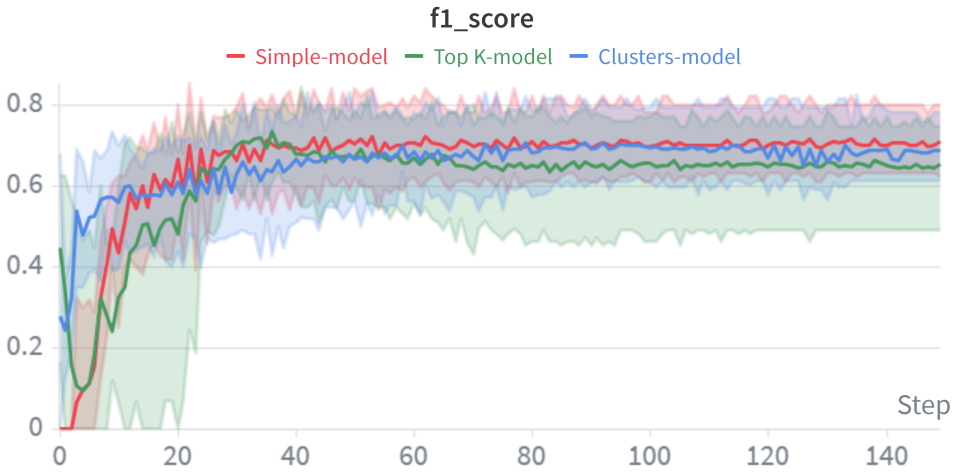
The explored hyperparameters for these models are summarized in Table 3.7. The most relevant ones include the number k of nearest neighbors used to construct the graph, the number of GraphSAGE layers, the aggregation method used in the first two models, and the number of clusters used in the cluster-based model. Figure 4.1.5 reports the evolution of F1-score and validation loss for the best configuration of each of the three GNN-based architectures. For visualization purposes, the variability of the validation loss of the cluster-based model has been partially truncated in the plot, in order to allow a clearer comparison with the other models.

Observations on GNN-based models results

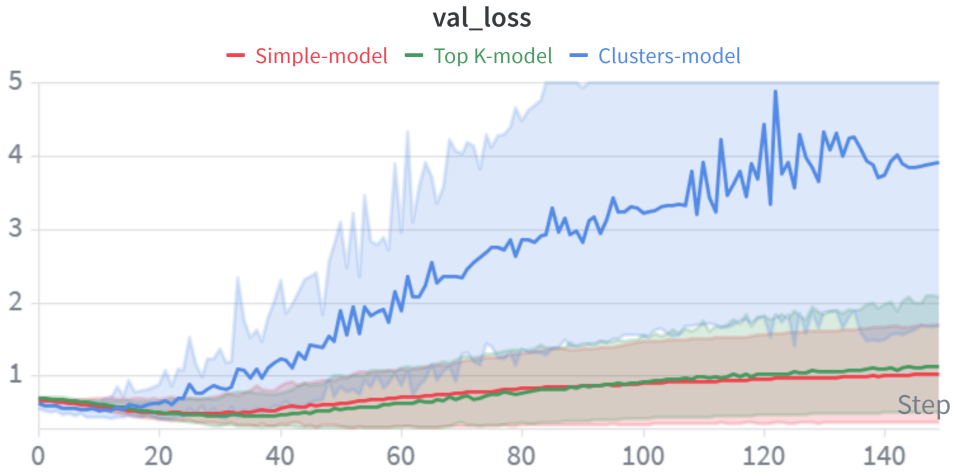
From the plots, the most evident behavior is the significantly larger validation loss and fold variability observed for the cluster-based model. The loss increases more rapidly and reaches higher values compared to the other two architectures. However, this behavior does not correspond to particularly poor F1-score performance, which remains comparable to that of the simple GNN model and shows moderate variability across folds.

