Università degli Studi di Milano-Bicocca

School of Science

Department of Physics

Master Degree in Particle Physics

Overcoming medical data scarcity: transfer learning from synthetic particle jets images to lung CT using deep neural networks

Supervisor:
Prof. Pietro Govoni
Co-Supervisor:
Dott. Simone Gennai

Candidate:
Beatrice Scotti
Student ID: 858535

September 19th, 2025

Academic Year 2024-2025

Contents

Abstract

Early diagnosis is the most effective weapon in the fight against cancer, with computed tomography and medical imaging playing a crucial role in this process. Recently, artificial intelligence has shown great potential to identify and classify tumor lesions in their early stages. However, the application of these techniques faces a fundamental limitation: the scarcity of annotated data available for training.

As is well known in machine learning, the quality of results depends strictly on the quality and completeness of the training data ("garbage in, garbage out"). To overcome this limitation, this thesis explores the use of transfer learning, a technique that allows the transfer of knowledge from a data-rich source domain to a data-limited target domain.

Specifically, the research focuses on the use of simulated images of high-energy particle jets, constructed specifically for this study, characterized by two distinct classes of physical phenomena: a noisy background with physical characteristics similar to the main jets and morphological properties similar to tumor structures.

The proposed approach involves a pretraining phase on a large high-energy physics dataset, transferring the learned weights to the medical domain, and then fine-tuning them on limited clinical datasets. This method aims to leverage the feature extraction capabilities developed in the particle physics domain, adapting them to the medical context where annotated data are scarce, but diagnostic accuracy is crucial.

Introduction

Chapter 1

Computed Tomography

Computed Tomography (CT) is a medical imaging technique that uses X-rays to generate detailed cross-sectional images of the human body. Since its introduction in the 1970s, CT has revolutionized diagnostic radiology by enabling the visualization of internal structures with high spatial resolution and three-dimensional reconstruction capabilities. Unlike conventional radiography, which produces a single projection image, CT acquires multiple X-ray measurements from different angles around the patient and uses computer algorithms to reconstruct the corresponding anatomical slices.

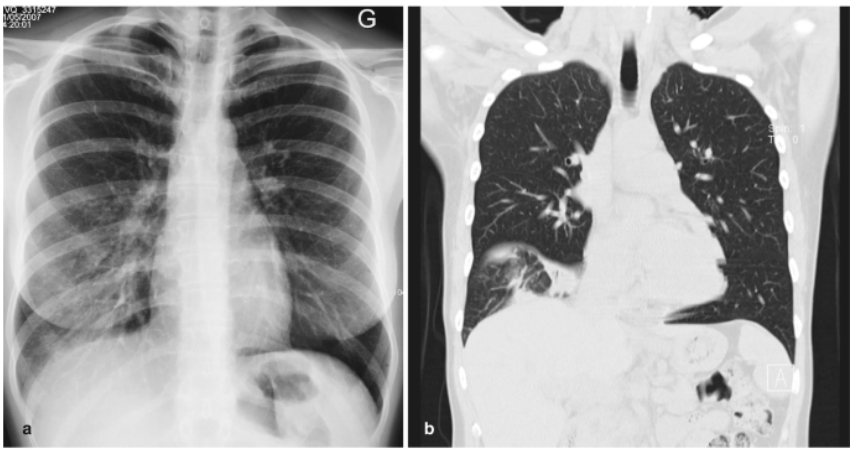


Figure 1.1: On the left (figure a.) we can see a chest radioraphy, on the right (figure b.) we can see a CT scan of the same area. The CT scan provides a much more detailed view of the internal structures, allowing for better diagnosis and treatment planning.

CT plays a crucial role in clinical practice due to its ability to provide fast, accurate, and non-invasive diagnostic information for a wide range of applications. Modern CT scanners are equipped with advanced technologies that significantly improve image quality, reduce scan time, and minimize radiation dose.

This chapter provides an overview of the main aspects of CT imaging that are relevant for medical physics applications. The first part briefly describes the experimental setup of a CT scanner, the second part focuses on the mathematical principles behind image reconstruction, highlighting the algorithms used to convert projection data into tomographic slices. Finally, the chapter introduces the concept of image post-processing, with particular attention to windowing and its impact on image interpretation.

1.1 Tomography setup

The basic idea behind tomography is to obtain a three-dimensional image of an object from a series of two-dimensional images acquired from different angles. For this purpose, a CT scanner consists, like any radiation measurement system, of a radiation source (an X-ray tube) and an array of particle detectors. To acquire projections from multiple angles, typically ranging from 0 to 180 degrees, the X-ray tube and the detectors are mounted on a rotating gantry, which allows the system to rotate around the patient or, more generally, around the object being examined.

As shown in Figure 1.2, the detectors are not arranged on a plane perpendicular to the X-ray source, but rather along a circular arc. This configuration ensures that all pixels in the projection correspond to the same physical size, thus avoiding geometric distortions in the reconstructed image. Depending on the scanner design, the acquisition can be performed in a step-and-shoot mode or using continuous rotation, as in helical (spiral) CT. The X-ray beam is collimated into a fan or cone shape to cover the entire detector array. Each detector element typically consists of a scintillator coupled to a photodiode, which converts the X-ray photons into an electrical signal proportional to the received dose. Synchronizing the gantry rotation with data acquisition allows the collection of a sufficient number of projections to ensure accurate image reconstruction.

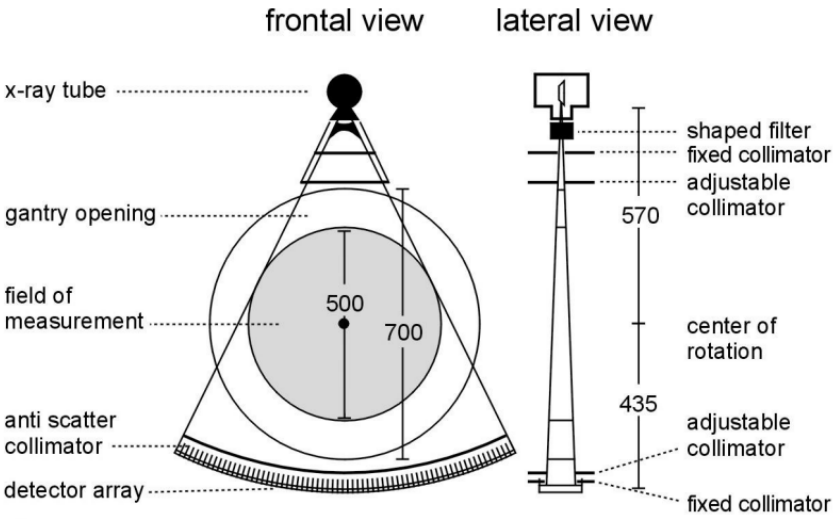


Figure 1.2: In the frontal view we can see, in order, the X-ray tube (radiation source), the rotating gantry (support for the X-ray tube and detectors), the field of measurement (where the patient or the object is allocated), the anti scatter collimators (to reduce the amount of scattered radiation reaching the detectors), and the detectors (which measure the intensity of the X-rays after they pass through the object). In the lateral view we can see the presence of collimators, used to adjust the narrowness of the beam depending on the type of scan and the area of interest.

1.2 Image Reconstruction

In computed tomography (CT), image reconstruction refers to the process of transforming raw projection data, acquired from different angles, into a cross-sectional image of the scanned object. The core idea is to recover a 2D function $f(x, y)$ — representing the

internal structure — from its projections. Each projection contains information about how much an X-ray beam is attenuated as it travels through the object along a certain direction.

1. The Radon Transform

Mathematically, each X-ray beam travels along a straight line. For a beam oriented at angle θ, the line is defined by:

$$x\cos\theta + y\sin\theta = s$$

where s is the perpendicular distance from the origin to the line.

The projection $p(s,\theta)$ is obtained by integrating the object's density $f(x,y)$ along that line:

$$p(s,\theta) = \int_{-\infty}^{+\infty} f(x,y)\,\delta(x\cos\theta + y\sin\theta - s)\,dx\,dy$$

This is called the **Radon transform** of the function $f(x,y)$.

2. The Fourier Slice Theorem

To reconstruct the image, a key result is the **Fourier Slice Theorem**. It says that the 1D Fourier transform of a projection at angle θ gives a slice of the 2D Fourier transform of $f(x,y)$ taken in the same direction.

We compute the 1D Fourier transform of the projection:

$$P(\theta,u) = \int_{-\infty}^{+\infty} p(s,\theta)\,e^{-2\pi i u s}\,ds$$

Plugging in the expression for $p(s,\theta)$, we get:

$$P(\theta,u) = \iint f(x,y)\,e^{-2\pi i u(x\cos\theta + y\sin\theta)}\,dx\,dy$$

This expression matches the 2D Fourier transform of $f(x,y)$, evaluated at:

$$u_x = u\cos\theta, \quad u_y = u\sin\theta$$

so we can write:

$$P(\theta,u) = F(u\cos\theta, u\sin\theta)$$

In other words, each projection gives us a radial slice of the 2D Fourier transform of the image.

3. Inverse Transform and Filtering

If we collect projections over many angles, we can build the 2D Fourier space of the image. Then, to reconstruct $f(x,y)$, we apply the inverse 2D Fourier transform:

$$f(x,y) = \iint F(u_x,u_y)\,e^{2\pi i(u_x x + u_y y)}\,du_x\,du_y$$

Switching to polar coordinates:

$$u_x = u\cos\theta, \quad u_y = u\sin\theta$$

the Jacobian of the transformation gives:

$$du_x\, du_y = |u|\, du\, d\theta$$

Substituting into the inverse Fourier formula:

$$f(x,y) = \int_0^{\pi} \int_{-\infty}^{+\infty} |u|\, P(\theta,u)\, e^{2\pi i u(x\cos\theta + y\sin\theta)}\, du\, d\theta$$

Here, the factor $|u|$ acts as a high-pass filter, enhancing high-frequency components and correcting for the blurring introduced during projection. The equation above gives the foundation for the **Filtered Back Projection** algorithm, one of the most common reconstruction methods in CT. It works as follows:

1. Acquire projections $p(s,\theta)$ at many angles $\theta \in [0,\pi]$.
2. Compute their 1D Fourier transforms $P(\theta,u)$.
3. Multiply by $|u|$ (or an equivalent ramp-like filter) to enhance sharpness.
4. Invert the Fourier transform to get filtered projections.
5. Back-project the filtered projections over the image domain and sum them.

