UCLouvain

epl

École polytechnique de Louvain

Streamlined hybrid annotation framework using scalable codestream for bandwidth-restricted UAV object detection

Author: **Simon Delvaux**
Supervisor: **Benoît Macq**
Readers: **Christophe De Vleeschouwer, Karim El Khoury, Tiffanie Godelaine**
Academic year 2023–2024
Master [120] in Electrical Engineering

Abstract

The success of emergency missions depends on the rescuers' ability to intervene quickly. A work increasingly facilitated by unmanned aerial vehicles, which provide crucial visual information on the situation. However, the communication capacity is often constrained by a limited bandwidth, preventing fast transmission and thereby delaying the quick decision-making necessary in emergency situations. To address these challenges, this master thesis presents a hybrid annotation framework adapted for emergency situations that takes advantage of the JPEG 2000 compression algorithm. The framework is designed to facilitate object detection tasks with constrained bandwidth. Fast annotation is ensured by a fine-tuned deep learning model working at low-resolution, and reliability is provided by the correction of human annotator experts in enhanced resolution areas of uncertainty. It has been shown that the proposed framework decreases the response time up to 15 times while maintaining satisfactory performance, losing only 10% compared to a hybrid baseline approach.

Acknowledgements

I would like to thank all the people who helped me in one way or another during this master's thesis.

Firstly, I would like to thank my supervisor, Prof. Benoît Macq, for giving me the opportunity to work on this subject, but above all for his support and guidance throughout this master thesis.

Then, I would like to express special thanks to my thesis advisers, Karim El Khoury and Tiffanie Godelaine, who have helped me in my work by always being available to answer my questions, give me advice, and give me feedback. I would like to thank them again for helping me until the final deadline for this manuscript.

I would like to thank Prof. Christophe De Vleeschouwer for serving as my reader and jury member.

Furthermore, I express my profound gratitude to Baptiste Sambon and Javiera Viancos for their support and taking their precious time to review my master thesis at the end of this year.

Finally, I would like to thank my entourage for their friendly support and encouragement throughout my years of study. They helped me through the tougher moments and got me to where I am today.

For this work, *DeepL*, *LanguageTool*, and *ChatGPT* have been used to improve sentence formulation and check grammar.

Contents

Chapter 1

Introduction

In recent years, the use of Unmanned Aerial Vehicles (UAVs) has become more frequent in areas affected by disasters [23], where search and rescue missions [36, 46], damage assessment, and hazard detection are conducted. They give rescuers real-time information to plan their operations and to keep up-to-date insights about the changes in the terrain. UAVs are particularly helpful in the wilderness, where rescue workers are unable to come, or at least not easily, and where it is essential to intervene rapidly. However, in disaster-stricken areas, the communication network is usually damaged. Therefore, studies are looking at ways of optimizing communication between the victims and the rescuers, compensating for poor or even nonexistent networks [29, 35]. But generally, these solutions rely on the fact that disaster victims carry communication devices. So, it is necessary to multiply the ways of detecting people in danger. For that reason, the use of UAV embedded cameras is a promising alternative. They can be used by humans to determine areas where victims are more likely to be. But even if relying on images rules out the need for communication devices, transmitting images is bandwidth-consuming.

In order to effectively decrease bandwidth consumption, high-resolution images acquired by UAVs can be compressed, which reduces their size. But with compression, the image quality is affected, and the decision-makers may no longer be able to reliably distinguish people in danger within the image. Furthermore, it takes time to review large-scale images because scanning them to find small objects is a tedious task.

To overcome this challenge, hybrid annotation stands out as a useful method to improve the rapidness, exactness, and reliability of the data interpretation. This annotation is given by combining the predictions of an automated object detector and human annotators. This process accomplishes two objectives: (1) it allows rescuers to make fast and accurate decisions, and (2) it preserves valuable time for

human expert annotators.

However, hybrid annotation remains inefficient in areas where data transmission is not reliable. If the network does not guarantee continuous and high-speed transmission, UAV-to-ground communication is subject to delay. Thus, the decision-making process is slowed down, which is not conceivable in emergency scenarios.

Goals

In this thesis, a hybrid annotation framework using a scalable codestream for UAV bandwidth-restricted object detection is presented. It extends the hybrid annotation workflow by incorporating the JPEG 2000 compression algorithm [44].

The proposed approach employs a fine-tuned deep learning network for initial annotation of the image and uses the JPEG 2000's scalable codestream to selectively enhance the image resolution in areas that require human expert annotation.

The aim of this work is therefore to present the hybrid annotation framework and study the improvements it brings over a conventional hybrid annotation process.

Structure Of The Work

The work is structured as follows:

- Chapter 2 begins with an overview of the YOLOv8 object detection architecture. This is followed by an explanation of the JPEG 2000 compression standard to help understand how it works and how the standard is useful in the context of this thesis. Finally, the concept of active learning is explained.
- Chapter 3 gives a non-exhaustive overview of related works. It defines the four challenges to which the proposed framework seeks to respond and presents the state of the art in these challenges.
- Chapter 4 aims at presenting the baseline and proposed hybrid annotation frameworks as well as an active learning framework designed for on-the-job reinforcement.
- Chapter 5 provides a brief overview of the data and the metrics considered. Then, the experimental setups used to evaluate model performance are explained.

- Chapter 6 includes the results obtained with the Pléiades dataset and a discussion about them.
- Chapter 7 provides the conclusion of the work carried out during the thesis.

Chapter 2

Fundamentals

In this chapter, several important concepts are introduced in order to understand the rest of the thesis. First of all, the You Only Look Once deep learning models are presented (see Appendix A.1 for a basic introduction to machine learning and computer vision). Then, the functioning of the JPEG 2000 compression algorithm and its key features are addressed. Finally, the concept of active learning is explained.

2.1 The You Only Look Once Models

To automate the detection of objects of interest, deep learning (DL)-based convolutional neural networks are used. They allow interpreting input images to predict the locations and classes of objects within it.

Two types of object detection network can be distinguish: one-stage and two-stage networks.

Two-Stage Network

Two-stage models are composed of two neural networks [10], one responsible for extracting the regions where objects are potentially located and a second network for classifying each proposed bounding box. For example, the R-CNN architecture and its variants can be mentioned [21, 49].

Typically, this type of model achieves highly accurate object detection, but is slower than single-stage networks.