The *Simple GNN* and *Top-k GNN* models exhibit more stable validation loss curves. Regarding the F1-score evolution, the simple model reaches its maximum value earlier during training and then stabilizes around values close to 0.70. The Top-k model instead achieves a slightly higher peak value of approximately 0.73, although after reaching this maximum the F1-score decreases slightly and stabilizes around 0.65. The cluster-based model exhibits an intermediate behavior, with F1-score values generally lying between those observed for the simple and Top-k models.

Among the three architectures, the highest F1-score is obtained by the *Top-k GNN* model. For this reason, this configuration is selected as the reference GNN-based model. The corresponding hyperparameters and performance metrics are reported in Table 4.5. The slightly better performance of the Top-k model may be related to its ability to filter out less informative instances before performing the final aggregation. This mechanism may be beneficial in ultrasound data, where only a subset of frames or images may contain relevant diagnostic information.



(a) GNN-based F1-score



(b) GNN-based Validation Loss

Figure 4.5: Examples of F1-score and validation loss of GNN-based models

Top-k GNN-based model	
Hyperparameter	**Value**
Batch size	64
Number of frames	16
Learning rate	$5 \cdot 10^{-5}$
k (nearest n.)	8
Embedding dimension	128
# GSAGE layers	1
Top-k ratio	0.4
Aggrg. method	mean

Metric (epoch 36)	**Value**
F1-score	0.73
Val Accuracy	0.78
Train Accuracy	0.94
Recall	0.73
Precision	0.74
Specificity	0.81
AUC	0.82
Validation loss	0.46
Training loss	0.22

Table 4.5: Summary of the best GNN-based model hyperparameter configuration and corresponding performance.

4.2 Comparison of Model Performances

After having analyzed the performance of each architecture individually, this section provides a comparative analysis across all models in order to identify the most effective approach for the considered task. Table 4.6 summarizes the performance metrics obtained by each model at the epoch corresponding to the highest F1-score. Values in bold indicate the best result achieved for each metric among the considered architectures.

Metric	**Baseline**	**ABMIL**	**DSMIL**	**TransMIL**	**Top-k GNN**
Epoch	2	478	113	25	36
F1-score	0.63	0.72	**0.77**	0.75	0.73
Val. Accuracy	0.75	0.76	0.75	0.76	**0.78**
Recall	0.64	0.70	0.80	**0.81**	0.73
Precision	0.65	**0.74**	**0.74**	0.69	**0.74**
Specificity	0.80	0.80	0.79	0.73	**0.81**
AUC	0.80	0.82	**0.84**	0.80	0.82
Train Accuracy	0.93	0.95	0.91	0.87	0.94
Val. loss	0.64	0.53	0.44	0.53	0.46
Train. loss	0.19	0.12	0.23	0.30	0.22

Table 4.6: Comparison of the best-performing configurations for all considered models. Each value corresponds to the metrics obtained at the epoch achieving the highest F1-score.

Considering first the comparison based on the F1-score, which represents the

primary evaluation metric for this project, it can be observed that all Multiple Instance Learning (MIL) approaches outperform the baseline CNN model. The improvement ranges from approximately 14% for ABMIL to 22% for DSMIL. This result confirms the importance of adopting learning paradigms that are consistent with the structure of the dataset. The baseline model processes individual images independently and assigns a label to each instance, even though instance-level labels are not available in the dataset. In contrast, MIL-based architectures are specifically designed to aggregate information from multiple instances belonging to the same bag and perform classification using only bag-level labels. This paradigm is particularly suitable for the considered dataset, where each clinical case contains as items a collection of images and frames extracted from ultrasound videos.

Among the evaluated models, DSMIL achieves the highest F1-score as well as the best AUC value. The dual-stream architecture allows the model to identify the most informative instance within the bag and to measure the similarity of all other instances with respect to this critical element. In this way, the model captures meaningful correlations between instances while maintaining a moderate level of architectural complexity. Compared to ABMIL, DSMIL relaxes the assumption of independence among instances and explicitly models relationships between them. At the same time, its complexity remains lower than that of transformer-based architectures such as TransMIL, which compute global pairwise correlations between all instances in the bag. In the considered dataset, where bags may contain both independent images and temporally correlated frames extracted from videos, this intermediate level of complexity appears to provide a good balance between representational power and model stability.