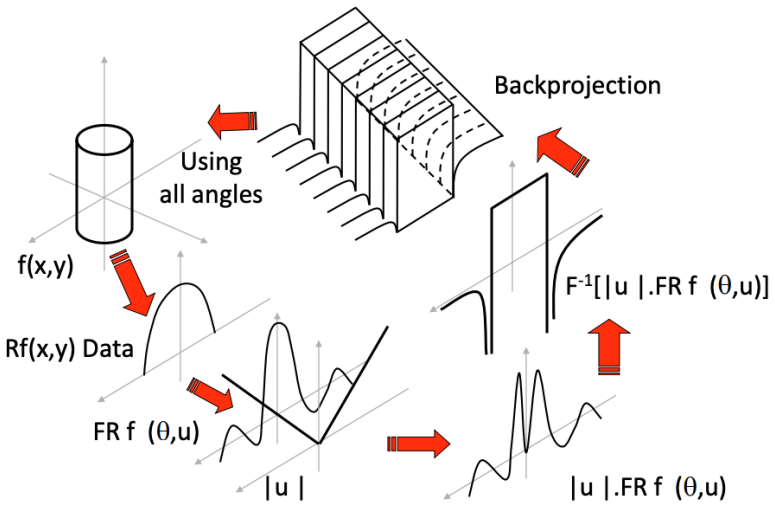


Figure 1.3: Radon transform and reconstruction via filtered back projection. Each projection contributes a slice in frequency space; the image is recovered by inverse Fourier transform.

The filtered back projection is fast and effective, which is why it is still widely used in clinical CT. However, modern scanners often integrate more advanced techniques such as iterative reconstruction or machine learning, which can offer better noise suppression and lower radiation dose. Nevertheless, understanding the Radon transform and the Fourier Slice Theorem provides deep insight into how tomographic images are formed.

1.3 Windowing

The pixel values are quantified in **Hounsfield units (HU)**, a standardized scale that reflects the radiodensity of the tissue. This scale is defined relative to the attenuation coefficients of water and air:

$$HU_{tissue} = 1000 \times \frac{\mu_{tissue} - \mu_{water}}{\mu_{water} - \mu_{air}} \tag{1.1}$$

where μ represents the linear attenuation coefficient. Key reference points: water = 0 HU for definition, air = -1000 HU.

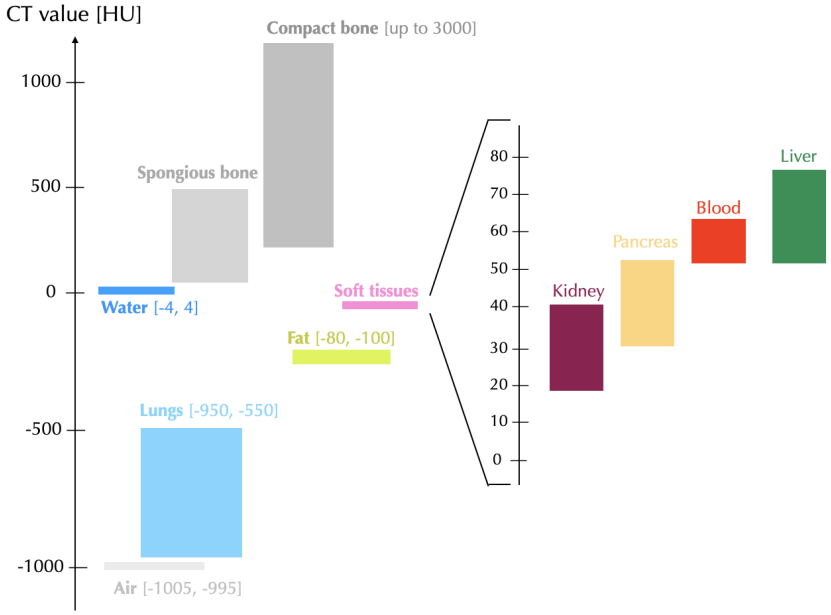


Figure 1.4: CT Hounsfield Unit (HU) ranges for key tissues. Highlighted regions show typical HU values for diagnostic reference, from dense compact bone ($\sim$ 3000 HU) to air (1000 HU), with soft tissues ($20 - 80$ HU) and fluids (blood $\sim$ 45 HU, water 0 HU) in intermediate ranges.

Windowing is a technique used in computed tomography (CT) to **enhance the visibility of specific tissues or structures within the body.** It involves adjusting the range of Hounsfield units (HU) displayed in the CT image, allowing radiologists to focus on particular types of tissues, such as bones, soft tissues, or air-filled spaces. The window width and center determine the contrast and brightness of the image, respectively.

- **Window width** (WW): This parameter controls the range of Hounsfield units displayed in the image. A narrow window width enhances the contrast between tissues with similar densities, while a wider window width allows a broader range of densities to be visualized.
- **Window center** (WC): This parameter sets the midpoint of the Hounsfield unit range displayed in the image. Adjusting the window level changes the brightness of the image, allowing radiologists to focus on specific tissue types.

The choice of window width and level depends on the specific clinical question and the type of tissue being examined.

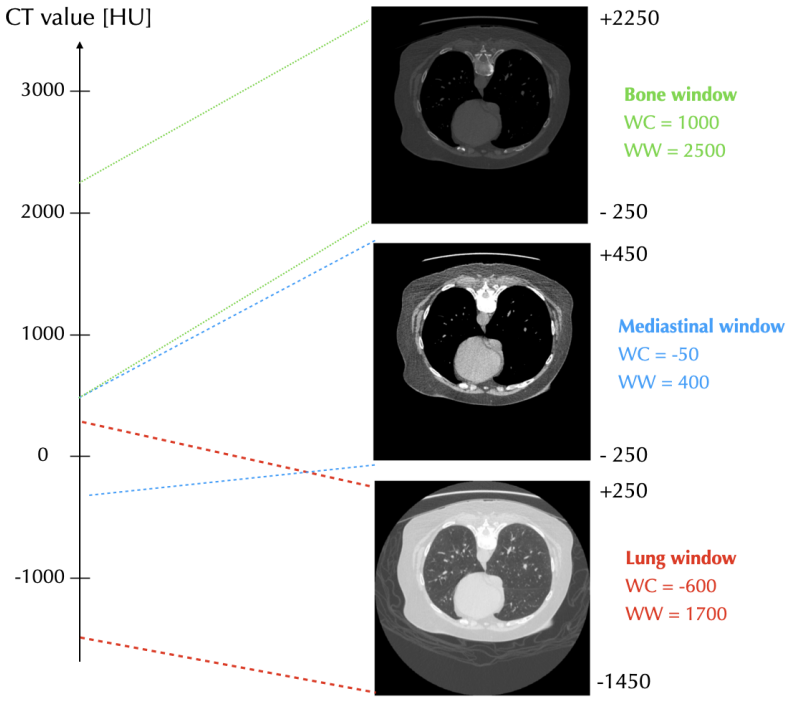


Figure 1.5: Standard CT windowing settings for different anatomical structures. Full CT value range ([-1000, 3000] HU). Clinical presets with window center (WC) and width (WW) values for bone (WC=1000, WW=2500), mediastinum (WC=-50, WW=400), and lung (WC=-600, WW=1700) visualization.

1.4 CT Planes

Images can be visualized in three standard orthogonal planes: axial, coronal, and sagittal. Each plane offers a unique perspective on anatomical structures, aiding in comprehensive clinical assessment.

- **Axial plane** (transverse): A horizontal plane that divides the body into superior (upper) and inferior (lower) parts. It is the primary acquisition plane in most CT scans.
- **Coronal plane**: A vertical plane that divides the body into anterior (front) and posterior (back) parts. Often used for viewing the frontal anatomy.
- **Sagittal plane**: A vertical plane that divides the body into left and right portions. It is particularly useful for evaluating asymmetries between the two sides.

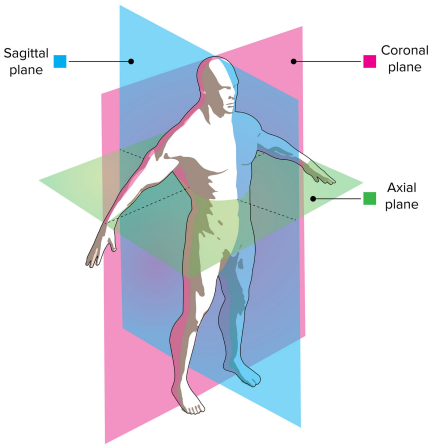


Figure 1.6: CT planes: axial, coronal, and sagittal. The axial plane is horizontal, the coronal plane is vertical and divides the body into front and back, and the sagittal plane is vertical and divides the body into left and right.

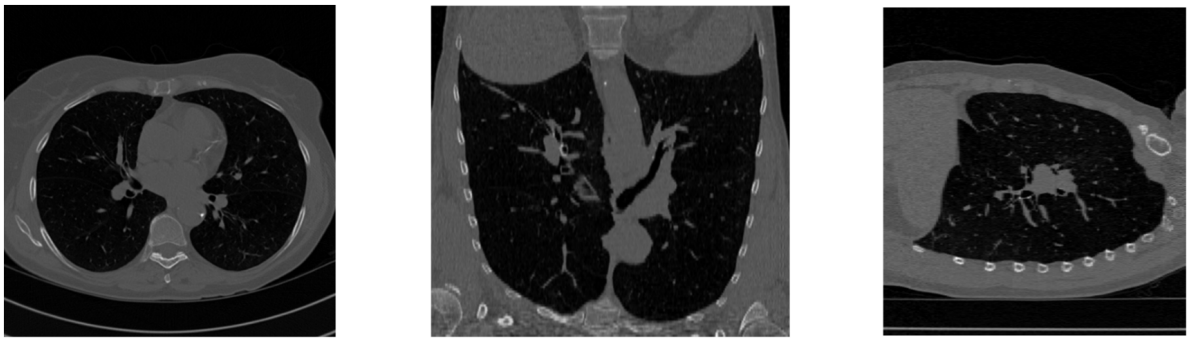


Figure 1.7: CT planes: axial, coronal, and sagittal. The axial plane is horizontal, the coronal plane is vertical and divides the body into front and back, and the sagittal plane is vertical and divides the body into left and right.

Chapter 2

Machine Learning in a nutshell

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead, these systems learn from data, identifying patterns and making decisions based on the information they have been trained on. Machine learning consists of designing efficient and accurate prediction algorithms. More generally, learning techniques are data-drivem methods combining fundamental concepts in computer science with ideas from statistics, probability and optimization.

Types of tasks

The following are some standard types of tasks in machine learning:

- **Classification:** Assigning labels to data points based on learned patterns (e.g., e-mail spam detection).
- **Regression:** Predicting continuous values based on input features (e.g., predicting house prices). In regression, the penalty for an incorrect prediction depends on the magnitude of the difference between the true and predicted values, in contrast with classification problem, where there is typically no notion of closeness between various categories.
- **Object detection:** Identifying and localizing objects within images or videos (e.g., detecting tumours in lung CT scans).
- **Clustering:** Grouping similar data points together without predefined labels.
- **Anomaly detection:** Identifying unusual patterns that do not conform to expected behavior.
- **Ranking**: Ordering items based on their relevance or importance.