One-Stage Network

One-stage models are composed of only one neural network that handles both tasks. The You Only Look Once (YOLO) architecture was firstly introduce by Redmon et al. [38] in 2016, and was a new approach to object detection. They proposed a single unified neural network able to predict both bounding boxes and class probabilities in one evaluation of the input image [5].

The YOLO compact architecture is extremely fast compared to two-stage detectors that need two passes through the image to detect and then classify the objects. The single architecture makes one-stage detectors more suited for real-time applications, since the forward propagation is performed faster [8].

In addition, this gain in inference time is achieved without losing too much in performance compared to other state of the art detection models, including Fast R-CNN [26].

Furthermore, thanks to the unified architecture, not only the inference is improved, but also the training, which is facilitated. Indeed, only a single network has to be optimized, whereas with two-stage detectors, each component has to be trained independently for both tasks.

YOLO Architecture

Generally speaking, YOLO models are deep convolutional neural networks that are composed of two main parts:

- **Backbone**: it constitutes the main body of the network. It includes the initial convolutional layers of the architecture, which are responsible of the feature extraction, and creating the abstract representation of the input image. The backbone of YOLOv8 uses a modified CSPDarknet53 [39]. The backbone includes a Spatial Pyramid Pooling Fast (SPPF) module allowing to handle various scale, size, and aspect ratio [19].
- **Head**: it is the fully-connected part of the network that generates the output of the model, the bounding box coordinates and the associated classes. The head is actually composed of two parallel heads, one for object classification and the other for predicted bounding box regression.

Early versions of the YOLO algorithm divided the input image into a grid of cells and used predetermined anchor boxes to generate the output bounding boxes. The objective of these anchor boxes was to capture the scale and aspect of object classes

of training samples. These predefined boxes were then positioned across the image and to each predicted bounding box was assigned a class with a confidence score. In the end, only the predictions with confidence scores exceeding a threshold were output by the model.

The problem with this approach is that, the model struggles with small objects appearing in groups. To overcome this, the latest versions have therefore changed and are now anchor-free.

The input is not anymore divided into a grid of cells and the bounding boxes are predicted without relying on predefined anchor boxes. The operation is performed by the parallel heads, based on Task-aligned Head (T-Head) module which unifies the two tasks performed by the two head branches [16]. The module allows to align the anchors from the object classification and location tasks.

In this work, the last release of YOLO from Ultralytics, YOLOv8[1] is used. In Figure 2.1 the architecture of the last release is shown in-depth. The version has been developed by taking the best from previous iterations, and by introducing new modifications to further enhance detection speed and accuracy.

It exists five derived object detection models: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, ranked in ascending order of number of trainable weights. These derived models differ in convolutional filter size (changing the receptive field size) and network depth. In general, smallest networks give lower performance, but are faster in terms of training and inference. On the other hand, larger models are more resource-demanding and time-consuming, but offer better performance. It has been chosen to work with the derived model YOLOv8s, which is not too time and resource consuming thus offering a good trade-off.

Training Particularities

As well as offering good performance and fast inference, YOLO models family offers a number of embedded features for training.

In particular, YOLO integrates adaptive learning rate [41], dropout [45], or even extensive data augmentation techniques [52], which help to reduce overfitting and improve generalization capacity. Overfitting occurs when the model learns too much specific characteristics of the training set and is no longer able to predict new inputs reliably.

[1] `https://docs.ultralytics.com`

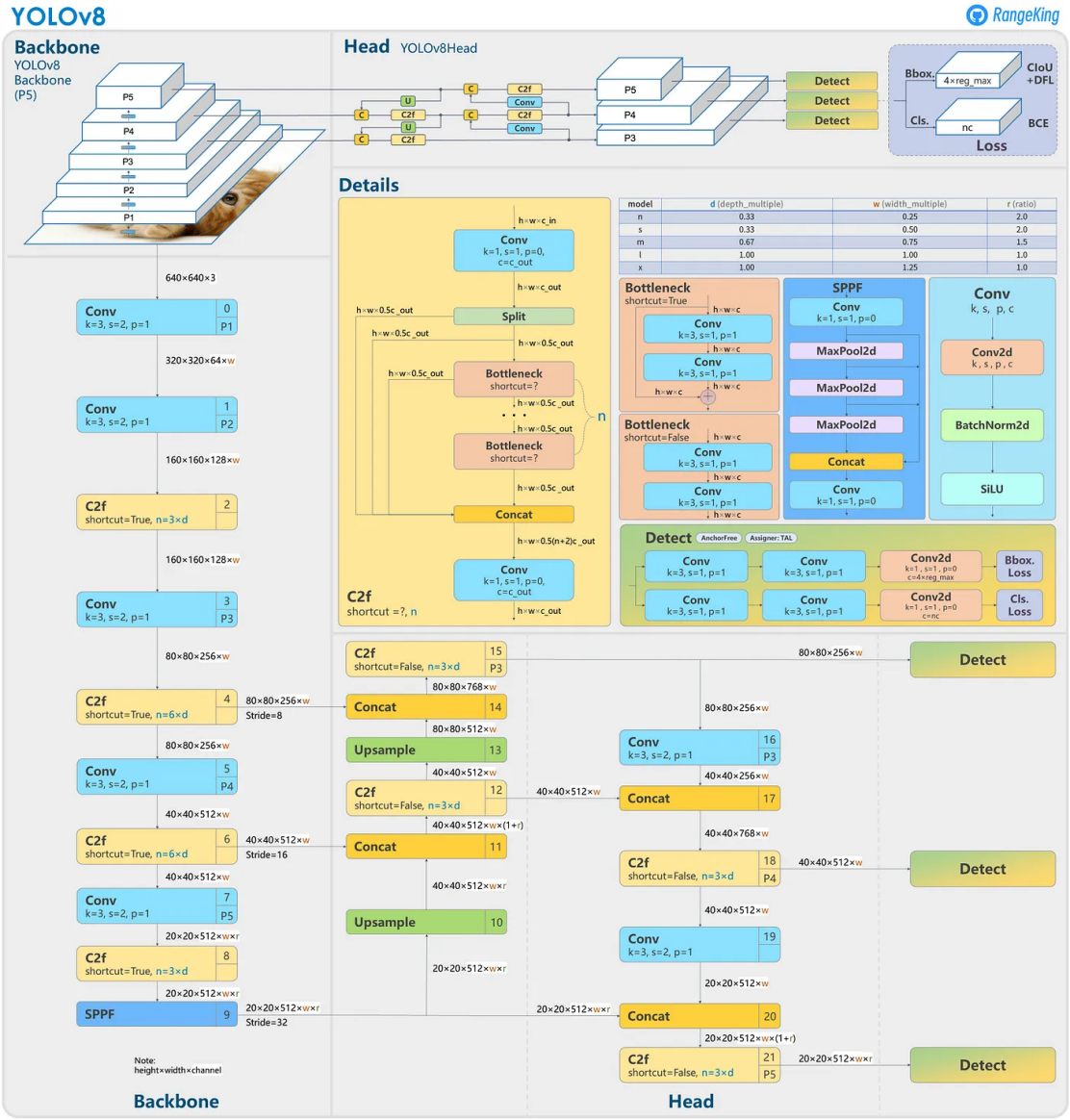

Figure 2.1: YOLOv8 detailed architecture[2].