TransMIL and the Top-k GNN model also achieve strong performance, clearly outperforming the baseline model. However, the higher validation loss values and the larger fold variability observed during training suggest that these architectures may present lower robustness and potentially reduced generalization capability. Their increased architectural complexity may lead to a larger number of trainable parameters and a higher risk of overfitting when applied to relatively limited datasets.

Another aspect worth highlighting concerns the heterogeneous composition of the bags in the dataset. Each bag may contain both independent images and frames extracted from videos, which may exhibit stronger correlations due to temporal continuity. Additionally, bags may have very different sizes, as many of them do not even contain videos, implying that their dimension could be limited to few images, even a single one. This mixture of instance types and sizes may influence the effectiveness of different architectures, favoring models capable of capturing relevant correlations without relying excessively on global pairwise interactions.

Figure 4.6 provides a visual summary of the F1-score obtained by the best configuration of each model.

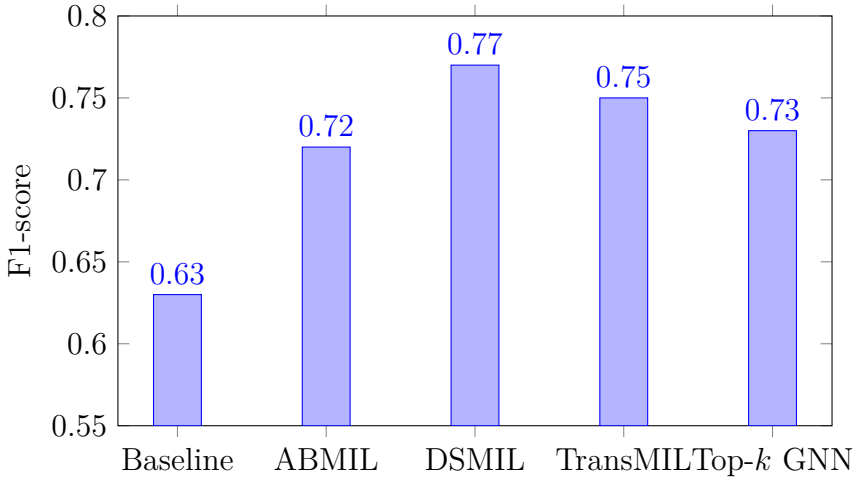


Figure 4.6: Comparison of the F1-score achieved by the best configuration of each model architecture.

As shown in the figure, all MIL-based models consistently outperform the baseline approach, highlighting the importance of apply models that follow the same structure as the available dataset.

It is important to note that the evaluation performed in this study is based on a stratified k-fold cross-validation procedure. Although this approach allows for a reliable comparison between models, the reported performance still depends on validation sets that are indirectly involved in the training and in the model selection process. Consequently, the generalization capability of the models cannot be fully assessed without the use of an independent external test set. Future evaluations should therefore include an additional dataset that is never observed during the training or hyperparameter optimization stages. Such an external evaluation would allow for a more reliable assessment of the generalization performance of the proposed architectures.

In the next chapter, the conclusions of the work are presented. The main results of the study are summarized, the limitations of the current approach are discussed, and possible directions for future research are outlined.

Conclusions

This thesis addressed the problem of developing automatic models for the analysis of ovarian tumors from ultrasound datasets, focusing on a realistic clinical scenario in which multiple ultrasound images and videos are acquired for each patient, while annotations are available only at the clinical case level. To address this problem, several models based on the Multiple Instance Learning (MIL) paradigm were investigated. In MIL, data are organized into groups of instances, called bags, and labels are available only at the bag level rather than for individual instances. This framework naturally reflects the clinical workflow of ultrasound examinations: multiple images or frames are collected for a single patient, and a unique diagnosis is provided based on the overall assessment of all available information. In the context of ovarian tumors, if at least one image reveals characteristics of malignancy, the tumor is classified as malignant; otherwise it is considered benign.