Algorithms that solve a learning task based on semantically annotated historical data are said to operate in a **supervised learning** mode. In contrast, algorithms that use data without any semantic annotation are said to operate in an **unsupervised learning** mode. In the latter case, the algorithm is expected to discover patterns in the data without any prior knowledge of the labels or categories. In this thesis I'll mainly focus on supervised learning.

Label set: We use Y to denote the set of all possible labels for a data point of a given learning problem. Note that the labels can be of two different types: categorical

labels, which are discrete and finite and define classification problems, and continuous labels, which can take any value in a continuous range and define regression problems.

2.1 Neural networks and deep learning

Neural networks are a class of machine learning algorithms inspired by the structure and function of the human brain. They consist of interconnected nodes (neurons) that process information in layers. Deep learning refers to the use of neural networks with many layers (deep neural networks) to model complex patterns in large datasets. This approach has led to significant advancements in areas such as image recognition, natural language processing, and game playing.

2.2 YOLO: You Only Look Once [2]

Object detection is a task that involves identifying and classifying objects present in images or videos. Initially, object detection was approached as a pipeline consisting of three main steps: proposal generation, feature extraction and region classification. However, this approach was computationally expensive and often led to suboptimal results. The emergence of deep learning brought a significant change in object detection, with deep convolutional neural networks (CNNs) playing a crucial role in this transformation. CNNs are designed to automatically learn hierarchical features from raw pixel data, eliminating the need for manual feature engineering. This shift allowed for more efficient and accurate object detection systems.

Currently, deep learning-based object detection frameworks can be classified in two families:

- **Two-stage detectors:** These methods first generate region proposals and then classify them. Examples include R-CNNs (Region-based Convolutional Neural Networks), that first generate region proposals using a selective search algorithm and then extracts features from these regions using a CNN; the extracted features are then fed into an SVM for object classification.

- **One-stage detectors:** These methods perform detection in a single pass, directly predicting bounding boxes and class probabilities. Examples include YOLO (You Only Look Once), which exists in eleven versions.The YOLO models are popular for their accuracy and compact size. It is a state-of-the-art model that could be trained on any hardware. YOLOv8, in particular, was developed by Ultralytics and introduced on January 2023. It is used to detect objects in images, classify images and distinguish onjects from each other.

2.2.1 Architecture of YOLOv8

The YOLOv8 architecture is composed of two major parts, namely the **backbone** and the **head**, both of which use a fully convolutional neural network.

- The **backbone** is responsible for extracting features from the input image. It consists of a modified version of the CSPDarknet53 architecture, which has 53 convolutional layers and employs a technique called cross-stage partial connections to

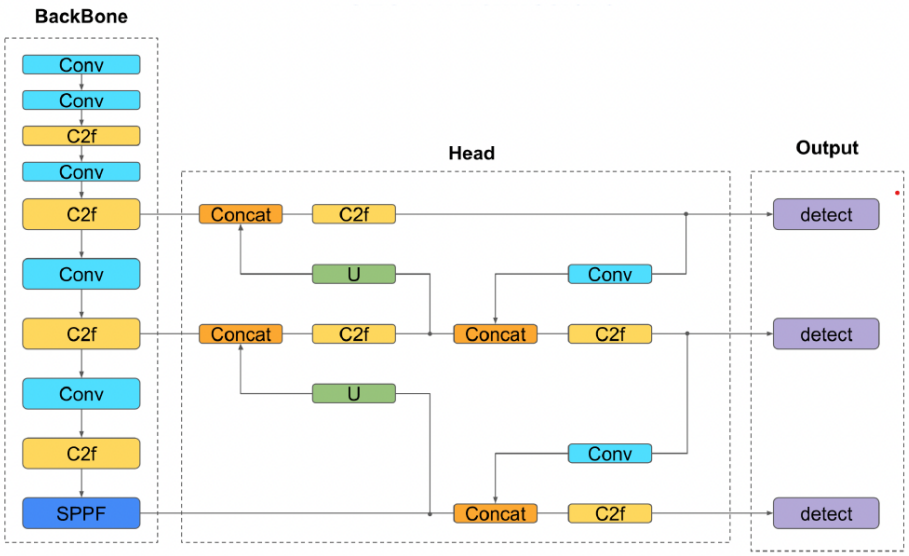


Figure 2.1: Architecture of YOLOv8. The backbone extracts features from the input image, while the head predicts bounding boxes and class probabilities.

enhance the transmission of information across the various levels of the network. The convolutional layers are organized in a sequential manner to extract relevant features from the input image.

- The **head** is responsible for predicting bounding boxes and class probabilities. It consists of a series of convolutional layers that take the features extracted by the backbone and apply additional operations to predict the bounding boxes and class probabilities for each object in the image. The head uses a technique called anchor boxes to handle objects of different sizes and aspect ratios.

The YOLOv8 framework can be used to perform computer vision tasks such as detection, segmentation, classification and pose estimation and comes with pre-trained models for each task. For detection, the models are pre trained on the COCO dataset, while for classification on ImageNet dataset. There are different versions of YOLOv8, each designed for different tasks and with different architectures. The most common versions are YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l and YOLOv8x, where the letter indicates the size of the model (n for nano, s for small, m for medium, l for large and x for extra large). The larger the model, the more parameters it has and the more computational resources it requires to train and run.

2.3 Object Detection Metrics

A metric is a function that quantifies and evaluates the performance of a model. In the context of object detection, metrics are used to assess how well a model can detect and classify objects in images.

Intersection over Union (IoU)

When we talk about object detection, we often refer to the concept of **bounding boxes**. A bounding box is a rectangular box that is used to define the location of an object in an

image, typically represented by its top-left corner coordinates (x, y), width, and height, or top-left and bottom-right coordinates. The bounding box is used to localize the object within the image and is often used in conjunction with a classification label to identify the object. Intersection over Union (IoU) is a metric used to evaluate the accuracy of the bounding box detection. It measures the overlap between the predicted and the ground truth box. The IoU is calculated as follows:

$$IoU = \frac{Area_{Intersection}}{Area_{Union}} = \frac{A_{pred} \cap A_{gt}}{A_{pred} \cup A_{gt}} \tag{2.1}$$

The IoU value ranges from 0 to 1, where 0 indicates no overlap and 1 indicates perfect overlap. A common threshold for considering a detection as a true positive is an IoU of 0.5 or higher.

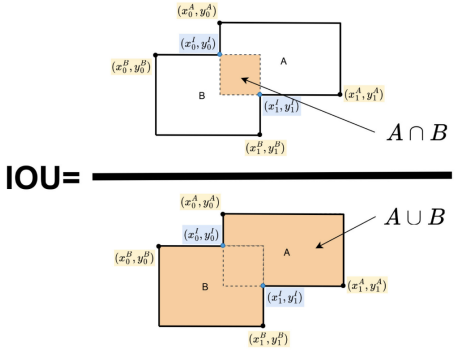


Figure 2.2

Algorithm 1 Intersection over Union (IoU) between two bounding boxes

1: **Input:** Boxes $A = (x_A, y_A, w_A, h_A)$ and $B = (x_B, y_B, w_B, h_B)$
2: **Compute bottom-right corners:**
$x2_A = x_A + w_A, \quad y2_A = y_A + h_A$
$x2_B = x_B + w_B, \quad y2_B = y_B + h_B$
3: **Compute intersection rectangle:**
$x^1_{\text{int}} = \max(x_A, x_B), \quad y^1_{\text{int}} = \max(y_A, y_B)$
$x^2_{\text{int}} = \min(x2_A, x2_B), \quad y^2_{\text{int}} = \min(y2_A, y2_B)$
4: **Compute intersection area:**
$w_{\text{int}} = \max(0, x^2_{\text{int}} - x^1_{\text{int}})$
$h_{\text{int}} = \max(0, y^2_{\text{int}} - y^1_{\text{int}})$
$A_{\text{int}} = w_{\text{int}} \times h_{\text{int}}$
5: **Compute areas of each box:**
$A_A = w_A \times h_A, \quad A_B = w_B \times h_B$
6: **Compute union area:**
$A_{\text{union}} = A_A + A_B - A_{\text{int}}$
7: **Return:** IoU $= \dfrac{A_{\text{int}}}{A_{\text{union}}}$

Precision and Recall

Precision and recall are fundamental metrics used to evaluate the performance of object detection models. These metrics provide valuable insights into the model's ability to identify objects of interest within images. Let's define some fundamental concepts before diving into precision and recall:

- **True Positive (TP):** These are istances where the model currently identifies and localize objectys, and the intersection over union (IoU) score between the predicted and the ground truth bounding box is equal or greater then a specified treshold.
- **False Positive (FP):** These are cases where the model incorrectly identifies an object that does not exist in the ground truth or where the predicted box has an IoU score below tha define treshold.
- **False Negative (FN):** These rapresent instances where the model fails to detect an object that is present in the ground truth. This means that the model has not detected an object that is present in the image, resulting in a missed detection.
- **True Negative (TN):** Not applicable in object detection. It represents correcly rejecting the absence of objects, but in object detection, the goal is to detect objects rather than the absence of them.

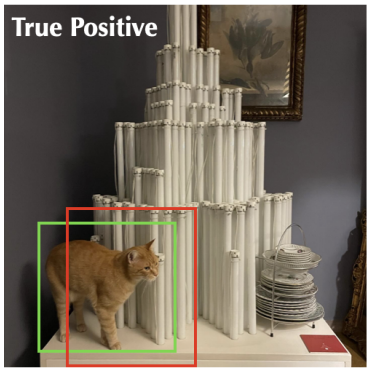


Figure 2.3: The object is there, and the model detects it, with an IoU above treshold.

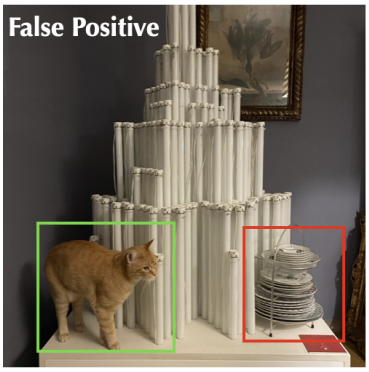


Figure 2.4: The object is there, but the predicted box has an IoU below the treshold.