For that reason, YOLO incorporates a pre-processing step allowing the modification of the training data to generate new samples and thereby enlarge data variations. The data augmentation techniques used include the saturation and exposure modification, horizontal flipping (mirroring), affine transformations (rotation, scaling, translation, and shearing), mosaic creation, and more.

[2] https://github.com/ultralytics/ultralytics/issues/189

During the training, a loss function is optimized. Loss functions are used to quantify the difference between the model's predicted outputs and the expected outputs from human annotation. As YOLOv8 no longer relies on dividing the input image into a grid of cells, the loss has changed compared to previous versions, now resulting from the combination of two elements:

- The **class loss** is responsible of measuring the error for the classification task, penalizing predictions for which the assigned classes are not correct.
- The **location loss** is responsible of measuring the error of location of an object, verifying whether the predicted bounding boxes are aligned with the ground truths.

In equation form, the composite loss function is written as

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{loc}, \tag{2.1}$$

where λ_1 and λ_2 are factors used to balance the losses differently, allowing to favor or mitigate a loss depending on the user needs.

The class loss is based on a multi-class Binary Cross-Entropy (BCE) loss [9], which is commonly used for classification tasks.

While, the location loss uses a Complete IoU (CIoU) loss [54], which is a variant of the classical IoU loss that do not only consider the overlap rate [17], combined with a Distribution Focal Loss (DFL) [28].

2.2 The JPEG 2000 Compression Standard

The standard considered to compress the high-resolution still images is the JPEG 2000 standard [44]. It represents a powerful compression tool that is optimized for efficiency, scalability, and interoperability in network and mobile environments. This standard has become a must and although it dates back to the turn of the century, it is still widely used today. JPEG 2000 provides a set of features that meet the needs of the new technologies, for example: superior low bit-rate performance, continuous-tone and bi-level compression, lossless and lossy compression, progressive transmission by pixel accuracy and resolution, region-of-interest coding, etc.

The choice fell on the JEPG 2000 compression technique for two main reasons. Firstly, even though its aim is to reduce the size of data transmitted as much as

possible, it is necessary to retain clear images that can be interpreted by humans or neural networks. Although the JPEG 2000 standard does not show the best performance at high bit rate, this is not the case at low bit rate, where it outperforms other wavelet domain-based methods [22]. Secondly, the proposed framework takes advantage of the progressive and hierarchical compression and decompression of the JPEG 2000 codestream.

JPEG 2000 Algorithm Operation

The JPEG 2000 algorithm block diagram comprising the encoding and decoding phases is illustrated in Figure 2.2. The original image is compressed via (1) discrete wavelet transform, (2) quantization, and (3) entropy coding blocks. The resulting codestream is arranged in such a way as to facilitate decoding, with a main header at the beginning of the codestream describing the image and its coding styles. Then, this header is used to locate, extract, decode, and reconstruct the image according the chosen criteria (e.g. number of decomposition levels, region-of-interest, number of quality layers, etc.). In order to retrieve the image, the codestream is (4) entropy decoded, (5) dequantized, and (6) inverse discrete wavelet transformed.

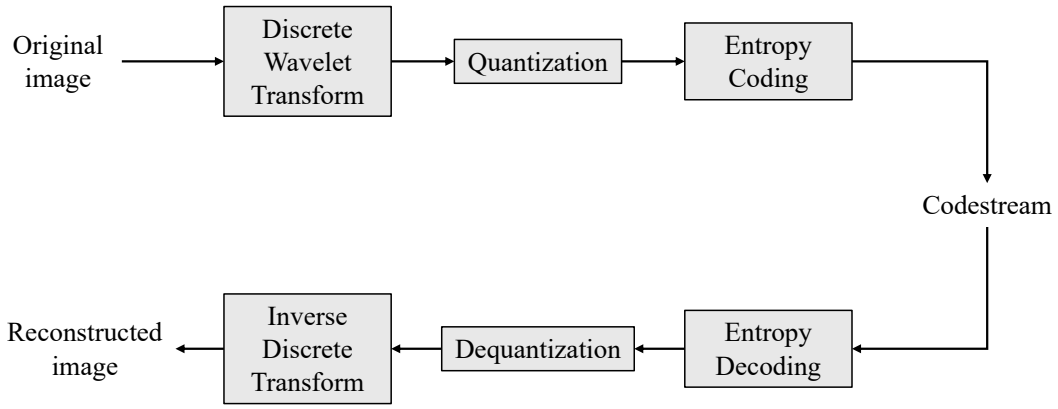

Figure 2.2: JPEG 2000 algorithm block scheme: compression and decompression parts.

The JPEG 2000 compression relies on 2D Discrete Wavelet Transform (DWT) [27], achieving multi-resolution image description. The original image is high-pass and low-pass filtered, yielding three sub-bands describing the details (LH, HL,

and HH) and one approximation sub-band (LL), respectively. These sub-bands consist of coefficients describing the vertical and horizontal frequency components of the image. Figure 2.3 shows one level of the 2D wavelet decomposition. Then, the approximation LL acts as the original image and can be further decomposed, depending on the number of decomposition levels desired. If several decomposition levels are used, the block diagram of Figure 2.3a is re-applied on the LL sub-band and the LL tile of Figure 2.3b is further divided.