In this work, four MIL-based models were implemented and compared with a supervised baseline model. The baseline approach performed instance-level classification by propagating the bag label to each instance, ignoring the weakly supervised nature of the dataset. In contrast, the MIL models explicitly addressed the bag-level learning problem. The architectures considered in this study were Attention-Based MIL (ABMIL), Dual-Stream MIL (DSMIL), Transformer-based MIL (TransMIL), and a Graph Neural Network (GNN)-based MIL model.

A common convolutional neural network feature extractor, based on DenseNet121 pretrained on ImageNet, was used to generate instance-level embeddings from the ultrasound images. These embeddings were then aggregated by the different MIL architectures in order to obtain a single bag-level representation on which the final binary classification was performed. Model selection and hyperparameter tuning were performed using a stratified k-fold cross-validation strategy on the training and validation sets. Several evaluation metrics were considered, while the primary metric used for comparison was the F1-score, which is particularly suitable for imbalanced datasets such as the one considered in this work, where malignant cases are less frequent than benign ones.

The experimental results demonstrate that Multiple Instance Learning models consistently outperform the supervised baseline in the task of bag-level classification.

In particular, MIL-based architectures achieved significantly higher F1-score values, confirming that explicitly modeling the bag structure of the data leads to improved performance in weakly supervised settings. Among the tested models, DSMIL obtained the best overall performance, achieving the highest F1-score as well as the highest Area Under the Curve and Precision values.

One possible explanation for this result lies in the architectural characteristics of DSMIL. Unlike simpler attention-based approaches, which treat instances as independent, DSMIL explicitly models relationships among instances by focusing on the most informative one. At the same time, its architecture remains less complex than Transformer-based models such as TransMIL, which attempt to learn all pairwise interactions among instances and therefore require a significantly larger number of trainable parameters. The results obtained in this work suggest that modeling instance interactions is beneficial for this task, but that an intermediate level of architectural complexity may provide the best balance between expressive power and generalization capability.

The main contributions of this thesis can be summarized as follows:

- Application of Multiple Instance Learning models to the problem of ovarian tumor diagnosis from ultrasound imaging data;
- Implementation and comparison of several MIL architectures within a unified experimental framework;
- Empirical demonstration that MIL-based approaches outperform a conventional supervised baseline when only case-level labels are available;
- Analysis of the role of instance aggregation strategies and instance interactions in improving bag-level classification performance.

To the best of our knowledge, MIL-based approaches have not been widely explored in the context of ovarian tumor diagnosis from ultrasound data. Therefore, this work contributes to bridging the gap between deep learning methodologies and realistic clinical data acquisition settings.

Despite the promising results obtained in this study, several limitations should be noticed. First, the absence of a dedicated external test set prevents a full assessment of the generalization capability of the models. Although cross-validation provides a reliable estimate of performance, the evaluation on an independent dataset would allow a more robust validation of the proposed approaches. Second, the preprocessing pipeline applied to the ultrasound images was relatively simple. In particular, the procedures used to remove annotations and black regions outside the fan-beam area were limited, and more advanced preprocessing techniques could further improve the quality of the input data.

Future work could explore several directions to further improve the results obtained in this thesis. One possible extension concerns the preprocessing stage, where more sophisticated methods for isolating the relevant ultrasound region and removing artifacts could be developed. Additionally, the current study focused on a binary classification problem between benign and malignant tumors. However, ovarian tumors can also belong to a third category, known as borderline tumors, which were not included in the dataset used in this work. Future studies could therefore investigate multi-class classification problems including borderline cases and other diagnostically challenging tumor types in order to evaluate the robustness of MIL-based models in more complex scenarios. Another interesting option for future works consists in observing also the attention scores produced by the MIL-based models. If a dataset with also item-level annotations is available, it would be interesting to compare them with the attention scores to assess the reliability of the models in detecting the most informative instances.