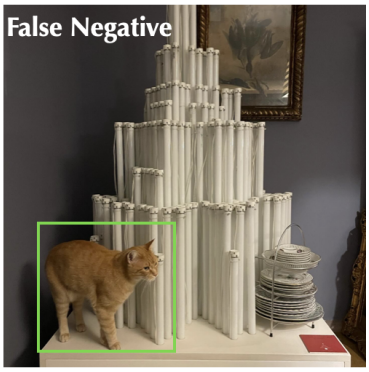


Figure 2.5: The object is there, and the model doesn't detect it. The ground truth object has no prediction.

Figure 2.6: Ground truth box is the green rectangle, while the predicted box is the red rectangle.

Let's not define the core metrics:

- **Precision**: Is a critical metric in model evaluation as it serves to quantify the accuracy of the positive predictions made by the model. It specifically assesses how well the model distinguishes true objects from false positives. In essence, precision provides insight into the model's ability to make positive predictions that are indeed accurate. A high precision score indicates that the model is skilled at avoiding false positives and provides reliable positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

- **Recall**: also known as sensitivity or true positive rate, is another essential metric used in evaluating model performance, especially in object detection tasks. Recall measures the model's capability to capture all relevant objects in the image. In essence, recall assesses the model's completeness in identifying objects of interest. A high recall score indicates that the model effectively identifies most of the relevant objects in the data.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

We can define the **Average Precision (AP)** as the area under the precision-recall curve, which is obtained by plotting precision against recall for different confidence thresholds [1]. The AP is a single value that summarizes the model's performance across different levels of precision and recall. It is calculated as follows:

$$AP = \int_0^1 P_\alpha(box_{gt}, box_{pred}) dR_\alpha \tag{2.4}$$

where P_α is the precision at a given IoU threshold α and R_α is the recall at the same threshold. Precision signifies the accuracy of the model's positive predictions, while recall quantifies the model's ability to succesfully identify all relevant objects. AP achieves a harmonious balance between false positive and false negatives, providing a comprehensive evaluation of the model's performance.

Mean Average Precision (mAP)

Mean Average Precision (mAP) extends the concept of Average Precision (AP) by providing a global summary of a model's performance across multiple object classes and, in some cases, multiple localization thresholds. While AP evaluates the precision-recall trade-off for a single class — by varying the confidence threshold and computing the area under the resulting precision-recall curve — mAP averages these AP values to reflect the model's overall detection capabilities.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.5}$$

where N is the number of classes and AP_i is the Average Precision for class i.

In practice, several versions of mAP are commonly reported. mAP@0.5 refers to the mean AP calculated at a fixed Intersection over Union (IoU) threshold of 0.5. mAP@[0.5:0.95], computes the average AP across ten IoU thresholds ranging from 0.5 to 0.95 (in increments of 0.05), providing a more rigorous and comprehensive evaluation that accounts for both classification and localization accuracy.

$$mAP@50 = \frac{1}{N} \sum_{k=1}^{N} AP_K@50 \tag{2.6}$$

[1] The confidence threshold is a tunable parameter used to filter out low-confidence predictions in object detection tasks. Each predicted bounding box is assigned a confidence score, typically representing the estimated probability that the detection both contains an object and correctly identifies its class. Varying this threshold affects the trade-off between precision and recall: lower thresholds increase recall but may introduce more false positives, whereas higher thresholds improve precision at the cost of reduced recall.

$$mAP@[50:95] = \frac{1}{N(95-50)} \sum_{\alpha=50}^{\alpha=95} \sum_{k=1}^{N} AP_K@\alpha \tag{2.7}$$

where N is the number of classes and α is the IoU threshold.

2.4 Transfer learning

Transfer learning is a machine learning technique in which knowledge gained through one task or data set is used to improve the performance of models on another related task and/or on a different data set. In other words, it uses what has been learned in one setting to improve generalization in another. The applications of transfer learning in deep learning are very interesting because models need a big amount of labeled data to learn and gain knoweledge, and a lot of times such big datasets are not avaiable.

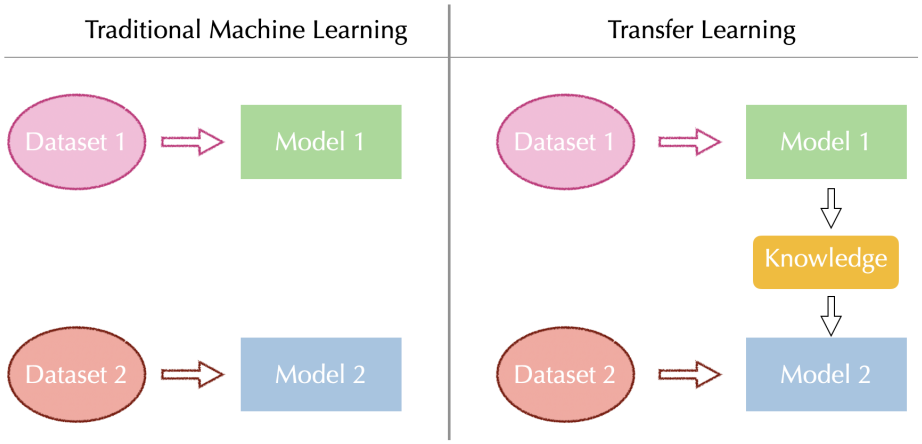


Figure 2.7

Traditional learning processes create a new model for each new activity based on the labeled data available. This is because traditional machine learning algorithms assume that training and test data come from the same feature space, so if the data distribution changes or if the trained model is applied to a new dataset, users need to retrain a newer model from scratch, even if they want to carry out an activity similar to that of the first model. Transfer learning algorithms, on the other hand, take already trained models or networks as a starting point, then apply that model's knowledge, gained in a task or initial source data to a new, but related activity or target data.

Depending on the amount of data available and the similarity between source domain and target domain, there are different transfer learning strategies:

- Feature extraction: freezes most of the model, training only the head. It is useful with very little data and similar domains.
- Partial fine-tuning: only the lowest layers freeze. Suitable for moderately different domains.
- Complete fine-tuning: only the lowest layers freeze. Suitable for moderately different domains.

In the context of medical imaging, transfer learning has shown promising results. Models pre-trained on large natural image datasets (e.g. ImageNet, COCO) can learn general low-level features (such as edges, textures, and shapes) that are also present in medical images. Fine-tuning such models on smaller labeled medical datasets allows for improved performance even with limited annotated data, making transfer learning a valuable tool in the medical AI pipeline.

The following chapters will explore specific cases of transfer learning applied to medical image datasets, comparing various pre-training strategies and model architectures.

2.5 Curriculum Learning

Curriculum Learning is a training strategy inspired by the way humans learn, initially tackling simple tasks and then moving on to more complex ones. Introduced by Bengio et al. in 2009 [1], this approach proposes to organize the training of a model according to an ordered sequence of examples, selected based on a difficulty criterion.

Instead of presenting all data simultaneously and in random order, Curriculum Learning involves the progressive selection of subsets of data, which are introduced in a gradual manner during training. This controlled sequence allows the model to first acquire the simplest and most robust regularities, and then refine its predictive ability by tackling more complex cases.

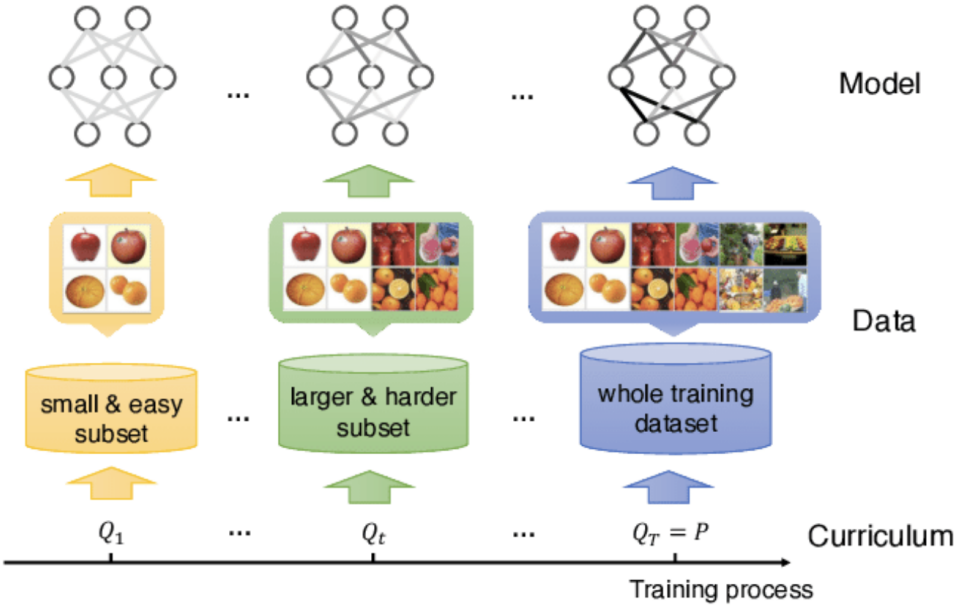


Figure 2.8

An effective curriculum requires defining three basic components:

- Difficulty criterion: is the mechanism through which a difficulty level is assigned to each example. It can be defined explicitly on the basis of known characteristics (eg geometric properties, data quality, object dimensions), or it can emerge automatically, for example by exploiting the loss obtained by the model during the early eras or the uncertainty of predictions.

- Pacing: is the function that regulates the pace at which the most difficult examples are introduced into training. Pacing can be defined a priori (e.g. by adding a certain number of examples to each epoch), or determined dynamically, for example by evaluating the performance of the model on a validation subset.
- Presentation order: consists of the construction of a sequence of data (or subsets) to be presented to the model. This order can be fixed or updated during training, and strongly affects the effectiveness of the curriculum.

This type of approach has proven useful in scenarios where the data has significant variability, contains noisy or unbalanced examples, or when it is desired to accelerate the convergence of optimization. Furthermore, Curriculum Learning can help prevent the model from learning abnormal characteristics or statistical noise too early, thus improving generalization.

Applications

In the visual field, a curriculum can be constructed in many ways: for example, a classification model can initially be trained on high-quality images and well-defined content, and then tackle more complex examples such as deformed, moving, partially occluded objects or present in unfavorable lighting conditions. In other contexts, such as the regression of physical quantities from images, the difficulty may depend on the structural variability of the visual content or the complexity of the information flow required for accurate prediction.

Practical considerations

Despite the benefits observed in numerous works, the effectiveness of Curriculum Learning is highly dependent on

- A correct definition of the difficulty;
- An appropriate pacing;
- the coherence of the curriculum with the final task.

In the absence of a clear criterion for distinguishing easy examples from difficult ones, or if pacing is poorly calibrated, the approach can even worsen performance compared to standard training.

However, if well designed, Curriculum Learning can be a powerful tool for controlling training, especially in contexts where data distribution is uneven, the useful signal is weak or learning can benefit from gradual progression.