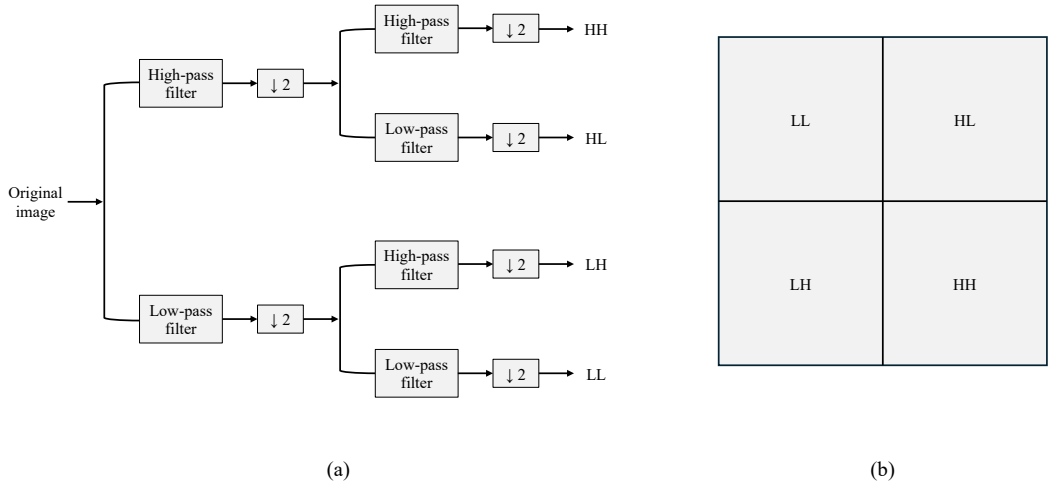

Figure 2.3: 2D discrete wavelet transform: (a) the block diagram of filter analysis and (b) the arrangement of sub-bands.

The quantization is the operation used to reduce the precision of the coefficients. The amount of data required to represent a coefficient is thus decreased, reducing data size. It might be a lossy operation, since some values are approximated. This operation is characterized by the quantization step-size, which describes the interval between two coefficients within the range of possible values. When the parameter is unitary, the compression is lossless, which is considered for the entirety of this work. The operation is written as

$$q_b(u,v) = sign(a_b(u,v))\lfloor \frac{|a_b(u,v)|}{\Delta_b} \rfloor, \tag{2.2}$$

where $a_b(u,v)$ are the coefficient of sub-band b, Δ_b is the quantization step-size, and $q_b(u,v)$ are the quantized values. It can be noted that each sub-band can have a different step-size.

JPEG 2000's entropy coding uses arithmetic coding and applies it to code-blocks, representing the smallest subdivision of the sub-bands. The probability of occur-

rence of a given bit is determined from the contextual information, obtained during three passes over the code-blocks: the significance propagation, the magnitude refinement, and the clean-up pass. This generates a separate bit-stream for each code-block, which are then arranged in packets and layers, each having their own header and constituting the codestream. The arrangement of the JPEG 2000 codestream makes it easy to manipulate, and thus allows to progressively and hierarchically access the information of an image.

Scalability And Codestream Formatting

There are two features that make JPEG 2000 a relatively interesting compression standard for the proposed hybrid annotation framework. Indeed, the annotation process takes advantage of the possibility of accessing several resolutions and regions of interest, both without the need to decode the entire codestream.

More precisely, the ability to encode images at multiple resolution levels simultaneously is called scalability. The coded bit-stream is arranged in a specific manner simplifying the decoding of the complete image at the different resolutions. To retrieve the low-resolution image, only a small subset of the bit-stream is decoded, allowing the progressive decoding of the image in the context of continuous transmission. In order to recover higher resolutions, the rest of the bit-stream can be decoded, until the end of the bit-stream is reached, which results in the image in its original resolution. This can be also used in a bandwidth-restricted context, where the low-resolution part is sent and decoded for fast manipulation, while the rest of the bit-stream is arriving limited by the data rate of the channel.

During the encoding process the image is divided into smaller non-overlapping blocks, called tiles. Then, all encoding and decoding operations are applied independently on each tile.

By applying all operations independently, the JPEG 2000 algorithm offers error resilience. For example, if bit errors happen inside a tile during encoding, only the part of the codestream corresponding to the tile is impacted. The error is not propagated.

But above all, the codestream is arranged in such a way as to allow facilitated access to the tiles without the need to decode entirely the bit-stream. The header indicates which parts have to be decoded to retrieve the desired tiles.

By combining both features, the JPEG 2000 compression algorithm offers the ability to encode an image at several resolution levels. Then, when the codestream

has to be decoded, it is possible to have direct access to regions of interest at the desired resolution.

It is interesting to note that, for the rest of this work, the resolution levels correspond to the number of DWT decomposition levels not discarded. At compression time, the images have been decomposed on five 2D wavelet decomposition levels. The resolution 5 means keeping all the DWT decomposition levels at decompression, resulting in recovery of the original image, as quantization is neglected (unitary quantization step-size). Level 4 corresponds to discarding one decomposition level, the mostly detailed coefficients, and so on until level 1, leaving only a coarse image, the latest decomposition. Figure 2.4 shows an image decomposed by applying two times the DWT decomposition, leading to six details sub-bands and one approximation.

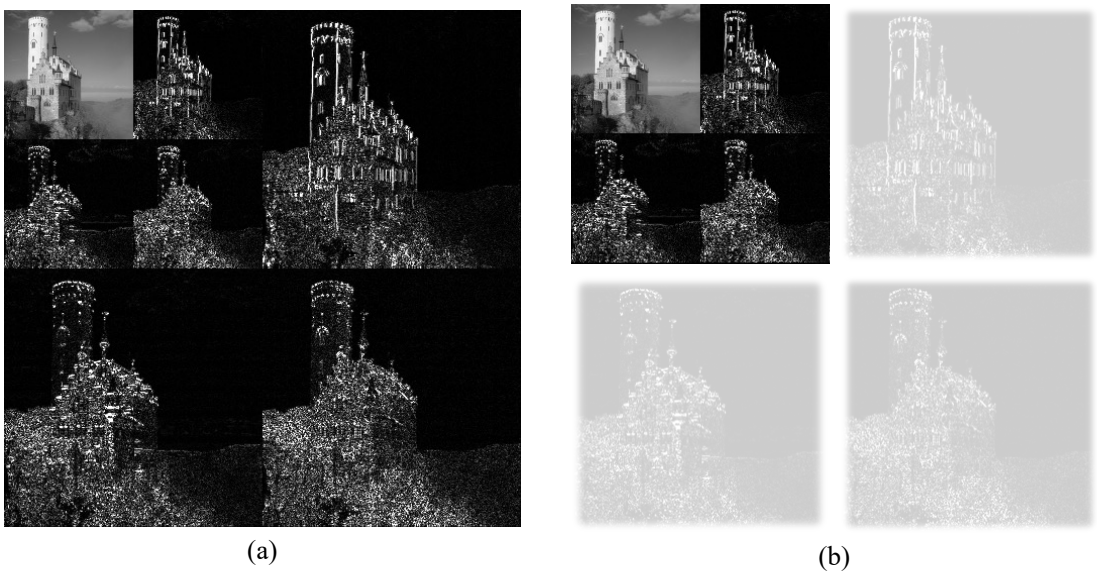

Figure 2.4: Image decomposed in 2 levels of DWT. (a) Image in resolution level 2 and (b) image in resolution level 1.