Another important direction for future research concerns the feature extraction stage. In this work, the DenseNet121 feature extractor was pretrained on ImageNet and kept frozen during training, while the MIL models operated on the extracted embeddings. Although this strategy significantly reduced computational cost and training time, it prevented the feature extractor from adapting specifically to the ultrasound domain. A potential improvement would consist in training the feature extractor jointly with the MIL aggregation module in an end-to-end manner. While this approach would increase the number of trainable parameters and computational requirements, it could also enable the model to learn feature representations better suited to the characteristics of ultrasound images.

Overall, the results presented in this thesis demonstrate that Multiple Instance Learning represents a promising framework for addressing medical imaging problems in which detailed annotations are not available and data are naturally organized at the case level. By aligning machine learning methodologies with the structure of real clinical data, MIL-based approaches have the potential to support the development of more effective and practical decision-support tools for the diagnosis of ovarian tumors from ultrasound examinations.

Bibliography

[1] D Fischerova. «Ultrasound scanning of the pelvis and abdomen for staging of gynecological tumors: a review». In: *Ultrasound in obstetrics & gynecology* 38.3 (2011), pp. 246–266 (cit. on p. 1).

[2] Sian Mitchell, Manolis Nikolopoulos, Alaa El-Zarka, Dhurgham Al-Karawi, Shakir Al-Zaidi, Avi Ghai, Jonathan E Gaughran, and Ahmad Sayasneh. «Artificial intelligence in ultrasound diagnoses of ovarian cancer: a systematic review and meta-analysis». In: *Cancers* 16.2 (2024), p. 422 (cit. on pp. 1, 5, 18).

[3] World Cancer Research Fund. *Ovarian cancer statistics worldwide*. `https://www.wcrf.org/preventing-cancer/cancer-statistics/ovarian-cancer-statistics/`. Accessed: 15 Jan 2026. 2022 (cit. on p. 3).

[4] SEER National Cancer Institute. *Ovarian cancer statistics in USA*. `https://seer.cancer.gov/statfacts/html/ovary.html`. Accessed: 30 Jan 2026. 2021 (cit. on p. 3).

[5] EMJ Meys, J Kaijser, RFPM Kruitwagen, BFM Slangen, Ben Van Calster, Bert Aertgeerts, JY Verbakel, Dirk Timmerman, and T Van Gorp. «Subjective assessment versus ultrasound models to diagnose ovarian cancer: A systematic review and meta-analysis». In: *European Journal of Cancer* 58 (2016), pp. 17–29 (cit. on p. 4).

[6] Dirk Timmerman, Lil Valentin, TH Bourne, WP Collins, Herman Verrelst, and Ignace Vergote. «Terms, definitions and measurements to describe the sonographic features of adnexal tumors: a consensus opinion from the International Ovarian Tumor Analysis (IOTA) Group». In: *Ultrasound in Obstetrics and Gynecology: The Official Journal of the International Society of Ultrasound in Obstetrics and Gynecology* 16.5 (2000), pp. 500–505 (cit. on p. 4).

[7] Gwenolé Quellec, Guy Cazuguel, Béatrice Cochener, and Mathieu Lamard. «Multiple-instance learning for medical image and video analysis». In: *IEEE reviews in biomedical engineering* 10 (2017), pp. 213–234 (cit. on p. 6).

[8] Frank Rosenblatt. «The perceptron: A probabilistic model for information storage and organization in the brain». In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519 (cit. on p. 11).

[9] Muhammad Waqas, Syed Umaid Ahmed, Muhammad Atif Tahir, Jia Wu, and Rizwan Qureshi. «Exploring multiple instance learning (MIL): A brief survey». In: *Expert Systems with Applications* 250 (2024), p. 123893 (cit. on pp. 15, 18, 22).