Chapter 3

Hadronic Collisions and Jet Reconstruction

3.1 High energy collisions in hadronic accelerators

In a hadronic collider, such as the LHC or the Tevatron, beams of hadrons—primarily protons and antiprotons—are collided at extremely high energies. Unlike electrons, protons are not elementary particles; they possess a complex internal structure composed of quarks and gluons. This internal structure is described by the parton distribution functions (PDFs), denoted in Fig. 3.1 as $f_i(x_j, \mu_F)$, where i indicates the parton type (quark or gluon), and j refers to the beam. The PDF represents the probability of finding a parton of type i carrying a fraction x_j of the proton's momentum at the time of collision.

Therefore, when two protons P_1 and P_2 collide, the interaction does not occur between the protons themselves, but rather between their constituents—the partons—with momenta $x_j P_j$. The partons that do not participate in the hard scattering process are referred to as underlying events, as illustrated in the figure.

When high-energy quarks are produced in the final state, they emit additional partons through a cascading process known as parton showering. This process continues until the energy scale drops sufficiently, at which point the partons recombine into observable hadrons in a process called hadronization.

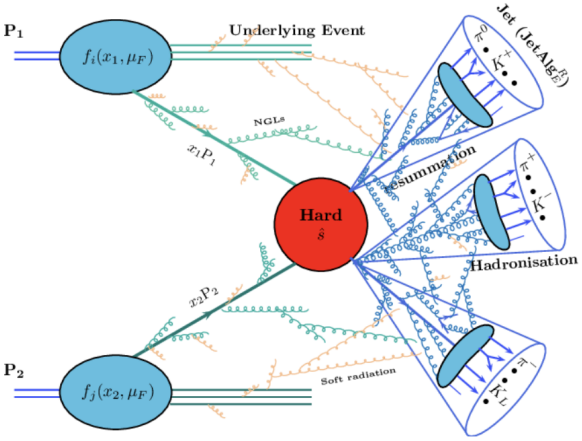


Figure 3.1: Schema di un collisione adronica. I protoni si scontrano e producono una cascata di particelle, che si trasformano in jet di particelle.

Jets

In order to reconstruct the primary event that occurred before the parton shower and hadronization, the strategy is to measure the energy of the final-state particles using calorimeters, and then group them based on their transverse momentum and direction. This procedure leads to the formation of what are known as jets. Jets are essentially the result of a clustering algorithm that groups final-state particles with similar transverse momentum and direction, allowing us to trace back to the primary parton-level event that initiated the particle cascade.

3.2 Useful variables to describe an hadronic process

In collider physics, the kinematic properties of particles are typically described using variables that are well suited to the cylindrical geometry of the detectors and to the characteristics of the collisions. Among these, the most commonly used are the transverse momentum p_T, the azimuthal angle ϕ, and the pseudorapidity η.

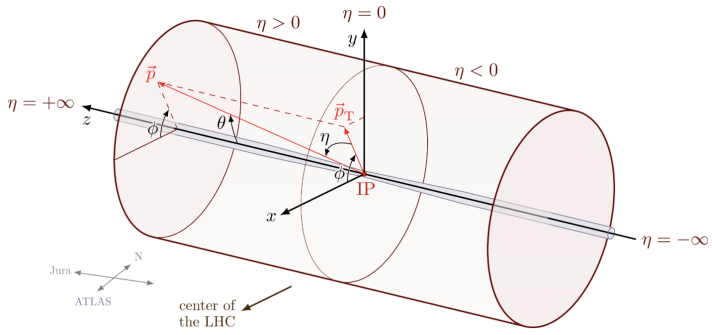


Figure 3.2: Kinematic variables used in collider physics. The transverse momentum p_T is the component of momentum perpendicular to the beam axis, ϕ is the azimuthal angle, and η is the pseudorapidity.

p_T - transverse momentum

The transverse momentum p_T is defined as the component of a particle's momentum that is perpendicular to the beam axis (commonly referred to as the z-axis in collider experiments). Mathematically, it is given by:

$$p_T = \sqrt{p_x^2 + p_y^2} \tag{3.1}$$

where p_x and p_y are the components of the momentum in the plane transverse to the beam (also called the transverse plane).

This quantity is especially important in hadron colliders, where the incoming protons travel along the z-axis, but the exact longitudinal momentum of the interacting partons is unknown on an event-by-event basis. However, since the protons travel toward each other along the beam axis, the net transverse momentum of the system before the collision is essentially zero. As a result, conservation of momentum implies that the total transverse momentum of the final-state particles should also be zero, up to detector resolution and effects from invisible particles.

For this reason, p_T is a Lorentz-invariant quantity under boosts along the beam axis, and provides a frame-independent way to study the dynamics of the collision. It is also widely used in searches for new physics, where an imbalance in the total transverse momentum (often called missing transverse energy, or MET) can signal the presence of non-interacting or undetected particles.

ϕ - azimuthal angle

The azimuthal angle ϕ describes the direction of a particle in the transverse plane, i.e., the plane perpendicular to the beam axis (the x-y plane if the beam is aligned along the z-axis). It is defined as the angle between the particle's transverse momentum vector and a fixed reference axis (typically x-axis) measured counterclockwise:

$$\phi = \tan^{-1}\left(\frac{p_y}{p_x}\right) \tag{3.2}$$

The azimuthal angle takes values in the range $[-\pi, \pi]$ or $[0, 2\pi]$, depending on convention. This variable is invariant under longitudinal boosts.

η - pseudorapidity

The pseudorapidity η is a measure of the angle of a particle relative to the beam axis, defined as:

$$\eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right) \tag{3.3}$$

where θ is the polar angle of the particle with respect to the beam axis.

3.3 Algorithms for jet reconstruction

Jet reconstruction algorithms are essential for identifying and grouping particles produced in high-energy collisions into jets, which represent the hadronic remnants of the original partons. These algorithms can be broadly categorized into two main types: **cone algorithms** and **sequential clustering algorithms**.

3.3.1 Cone Algorithms

Cone algorithms assume that particles in jets will show up in conical regions and thus they cluster based on $\eta - \phi$ space, resulting in jets with rigid circular boundaries. Most of the cone algorithms are iterative cones (IC). In such algorithms, a seed particle i sets some initial direction, and one sums the momenta of all particles j within a circle of radius R around i in azimuthal angle ϕ and pseudorapidity η, i.e. taking all j such that

$$\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2 < R^2 \tag{3.4}$$

where y_i and y_j are the pseudorapidities of particles i and j, respectively, and ϕ_i and ϕ_j are their azimuthal angles. The total momentum of the jet is then calculated as the sum of the momenta of all particles within this cone. The direction of the jet is then recalculated as the weighted average of the momenta of all particles within the cone, and the process is repeated until no more particles can be added to the jet.

3.3.2 Sequential Clustering Algorithms

Gli algoritmi a ricombinazione sequenziale utilizzano informazioni geometriche ed energetiche per aggregare le particelle che potenzialmente arrivano dallo stesso partone, ricostruendo così la storia del getto.

k_T Algorithm

The k_T algorithm is a sequential clustering algorithm that uses the concept of distance between particles to group them into jets. The distance between two particles i and j is defined as:

$$d_{ij} = \min(p_{T,i}^2, p_{T,j}^2)\frac{\Delta R_{ij}^2}{R^2} \tag{3.5}$$

while the distance between a particle i and the beam axis is defined as:

$$d_{iB} = p_{T,i}^2 \tag{3.6}$$

For each particle, the algotithm calculated the minumum distance betwen d_{ij} and d_{iB}.

$$d_{min} = \min(d_{ij}, d_{iB}) \tag{3.7}$$

If d_{min} is smaller than a predefined threshold, the particle is added to a jet; otherwise, it is considered a standalone particle. The algorithm proceeds iteratively, recalculating the distances after each clustering step until all particles are grouped into jets or no more particles can be clustered.

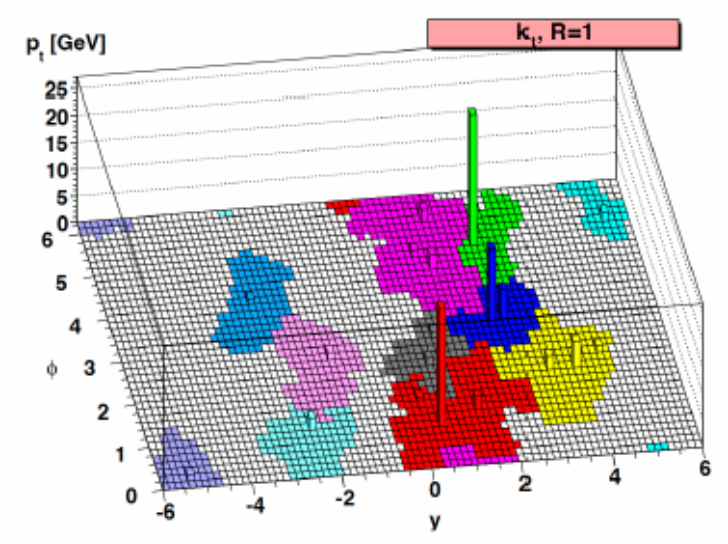


Figure 3.3: Schematic representation of the k_T algorithm. The algorithm iteratively clusters particles based on their transverse momentum and distance in $\eta-\phi$ space, forming jets.

Anti-k_T Algorithm

The anti-k_T algorithm is a variant of the k_T algorithm that uses a different distance measure to cluster particles. It is designed to produce jets with more uniform shapes and

is particularly effective in high-energy collisions where jets can be closely spaced. The distance between two particles i and j is defined as:

$$d_{ij} = \min(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2})\frac{\Delta R_{ij}^2}{R^2} \tag{3.8}$$

while the distance between a particle i and the beam axis is defined as:

$$d_{iB} = \frac{1}{p_{T,i}^{2\beta}} \tag{3.9}$$

The algorithm proceeds similarly to the k_T algorithm, calculating the minimum distance between particles and the beam axis, and clustering them into jets based on this distance. The key difference is that the anti-k_T algorithm tends to produce jets with more uniform shapes, making it more suitable for high-energy collisions where jets can be closely spaced.