In this work, in order to compress and decompress the still images, the OpenJPEG 2.5.0 released library[3] has been used, which is an open-source JPEG 2000 codec written in C language.

[3] `https://www.openjpeg.org`

2.3 Active Learning

The annotation of images, and especially large-scale images, is a costly and time-consuming task for human annotators. To reduce the number of annotated images, Active Learning (AL) stands out as a promising solution. In AL, the learning algorithm selects iteratively subsets of data to be human annotated from a pool of unlabeled data. The approach hopes to reduce the amount of data to be annotated while maintaining a desired level of performance.

Active Learning Functioning

Generally, the AL loop starts with an initial training of the ML model using a small set of labeled samples, annotated by human experts. Then, a query strategy is used to select from the pool of unlabeled data which are the most relevant samples to annotate. These examples are chosen on the basis of a criterion that asks the machine learning model which ones are the most pertinent to it. The selected data is annotated by the oracle and then provided to the model for re-training. The steps are repeated several times until the model has reached a desired level of performance. Figure 2.5 represents schematically the approach.

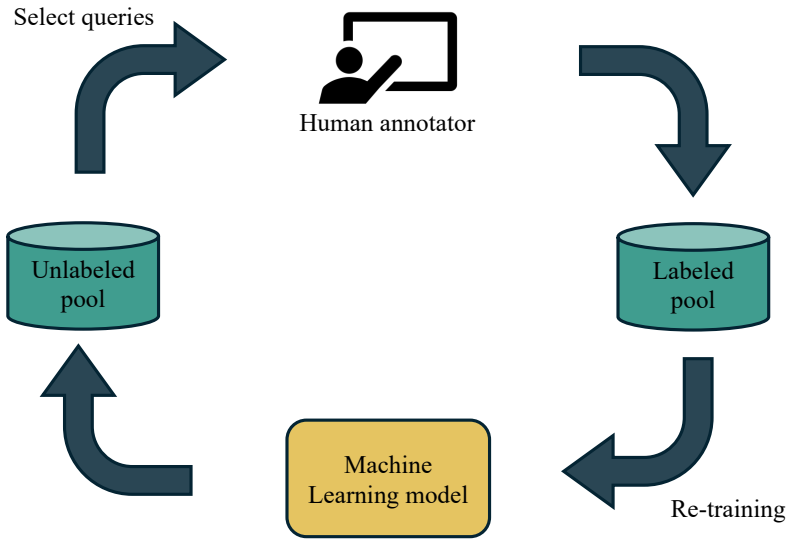

Figure 2.5: Generic Active Learning loop.

Query Strategies

It exists two major query strategies to select the most informative data, which are:

- **Uncertainty-based approach:** the predictions with the highest level of uncertainty are selected first [51]. It allows to annotate samples for which the model has difficulty in predicting with confidence, and train on it.
- **Diversity-based approach:** the samples are selected to ensure a diversity among the data to be annotated [24]. It is expected to enhance the model capacity to generalize to each class specific feature.

Both come with their benefits and disadvantages, hence an **approach combining uncertainty and diversity** can be used [20, 47, 56]. Sometimes, when relying on uncertainties, the selected samples show feature redundancy, reducing the efficiency of the method. Therefore, the idea is to introduce diversity among the uncertain selected samples.

Active Learning Advantages

First, AL allows to reduce the number of labeled data, limiting the need for a large number of samples to train deep learning model. Only the most relevant data is annotated, redundant examples are not used. This saves valuable human annotator experts time, who do not have to annotate every images in the dataset.

Then, models trained with an AL approach frequently produce high accuracy and have better generalization capability [50]. They converge faster to their optimum performance.

2.4 Chapter Conclusion

In this chapter, three tools used by the proposed framework were presented. These explanations serve as a reminder to understand the rest of the thesis.

The YOLOv8 deep learning model is responsible for detecting objects of interest. It represents a cutting-edge object detector particularly suited for applications requiring rapid inference. In the literature, it has been proven that the model offers high performance for computer vision tasks.

The JPEG 2000 algorithm is used to compress high-resolution UAV images. This codec offers a great image quality at low bit rate. In addition, its codestream

arrangement and scalability allow easy access to parts of images at different resolutions.

Finally, active learning techniques are used to reduce the amount of labeled data by selecting only the samples that really allow the improvement of machine learning model.

Chapter 3

Related Works

In this chapter, the works related to the proposed hybrid annotation framework for bandwidth-restricted UAV object detection are presented. The framework seeks at addressing different challenges, which are:

(I) High-resolution image compression,

(II) DL-based object detection,

(III) Hybrid annotation,

(IV) Applicability to bandwidth-restricted UAV scenarios.

The framework uses compression for transmission of large-scale and high-resolution images, identified as **challenge (I)**. As drones offer increasingly high-resolution images and videos, compression is essential to offer transmission in real-time or with insignificant delay.

Then, the object detection task has to be automated, and with **challenge (II)**, the aim is to rely on DL-based model to accurately detect objects of interest in images.

But in emergency situations, totally relying on an automatic model to detect people in danger is not an option. The automated part of the system is designed to assist rescue workers in their mission. Image annotation is therefore a combination of DL-based models and humans, an approach identified as **challenge (III)**.

Finally, with **challenge (IV)**, the objective is to be able to deploy the system in emergency situations, where the bandwidth is limited, while ensuring a desired level of performance.

Works Related To (II)

The proposed framework relies on a DL-based object detection model to perform an initial annotation of the still images. It allows automating the work usually done by human annotator experts, saving them valuable time by directly selecting the most interesting areas of the image.