[10] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. «Solving the multiple instance problem with axis-parallel rectangles». In: *Artificial intelligence* 89.1-2 (1997), pp. 31–71 (cit. on p. 14).

[11] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. «Multiple instance learning: A survey of problem characteristics and applications». In: *Pattern recognition* 77 (2018), pp. 329–353 (cit. on p. 15).

[12] Samman Fatima, Sikandar Ali, and Hee-Cheol Kim. «A comprehensive review on multiple instance learning». In: *Electronics* 12.20 (2023), p. 4323 (cit. on p. 16).

[13] F Christiansen, EL Epstein, E Smedberg, M Åkerlund, Kevin Smith, and E Epstein. «Ultrasound image analysis using deep neural networks for discriminating between benign and malignant ovarian tumors: comparison with expert subjective assessment». In: *Ultrasound in Obstetrics & Gynecology* 57.1 (2021), pp. 155–163 (cit. on p. 19).

[14] Yue Gao et al. «Deep learning-enabled pelvic ultrasound images for accurate diagnosis of ovarian cancer in China: a retrospective, multicentre, diagnostic study». In: *The Lancet Digital Health* 4.3 (2022), e179–e187 (cit. on p. 19).

[15] Huiquan Wang et al. «Application of deep convolutional neural networks for discriminating benign, borderline, and malignant serous ovarian tumors from ultrasound images». In: *Frontiers in oncology* 11 (2021), p. 770683 (cit. on p. 19).

[16] Yuyeon Jung, Taewan Kim, Mi-Ryung Han, Sejin Kim, Geunyoung Kim, Seungchul Lee, and Youn Jin Choi. «Ovarian tumor diagnosis using deep convolutional neural networks and a denoising convolutional autoencoder». In: *Scientific Reports* 12.1 (2022), p. 17024 (cit. on p. 19).

[17] Jun Wang, Yu Mao, Nan Guan, and Chun Jason Xue. «Advances in multiple instance learning for whole slide image analysis: Techniques, challenges, and future directions». In: *arXiv preprint arXiv:2408.09476* (2024) (cit. on p. 20).

[18] Maximilian Ilse, Jakub Tomczak, and Max Welling. «Attention-based deep multiple instance learning». In: *International conference on machine learning*. PMLR. 2018, pp. 2127–2136 (cit. on pp. 20, 47).

[19] Bin Li, Yin Li, and Kevin W Eliceiri. «Dual-stream multiple instance learning network for whole slide image classification with self-supervised contrastive learning». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 14318–14328 (cit. on pp. 21, 23, 48, 49).

[20] Zhuchen Shao, Hao Bian, Yang Chen, Yifeng Wang, Jian Zhang, Xiangyang Ji, et al. «Transmil: Transformer based correlated multiple instance learning for whole slide image classification». In: *Advances in neural information processing systems* 34 (2021), pp. 2136–2147 (cit. on pp. 24, 25, 49).

[21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. «A comprehensive survey on graph neural networks». In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24 (cit. on p. 27).

[22] Ming Tu, Jing Huang, Xiaodong He, and Bowen Zhou. «Multiple instance learning with graph neural networks». In: *arXiv preprint arXiv:1906.04881* (2019) (cit. on pp. 27, 29, 52, 53).

[23] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. «Hierarchical graph representation learning with differentiable pooling». In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 28).

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. «Inductive representation learning on large graphs». In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 29, 52).

[25] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 36, 37).

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. «Densely connected convolutional networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708 (cit. on p. 44).

[27] Francisco M Castro-Macıas, Francisco J Sáez-Maldonado, Pablo Morales-Álvarez, and Rafael Molina. «torchmil: A PyTorch-based library for deep Multiple Instance Learning». In: *arXiv preprint arXiv:2509.08129* (2025) (cit. on p. 54).

[28] Matthias Fey and Jan E. Lenssen. «Fast Graph Representation Learning with PyTorch Geometric». In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019 (cit. on p. 54).