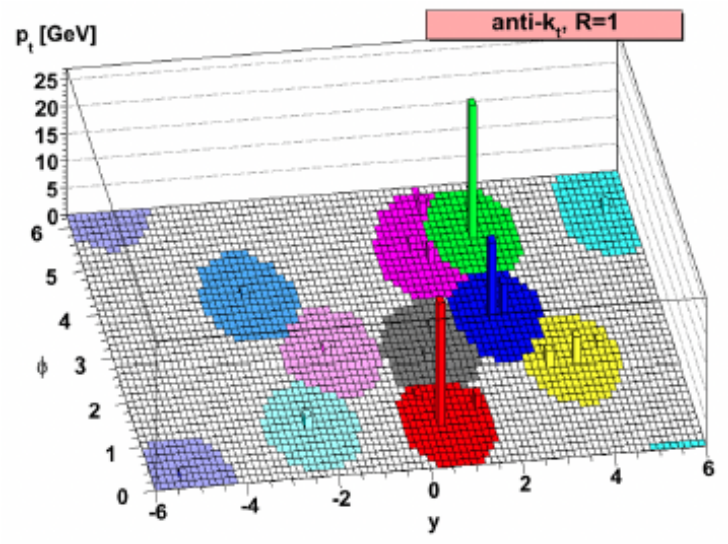


Figure 3.4: Schematic representation of the anti-k_T algorithm. The algorithm iteratively clusters particles based on their transverse momentum and distance in $\eta - \phi$ space, forming jets with more uniform shapes.

Cambridge/Aachen Algorithm

The Cambridge/Aachen algorithm is another sequential antik_T clustering algorithm that uses a different distance measure to cluster particles. It is designed to produce jets with more uniform shapes and is particularly effective in high-energy collisions where jets can be closely spaced. The distance between two particles i and j is defined as:

$$d_{ij} = \Delta R_{ij}^2 \tag{3.10}$$

while the distance between a particle i and the beam axis is defined as:

$$d_{iB} = 1 \tag{3.11}$$

Physically, the differences between the three algorithms are contained in the momentum weighting. For the k_T algorithm, the weighting is done to preferentially merge

costituents with low transverse momentum with respect their nearest neighbours. For the antik_T, the weighting is done so as to preferentially merge constituents with high transverse momentum with respect to their nearest neighbours. The C-A algorithm relies only on distance weighting with no k_T weighting at all.

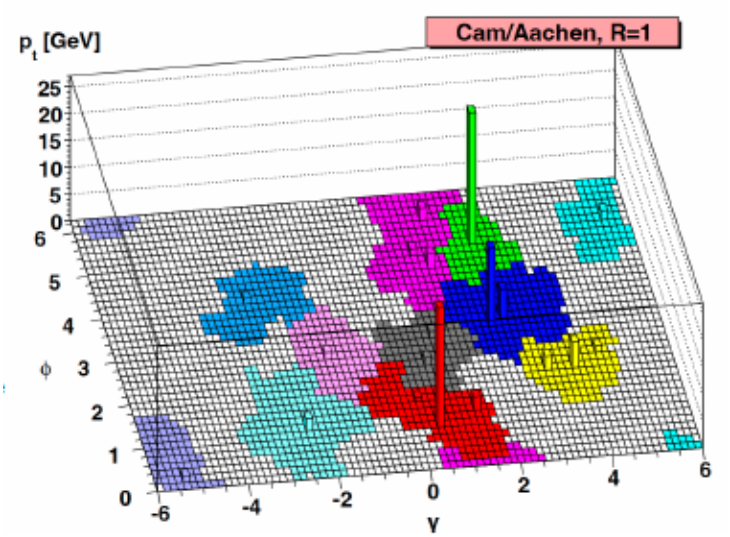


Figure 3.5

Chapter 4

Synthetic Particle Dataset

4.1 Jet Generation

4.2 Image Assembling

4.3 Background Generation

Chapter 5

Medical Dataset

5.1 NLST - National Lung Screening Trial Dataset

The National Lung Screening Trial (NLST) is a large-scale, multicenter clinical study designed to evaluate the effectiveness of low-dose computed tomography (CT) in reducing lung cancer mortality among high-risk individuals. The trial enrolled over 53,000 participants who underwent annual screening exams with either low-dose CT or standard chest X-rays. The main goal of the NLST was to assess whether early detection of lung cancer via low-dose CT could improve patient outcomes.

The NLST dataset includes chest CT scans, each consisting of multiple axial slices, typically around 143 slices per scan. While the dataset is extensive, not all slices contain clinically relevant abnormalities or tumor lesions. Radiologists reviewed the scans independently, identifying and annotating suspicious findings, with particular attention to nodules or masses measuring 4 mm or larger.

For the purposes of this work, the NLST dataset is used solely as a source of annotated medical images to train a deep learning model for lung lesion detection. Since the focus is on supervised learning, only slices containing malignant lesions with clear annotations were selected. This involved pruning the dataset by filtering slices based on clinical annotations provided in accompanying metadata files. After this process, approximately 9,000 slices containing malignant lesions equal to or larger than 4 mm were retained for training.

It is important to note that slices with benign lesions were not included in the detection task due to the absence of bounding box annotations, which limits their utility for supervised localization training. The dataset thus offers a robust foundation of positive samples for the development of lesion detection algorithms but presents certain limitations in terms of lesion diversity and negative sample representation.

In summary, the NLST dataset serves as a valuable resource of high-quality chest CT images with clinically verified tumor annotations, enabling the training and evaluation of machine learning models aimed at improving lung cancer screening and diagnosis. This work leverages this dataset to develop and assess a neural network tailored to detect lung nodules and masses from axial CT slices, focusing on malignant cases with precise spatial labels.2

Dataset Annotations

The annotations are provided as a CSV file containing the following fields:

- **PID**: Unique identifier for each patient.
- **Slice Number**: The specific slice within the CT scan, which also corresponds to the filename.
- **CT Filter**: Indicates the type of filter applied to the CT image.
- **x, y, width, height**: Coordinates defining the bounding box around the lesion; x and y represent the top-left corner of the box, while *width* and *height* specify its dimensions.

These annotations serve as the ground truth labels for supervised learning, enabling the model to localize and detect lesions within the CT slices.

Extraction of the Pixel Array

The CT images themselves are stored in the DICOM format, which is the standard for medical imaging data. Each CT scan is composed of multiple axial slices, each represented as a two-dimensional image. The pixel values in these images are expressed in Hounsfield Units (HU), a scale that quantifies tissue radiodensity.

Each DICOM file contains two main components relevant to this work:

- **Pixel Array**: A 2D array of pixel intensities corresponding to the image. Each pixel value is typically stored as a 16-bit integer, representing the radiodensity at that location.
- **Metadata**: Information about the image acquisition and patient, including details such as patient ID, acquisition date, image orientation, pixel spacing, and slice thickness. This metadata is essential for accurate interpretation and processing of the images.

Understanding both the pixel data and the associated metadata is crucial for proper pre-processing and analysis of the CT images.

Image Metadata

The metadata associated with each CT slice provides crucial information regarding image acquisition parameters and physical properties, which influence how the images are interpreted and processed. Key metadata fields include:

- **Pixel Spacing (mm)**: Physical distance between adjacent pixels along the x and y axes, measured in millimeters.
- **Slice Thickness (mm)**: Thickness of each axial slice along the z axis, measured in millimeters.
- **Field of View (FOV) (cm)**: Physical size of the imaged area, typically given in centimeters.
- **Image Orientation**: Specifies the orientation of the image using row and column direction vectors.

- **Image Position**: The 3D spatial coordinates (x, y, z) indicating the position of the slice within the patient's body.

Properly accounting for these parameters ensures that spatial relationships and measurements within the CT scans are accurately maintained during model training and inference.

5.2 DLCS - Duke Lung Cancer Screening Dataset

Extraction of the 2D slices

Dataset Pruning

5.3 Pre processing the images

Medical image preprocessing is essential to train accurate tumor detection models. Raw scans often contain noise, inconsistent resolutions, or irrelevant regions that can mislead algorithms. Our pipeline standardizes the data through key steps: resampling ensures uniform resolution, windowing highlights tumor-relevant contrasts, and masking removes healthy tissue to focus analysis. These optimizations allow the model to learn meaningful patterns, improving its ability to detect tumors reliably across diverse clinical cases.

Resampling

Since the CT images were acquired using different machines and protocols, the pixel spacing (i.e., the physical dimension per pixel or voxel) is not consistent across all scans, which in turn affects the apparent size of lesions. To standardize, we apply resampling to ensure that all voxels represent the same physical size in centimeters. Consequently, the pixel dimensions of the images may change. To determine a suitable uniform voxel size, the pixel spacing was extracted from each image and computed the mean (x and y spacing is equal), resulting in:

$$(\text{pixel spacing})_x = (\text{pixel spacing})_y = 0.623mm$$

Then, using this target spacing, all images are resampled accordingly. It can be achived using SimpleITK, a medical imaging library, which provides a robust implementation for resampling. For example:

Algorithm 2 Resampling DICOM Slices

Input:

- Directory D containing DICOM files d
- Target pixel spacing in x and y: $t_x = t_y = 0.623$ mm

Output: Resampled PNG slices
for each file $d \in D$ **do**
    I = read image with SimpleITK
    Get original Spacing (sp_x, sp_y) and Size (si_x, si_y)
    New Size$_i$ $= \dfrac{si_i \cdot sp_i}{t_i}$ for i = x,y

    Apply SimpleITK function to resample images:
    Set new size, set new spacing
    Set new output origin, output direction and get interpolator

    Execute these function on I to obtain the resampled image
end for

Padding

After resizing, all images now share a uniform pixel spacing, but their pixel dimensions differ. Since YOLOv8—the model employed in this thesis—requires square images of identical pixel size as input, a padding step is necessary.