Wang et al. [48] studied the rapidness and exactness of the prediction of CNN-based object detectors. They observed that deep learning models perform better than other models for computer vision tasks on UAV images. Their experiments validate the idea of using DL-based models for object detection tasks, but they did not consider the constraints of the environment. They assumed that their communication was bandwidth-unlimited, so they proposed the use of image compression to reduce the computation workload and bandwidth consumption.

Budiharto et al. [4] developed a fast DL-based object detector for quadcopter drone cameras with low-specification in hardware, and they demonstrated that the system maintains a high-accuracy detection performance on 14 FPS videos, while other models only go up to 6 FPS. The system was tested in an ideal situation where bandwidth was not limited, as is the case in this work.

Ammour et al. [3] tackled the challenge of detecting and counting cars with UAVs, and they proposed an automatic method using deep learning to solve the problem. They demonstrated that their method outperforms other techniques in accuracy and computational time. They used a combination of a deep CNN and a Support Vector Machine (SVM) classifier to reduce the search space and extract descriptive features for accurate and fast classification. The method looks very promising, but it has only been tested on five different images. In addition, they did not work with compressed images; the cars they are trying to count have an average size of 200×90 pixels which is much larger than the objects in highly compressed images. Finally, the method is not particularly suitable for emergency applications with large-scale images, as the inference time for an image of size 2626×4680 is about 30 minutes with their method.

In these three contributions, the use of DL-based models is highlighted for object detection (II), but without taking into account the constraints imposed by the environment, and therefore without using image compression.

Works Related To (I) And (II)

The DL-based model does not directly predict on UAV high-resolution images. Instead, the model has to manipulate low-resolution samples since images are compressed, and only the part of the codestream corresponding to lower resolution is kept and transmitted. This operation can have a negative impact on images, creating compression artifacts and making model detection more difficult. Indeed, distinctive class features are lost or attenuated, and small objects become even smaller, which reduces performance.

Ehrlich et al. [12] studied the effects of JPEG compression on deep neural networks' performance for common computer vision tasks and on a various range of datasets. In addition, they also introduced a self-supervised artifact correction approach that improves object detection results. They observed a huge drop in performance due to high-compression ratio, and a diminution in performance of 14% was observed for the object detection task when working with compressed data.

It is also possible to modify the object detection model to facilitate the detection of small objects that are common in large-scale UAV images. Liu et al. [31] have optimized a method based on YOLOv3 for small object detection, enlarging the receptive field by enriching the spatial information in the early stage of the convolutional network.

From the work conducted by Ghazvinian Zanjani et al. [18], it is noted that training the model with compressed images mitigates the impact of compression on YOLO detection ability. They studied the impact of JPEG 2000 compression on CNN-based detectors for medical images, on which compression cannot affect the diagnosis. With their experiments, they observed that classification is relatively robust to compression up to a certain ratio when the model is trained only on high-resolution. In addition, they observed that giving compressed examples during training enhances the model's performance when classifying at the same resolution.

The latter observation is in line with that made by El Khoury et al. [13]. They compared the robustness of 2D versus 3D U-Net against JPEG 2000 compression for male pelvic organ segmentation and improved the compression tolerance using fine-tuning. They observed that fine-tuning the network with images compressed at the same ratio as the network has to handle helps it adapt to compression effects and maintain high-performance. Moreover, El Khoury et al. [15] iteratively fine-tuned a 3D U-Net for improving segmentation against JPEG 2000 compression to reach high-compression ratios. Recently, El Khoury et al. [14] proposed a DL-based object detector and tracker working exclusively on residual frames, ensuring data

privacy and reducing the storage costs.

To overcome bandwidth-restricted networks, Chamain et al. [6] proposed a deep-learning based classifier trained on JPEG 2000 encoded images, achieving faster and accurate predictions.

Thanks to these contributions, it is evident that compression impacts the performance of DL-based models for object detection. However, this decrease can be mitigated by preparing the model for the artifacts introduced by compression, making it more robust at high-compression ratios. In all of these contributions, there is no hybrid annotation workflow considered.

Works Related To (I) And (IV)

Indradjad et al. [22] compared four wavelet-based compression methods, and they demonstrated that the JPEG 2000 algorithm image quality outperforms the other codecs at low bit rates.

This observation corresponds to that of Zhou et al. [55]. Indeed, they gave an overview of remote sensing image compression algorithm adopted in satellite systems. They also highlighted superior preservation of image quality at low bit rate.

Both contributions support the use of JPEG 2000 for UAV transmission when bandwidth is limited.

Works Related To (I), (II), And (IV)

Preethy Byju et al. [37] proposed an approach for classifying scenes in JPEG 2000 compressed remote-sensing images using a DL network. The method is interesting because it enables classification to be carried out on images that have not been fully decompressed while maintaining classification accuracy. This reduces the computation time required for image decompression.

Zhang et al. [53] proposed a real-time end-to-end transmission framework based DL in UAV-aided networks. The mechanism compresses the data into latent codes, allowing fast transmission while maintaining the quality of visual data for small objects despite bandwidth constraints and compression artifacts. But in their work, it was assumed to have uninterrupted line-of-sight communication, thereby limiting its applicability in complex environments.

The two papers present approaches based on DL model to manipulate UAV compressed images in bandwidth-limited context. But they do not incorporate a hybrid annotation framework.

Works Related To (II), (III), And (IV)

Yamani et al. [51] proposed an active learning framework for single-stage object detection in UAV images. Their framework selects the images with a wide variety of classes with high uncertainties while mitigating the class imbalance. Single-stage selection process is convenient since it selects the most informative samples directly based on the output uncertainties of the object detection model while, two-stage AL techniques dissociate the object detection step and the selection of the most relevant images. They demonstrated that their AL framework needs less training samples to reach equivalent performance than using the whole training dataset.

Kellenberger et al. [24] presented an AL strategy integrating a deep CNN object detector as criterion. Their research focus on the detection of wild animals in UAV acquired images. Instead of using model uncertainty to select the training samples, their strategy was to use the CNN scores to rank the samples and choose very confident and typically rare samples. Their experiment have shown that using less than half of labeled images is enough to retrieve 80% of animals in challenging sets.

Both contributions show AL-based frameworks responsible of detecting objects in UAV acquired images but they do not take into account image compression required in bandwidth-restricted scenarios.

3.1 Chapter Conclusion

In this chapter, the four challenges addressed by the proposed framework were presented. Then, a non-exhaustive list of works relating to these challenges has been established.

Among all the articles, the best one found meets at most three of the four challenges at once.

The proposed streamlined hybrid annotation framework using scalable codestream for bandwidth-restricted UAV object detection is the first contribution that answers all four challenges simultaneously. Table 3.1 synthesizes the challenges addressed by the related works.

	[3]	[4]	[6]	[12]	[13]	[14]	[15]	[22]	[24]	[31]	[37]	[48]	[51]	[53]	[55]	Ours
(I)			✓	✓	✓	✓	✓	✓			✓			✓	✓	✓
(II)	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		✓
(III)									✓				✓			✓
(IV)								✓	✓		✓		✓	✓	✓	✓

Table 3.1: Summary of related works addressing the four challenges.

Chapter 4

Hybrid Annotation Frameworks

In this chapter, a conventional baseline hybrid annotation strategy and the proposed hybrid annotation framework are developed. Then, the active learning functioning of system is explained. System blocks are developed in order to understand how they work and how they are implemented.

Chapter Notations

In this chapter, several variables are used to describe the frameworks, therefore Table 4.1 summarizes these notations.

Variable name	Description
I^R	Still image at resolution R
T_m^R	m-th tile at resolution R
LR	Low-resolution, highest level allowed for transmitting all tiles
HR	High-resolution
$Rlvls$	List of resolution levels considered
$\mathscr{M}$	Set of tile indices
$\mathscr{N}$	Set of tile indices to be annotated
BB_{DL}	Model predictions (bounding boxes and classes)
CS_{DL}	Model confidence scores
BB_{HUM}	Human predictions (bounding boxes and classes)
CS_{HUM}	Human confidence scores
$data\_rate$	Data rate [kbps]
$t_{TRlimit}$	Transmission time limit [s]

Table 4.1: Notations used for describing the frameworks.

4.1 Baseline Hybrid Annotation Framework

The baseline framework defined as reference is emulating the standard hybrid annotation approach used by UAVs in emergency situations. It translates a naive way of transmitting images from UAVs to ground station. The entire high-resolution image is sent without taking into account the surrounding environment. Figure 4.1 represents the block diagram of the baseline framework.

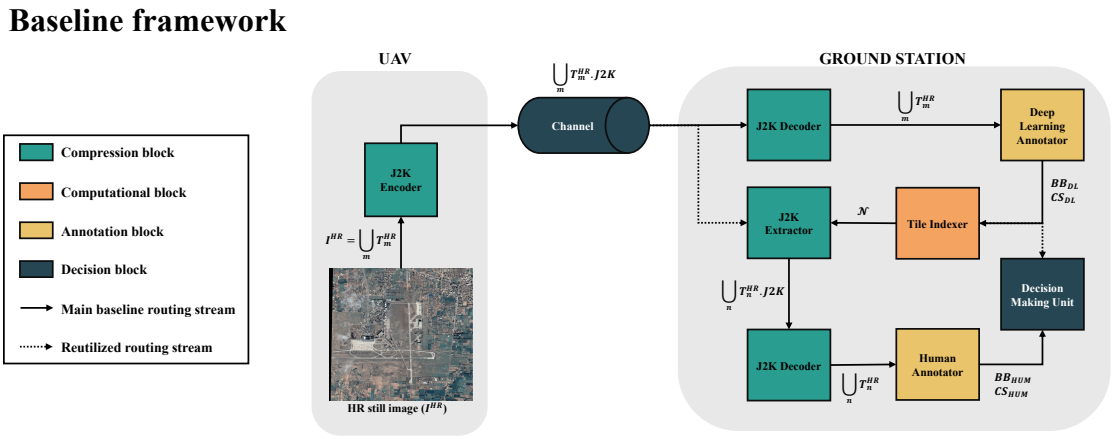

Figure 4.1: Block diagram of the baseline hybrid annotation framework.

Operating Procedure

The routine from the high-resolution still image acquired by the UAV to the ground station where the decision is made can be summarized as follows:

1. The high-resolution still image I^{HR} is acquired by the UAV. The whole image can also be seen as a grid of tiles, mathematically written as $I^{HR} = \bigcup_m T_m^{HR}$.

2. The *J2K Encoder* compresses the whole image, meaning all tiles, using the compression algorithm JPEG 2000. In the context of this work, the image is compressed using five DWT decompositions. The encoding function is written as
$$\bigcup_m T_m^{HR}.J2K \leftarrow J2K(\bigcup_m T_m^{HR}, \mathscr{M}), \tag{4.1}$$
where $\mathscr{M} = \{1, 2, ..., |\mathscr{M}|\}$ is the set of tile indices constituting the image.

3. The whole high-resolution compressed image is sent through the lossless communication channel to the ground station.

4. At the reception, the compressed image is decompressed at the highest resolution level HR, by keeping all DWT decomposition levels. It allows to recover the original image I^{HR}. The JPEG 2000 decoding is given by

$$I^{HR} = \bigcup_m T_m^{HR} \leftarrow J2K^{-1}(\bigcup_m T_m^{HR}.J2K, \mathscr{M}). \tag{4.2}$$

5. The image is provided to the *Deep Learning Annotator* block which uses the model trained to detect objects of interest. It outputs the labeled bounding boxes, denoted as BB_{DL}, and the confidence scores CS_{DL} that provide the degree of certainty of each prediction. The predictions are given by

$$BB_{DL}, CS_{DL} \leftarrow DL(\bigcup_m T_m^{HR}, \mathscr{M}, HR). \tag{4.3}$$

6. The *Tile Indexer* block selects the tiles that need the human intervention based on the output of the deep learning model. Tiles are chosen according to the confidence score, the lowest values are selected first. The number of tiles selected is constrained by the human annotator budget, denoted as $|\mathscr{N}|$. The block outputs the indices of tiles requiring annotation as

$$\mathscr{N} \leftarrow IND(BB_{DL}, CS_{DL}, |\mathscr{N}|). \tag{4.4}$$

7. The tiles to annotate are extracted from the compressed image using the *J2K Extractor* block at the highest resolution level as

$$\bigcup_n T_n^{HR}.J2K \leftarrow EXT(\bigcup_m T_m^{HR}.J2K, \mathscr{N}, HR). \tag{4.5}$$