The padding is applied so that the original image is placed in the top-left corner of a black canvas, whose side length corresponds to the nearest multiple of 32 greater than or equal to the largest resampled image dimension:

- NLST: 704x704 pixel, spacing 0.623 mm x 0.623 mm
- DLCS: 736x736 pixel, spacing 0.7 mm x 0.7 mm

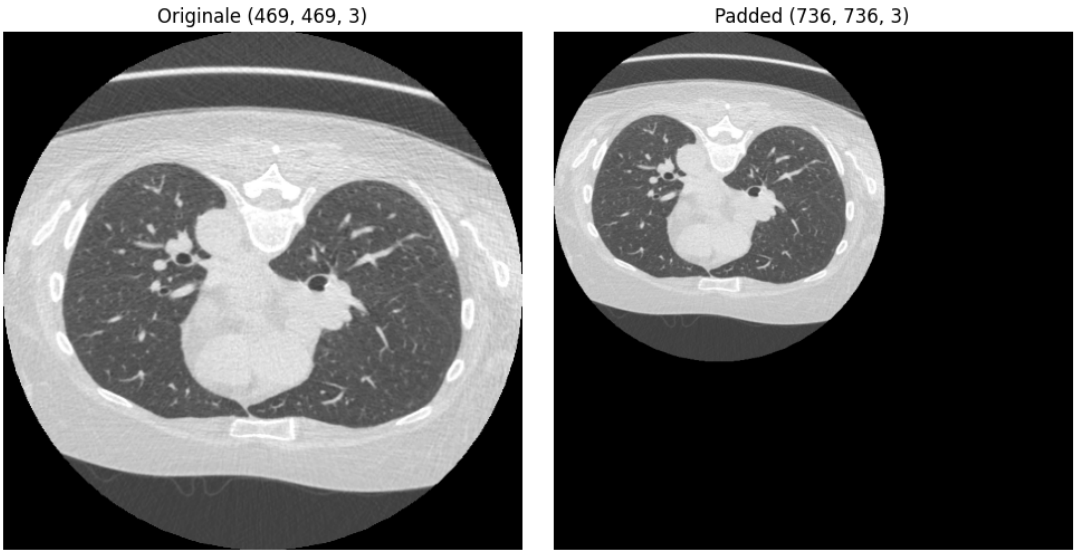


Figure 5.1: On the left: slice after resampling, on the right: same slice after padding

Windowing

As described in Chapter 1, the range of intensity values in a CT scan depends on the tissue absorption coefficients, typically spanning approximately from -1000 to +3000 Hounsfield Units (HU). Because this range does not have fixed minimum or maximum limits, directly visualizing the full range is not practical, especially when converting images to formats like PNG, which support only a limited number of intensity levels. To effectively visualize relevant anatomical structures, we apply a process called windowing, which restricts the displayed intensity values to a specific range that highlights the tissues of interest. In this case, to emphasize lung structures, we set the window center at -600 HU and the window width at 1700 HU, values that are well established in the literature and illustrated in the accompanying figure. his means that only intensity values within the window boundaries—calculated as:

$$\text{window minimum} = -600 - \frac{1700}{2} = -1450$$

$$\text{window maximum} = -600 + \frac{1700}{2} = +250$$

—are displayed linearly. Any values outside this range are clipped to the nearest boundary value to avoid distortion. This windowing technique is applied consistently to all images from both datasets, resulting in enhanced visualization of pulmonary structures, as shown below:

The following figure illustrates the step-by-step processing of a representative CT slice. On the left, the original image is shown with its native dimensions and pixel spacing. In the middle, the image after resampling is presented, including updated spatial resolution parameters. Finally, on the right, the windowing technique is applied to enhance the visualization of pulmonary structures by adjusting the intensity range. Dimensions and spacing values are displayed above each image for clarity.

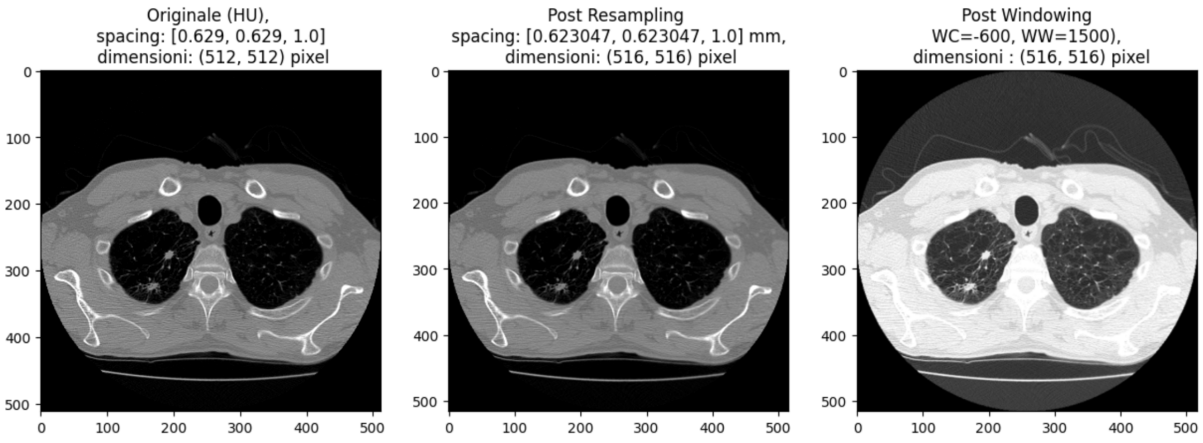


Figure 5.2: From the left: original image with size 512x512 and spacing 0.629x0.629 mm, post resampling image has dimension 516x516 after setting the new spacing 0.623x0.623, post windowing on the resampled image with typical WW and WC

RBG Windowing

CT images are grayscale, meaning they contain a single intensity channel. Since many models are pre-trained on three-channel images, a more dynamic intensity windowing approach could be employed to highlight different anatomical structures. One possible strategy is to construct a three-channel image where each channel applies a distinct windowing configuration. Before doing so, it is essential to analyze the distribution of pixel intensities in the dataset to determine the most effective windowing settings for enhancing lesion visibility. Figure 5.3 shows the pixel intensity distribution for the DLCS dataset.

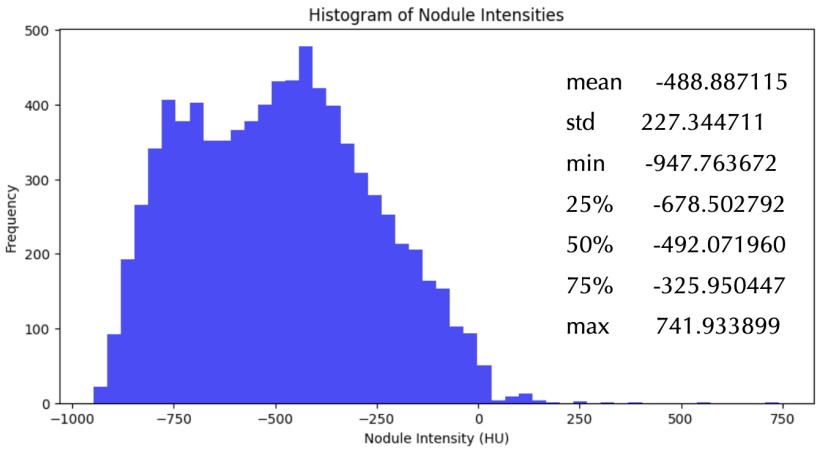


Figure 5.3: Sono riportati in figura i valori massimo, minimo, medio, deviazione standard (std), e il primo (25%), secondo (50%) e terzo (75%) percentile.

- **Channel Red (R)**: standard lung windowing with a window center (WC) of -600 and a window width (WW) of 1700.
- **Channel Green (G)**: window centered at the mean pixel intensity of the dataset (-488), with a width of 1000 to ensure sufficient contrast.
- **Channel Blue (B)**: window parameters chosen to cover the intensity range between the 1st and 3rd percentiles, corresponding to a center of -325 and a width of 700.

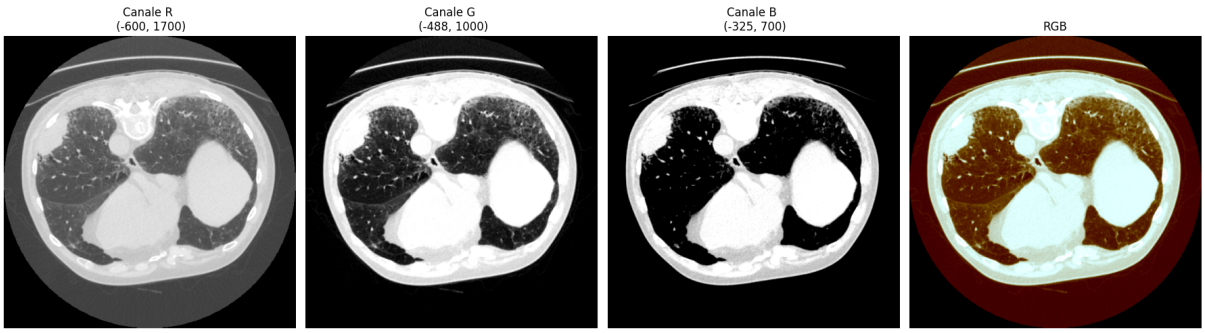


Figure 5.4: From left to right: channels R, G, and B, followed by the final RGB image obtained by stacking the three channels.

Masking

Another useful preprocessing step is *masking*, a technique aimed at isolating only the main anatomical structure of interest—in this case, the lungs—in order to prevent the network from focusing on irrelevant areas outside them, which certainly do not contain lesions.

Since CT values are expressed in Hounsfield Units (HU), we construct a binary mask by selecting HU values known to be characteristic of lung tissue. In the resulting mask, a value of 1 (white) corresponds to the lung, while a value of 0 (black) corresponds to the background.

Lesions, being calcifications, have much higher HU values than typical lung tissue. To ensure they are not inadvertently removed by the mask, we apply the `sitk.BinaryFillHoles` function from the `SimpleITK` library, which fills enclosed empty regions. Additionally, we implement custom functions to remove structures such as the table and trachea, which are often not excluded by a simple mask.

Applying the mask is straightforward: the original pixel array is multiplied element-wise by the binary mask.

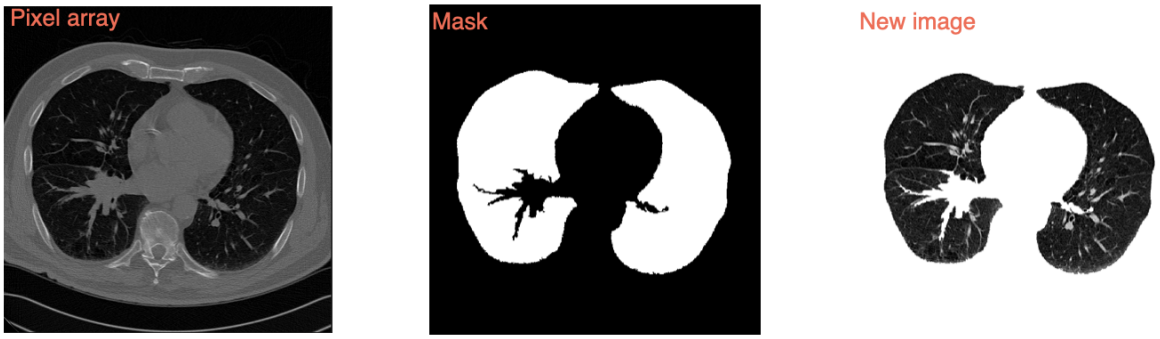


Figure 5.5: The leftmost image shows the original pixel array (prior to windowing or resampling), the center image displays the binary mask, and the rightmost image presents the final result after applying the mask.

Algorithm 3 Masking algorithm

Input: Directory D containing resampled and windowed DICOM files d
Output: Masked images

for each file $d \in D$ **do**
    I = read image with SimpleITK
    Get pixel array from image metadata
    Mask = pixel array < -300

    Apply function to **fill holes**
    Apply function to **remove trachea**
    Apply function to **remove table**

    New image = pixel array × mask

end for

We can choose whether to apply the windowing before or after the masking, the result is irrelevant.

Figure 5.6: Original image.

Figure 5.7: After windowing

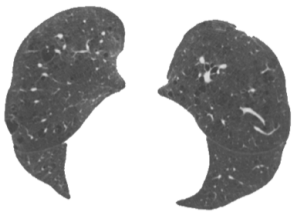


Figure 5.8: Masked image (after windowing).

Chapter 6

Dataset creation and Training Strategies

6.1 YOLO dataset

Before discussing the results, it is necessary to provide some background on Ultralytics, the company that developed the YOLO model.

In this thesis, we focus on supervised training, in which the model is provided with images and corresponding annotations during the training phase. These annotations include the bounding box coordinates (top-left coordinate, height, and width) as well as the object class, since our task is object detection.

In particular, Ultralytics requires the data to be organized in a specific structure, as illustrated in Figure 6.1. The dataset must be split into training, validation, and test sets, as in any standard workflow. Images are stored in their respective subfolders under `images`, while annotations are placed in corresponding subfolders under `labels`. Each annotation must be stored in a `.txt` file with the same name as its corresponding image.

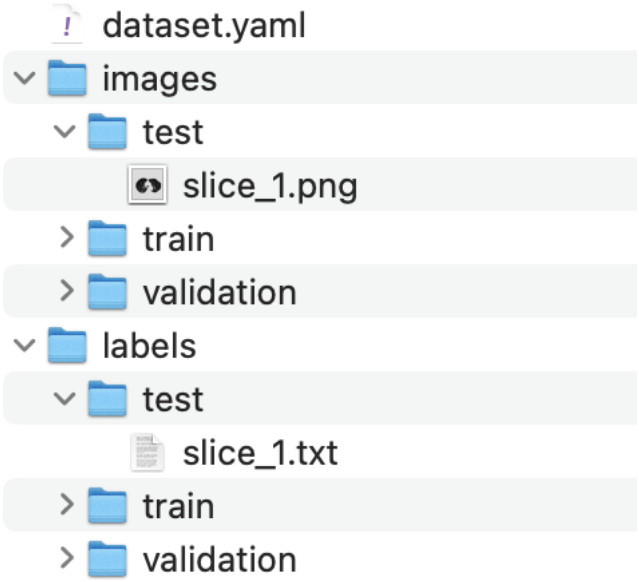


Figure 6.1

Inside the folder containing all images and annotations, a configuration file named `dataset.yaml` must be included. This file is provided to the model during training. It

specifies the relative paths to the training, validation, and test image directories (the paths to the labels are not required, as they follow the same folder structure as the images, and the annotation files share the same names as their corresponding images). The file must also define the number of classes and their names.

```
train: ../dataset/images/train
val: ../dataset/images/validation
test: ../dataset/images/test
nc: 2
names: ['cat', 'dog']
```

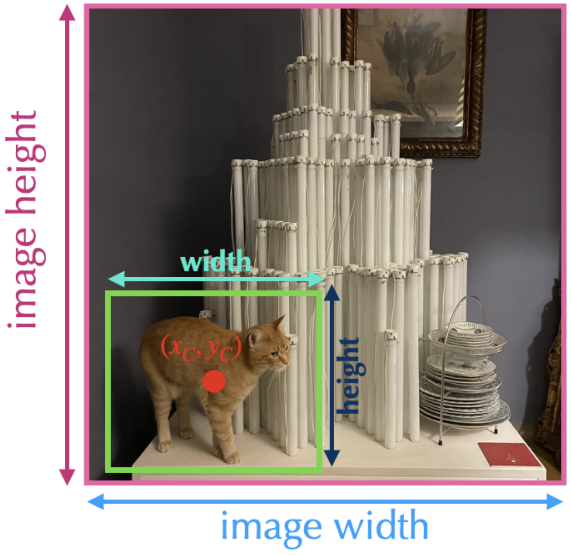
The `train`, `val`, and `test` keys specify the relative paths to the training, validation, and test image directories, respectively. The `nc` key indicates the number of classes, while the `names` key provides a list of class names.

Annotations

YOLO also requires annotations to be provided in a specific format:

$$\text{class} \quad \frac{x_{\text{center}}}{\text{image width}} \quad \frac{y_{\text{center}}}{\text{image height}} \quad \frac{\text{width}}{\text{image width}} \quad \frac{\text{height}}{\text{image height}}$$

Each annotation file contains one line for each object present in the corresponding image. The first value is the class index, followed by the normalized coordinates of the bounding box: the x- and y-coordinates of the center of the bounding box, and its width and height, all normalized by the dimensions of the image.



6.2 Training Settings

Training a deep learning model involves feeding it data and adjusting its parameters so that it can make accurate predictions.

6.2.1 Hyperparameters

The training process is controlled by a set of hyperparameters, which are parameters that are not learned during training but are set before the training begins. These hyperparameters include:

- **Learning Rate**: This is a crucial hyperparameter that determines how much the model's weights are updated during training. A higher learning rate can lead to faster convergence but may also cause the model to overshoot the optimal solution.
- **Batch Size**: This refers to the number of training examples used in one iteration of training. A larger batch size can lead to more stable gradients but requires more memory.
- **Number of Epochs**: An epoch is one complete pass through the entire training dataset. The number of epochs determines how many times the model will see the entire dataset during training.
- **Optimizer**: The optimizer is an algorithm used to update the model's weights based on the computed gradients. Common optimizers include Adam, SGD.
- **Data Augmentation**: This technique involves applying random transformations to the training data to increase its diversity and improve the model's generalization ability. Common augmentations include rotation, scaling, flipping, and color adjustments.
- **Early Stopping Criteria**: This is a condition that stops training when the model's performance on the validation set does not improve for a specified number of epochs. It helps prevent overfitting and saves computational resources.
- **Weight Decay**: technique used to prevent overfitting by adding a penalty term to the loss function based on the magnitude of the model's weights. It encourages the model to learn simpler patterns.
- **Dropout**: A regularization technique that randomly sets a fraction of the model's neurons to zero during training. This helps prevent overfitting by reducing the model's reliance on specific neurons.

6.2.2 Loss Function

The loss function is a critical component of the training process, as it quantifies how well the model's learning and the predictions match the ground truth. In the case of object detection using YOLOv8, the loss function typically consists of three components:

1. Box Loss

The box loss measures the difference between the predicted bounding box coordinates and the ground truth coordinates. It is calculated as the Complete Intersection over Union (**CIoU**) loss, which is an extension of the Intersection over Union metric discussed in Chapter 2, and takes into account not only the overlap between the predicted and ground

truth boxes but also their aspect ratio and distance between their centers. The CIoU loss is defined as:

$$L_{CIoU} = 1 - IoU - \frac{d^2}{c^2} - \alpha v \tag{6.1}$$

where IoU is defined in Eq.2.1, where v is a function of the boxes widths and heights and α is a trade-off parameter function of IoU.

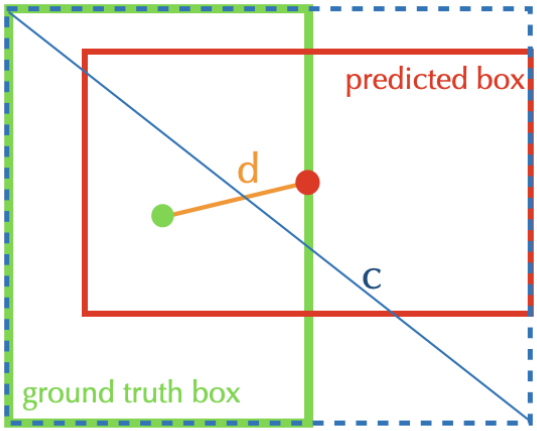


Figure 6.2: Representation of a ground truth box in green and its prediction in red. "d" is the euclidean distance between the centers, while "c" is the diagonal of the smallest box including the green and the red one.

2. Classification Loss

3. Dual Focal Loss

The classification loss measures the difference between the predicted class probabilities and the ground truth class labels. YOLOv8 uses a dual focal loss, which is a combination of binary cross-entropy loss and focal loss. The binary cross-entropy loss is used for multi-label classification tasks, while the focal loss helps to address class imbalance by down-weighting easy-to-classify examples and focusing more on hard-to-classify examples. The dual focal loss is defined as:

$$L_{dual} = \frac{1}{N} \sum_{i=1}^{N} (-\alpha(1-p_i)^{\gamma} \log(p_i) - (1-\alpha) p_i^{\gamma} \log(1-p_i)) \tag{6.2}$$

where N is the number of classes, p_i is the predicted probability for class i, α is a balancing factor, and γ is a focusing parameter that controls the down-weighting of easy examples.

Total Loss

The total loss for YOLOv8 is a weighted sum of the box loss, classification loss, and dual focal loss:

$$L_{total} = \lambda_{box} L_{CIoU} + \lambda_{cls} L_{cls} + \lambda_{dual} L_{dual} \tag{6.3}$$

where λ_{box}, λ_{cls}, and λ_{dual} are hyperparameters that control the relative importance of each loss component and can be set in the configuration scrpit. Default values are: $\lambda_{box} = 7.5$, $\lambda_{cls} = 0.5$, and $\lambda_{dual} = 1.5$.

6.2.3 Augmentation

Augmentation techniques are essential for improving the robustness and performance of YOLO models by introducing variability into the training data, helping the model generalize better to unseen data. Here's a list of the augmentations that have been used in this work and can be set in the configuration file with the others hyperparameters:

- `hsv`: modifies the hue, saturation, and value of the image, allowing the model to learn to recognize objects under different lighting conditions and color variations.
- `degrees`: applies random rotations to the image, helping the model become invariant to object orientation.
- `translate`: shifts the image horizontally and/or vertically, allowing the model to learn to recognize objects that may not be centered in the frame.
- `scale`: resizes the image, helping the model learn to recognize objects at different scales.
- `shear`: applies a shear transformation to the image, which can help the model learn to recognize objects that may be distorted or skewed.
- `flipud`: flips the image vertically, allowing the model to learn to recognize objects that may appear upside down.
- `fliplr`: flips the image horizontally, allowing the model to learn to recognize objects that may appear mirrored.
- `mosaic`: combines multiple images into a single image, creating a more diverse training set and helping the model learn to recognize objects in different contexts.
- `mixup`: combines two images and their corresponding annotations, creating a new image that contains features from both. This technique helps the model learn to recognize objects that may be partially occluded or overlapping.

Chapter 7

Training Results

7.1 Medical Images

7.2 Jet Images - Curriculum Learning

7.3 Transfer Learning in a data scarcity regime

7.4 Transfer learning in a normale data regime

Conclusion

Bibliography

[1] Y. Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. *Journal of the American Podiatry Association*, 60:6, 06 2009.

[2] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.