8. These selected tiles are then decompressed tanks to the *J2K Decoder*, which are written as

$$\bigcup_n T_n^{HR} \leftarrow J2K^{-1}(\bigcup_n T_n^{HR}.J2K, \mathscr{N}). \tag{4.6}$$

9. The *Human Annotator* corrects the predictions on the tiles supplied to him, producing the bounding boxes BB_{HUM} and the confidence scores CS_{HUM}. He can suppress bounding boxes that are not correctly localized, change the class, or even add new predictions. These predictions are written as

$$BB_{HUM}, CS_{HUM} \leftarrow HUM(\bigcup_{n} T_n^{HR}, \mathscr{N}) \tag{4.7}$$

10. Finally, the decision is taken based on the predictions from the deep learning annotator corrected by the human annotator,

$$Decision \leftarrow BB_{HUM}, CS_{HUM}, BB_{DL}, CS_{DL}. \tag{4.8}$$

The framework presents problems due to the continuous transmission of high-resolution images. It does not take into account the constraints imposed by the channel and the user.

If the data rate is low, it will take longer time to transmit the entire image, and decision-making time will be also longer. The same goes for short decision-making time where it cannot be afforded to wait for the entire high-resolution image to be transmitted.

4.2 Proposed Hybrid Annotation Framework

The proposed framework aims at reducing the overall bandwidth consumed by sending all the tiles of the image at high-resolution as well as the time interval between the acquisition and the end of the process.

In order to reach a rapid and reliable decision-making process, the baseline framework is adjusted and Figure 4.2 presents the modified hybrid annotation framework. It integrates four improvements that are presented in the following subsections.

Operating Procedure

The system works similarly to the baseline, but it integrates the constraints provided by the channel and the human annotators. The functioning is as follows:

Proposed framework

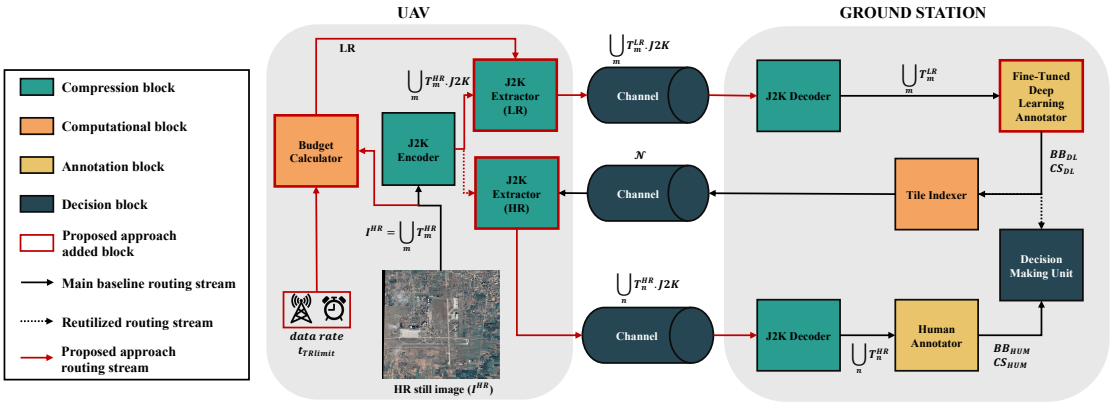

Figure 4.2: Block diagram of the proposed hybrid annotation framework.

1. The high-resolution still image I^{HR} is acquired by the UAV.

2. Based on the data rate imposed by the channel and the transmission time limit imposed by the application, the *Budget Calculator* computes the highest resolution allowed to send the entire image LR and the annotator budget $|\mathscr{N}|$. The block is further explained in Subsection 4.2.1.

   The chosen low resolution and the number of tiles are given by

   $$LR, HR, |\mathscr{N}|, |\mathscr{M}| \leftarrow BC(I^{HR}, size(T^{HR}), data\_rate, t_{TRlimit}, Rlvls), \tag{4.9}$$

   where $Rlvls = \{1, 2, 3, 4, 5\}$ in this work.

3. The *J2K Encoder* compresses the input image using JPEG 2000,

   $$\bigcup_m T_m^{HR}.J2K \leftarrow J2K(\bigcup_m T_m^{HR}, \mathscr{M}). \tag{4.10}$$

4. The part of the JPEG 2000 codestream corresponding to the low-resolution is extracted from the whole codestream, which gives

   $$\bigcup_m T_m^{LR}.J2K \leftarrow EXT(\bigcup_m T_m^{HR}.J2K, \mathscr{M}, LR). \tag{4.11}$$

The output corresponds to the whole compressed image at low-resolution. The step is further explained in Subsection 4.2.2.

5. The compressed image is sent through the lossless communication channel to the ground station.

6. At the reception, the whole codestream is decompressed, which allows to recover the low-resolution image I^{LR},

$$\bigcup_m T_m^{LR} \leftarrow J2K^{-1}(\bigcup_m T_m^{LR}.J2K). \tag{4.12}$$

7. The image is provided to the *Fine-Tuned Deep Learning Annotator* block which uses the model trained to detect objects of interest. The network is specialized for each specific resolution level, allowing high-performance despite the poor resolution. Further details are available in Subsection 4.2.3. As for the baseline, it outputs BB_{DL} and CS_{DL} as

$$BB_{DL}, CS_{DL} \leftarrow DL(\bigcup_m T_m^{LR}, \mathscr{M}, LR). \tag{4.13}$$

8. The *Tile Indexer* block selects the tiles that require human intervention based on the output of the fine-tuned deep learning model. The block operation is developed in Subsection 4.2.4. The indices are given by

$$\mathscr{N} \leftarrow IND(BB_{DL}, CS_{DL}, |\mathscr{N}|). \tag{4.14}$$

9. The indices of the tiles selected $\mathscr{N}$ are sent back to the UAV through the lossless channel.

10. The tiles to annotate are extracted from the compressed codestream, by keeping only the parts corresponding to the selected tiles. This time, the tiles are in high-resolution,

$$\bigcup_n T_n^{HR}.J2K \leftarrow EXT(\bigcup_m T_m^{HR}.J2K, \mathscr{N}, HR). \tag{4.15}$$

11. The compressed high-resolution tiles are sent back to the ground station for annotation.

12. The received codestream is then decompressed to recover the high-resolution tiles,
$$\bigcup_n T_n^{HR} \leftarrow J2K^{-1}(\bigcup_n T_n^{HR}.J2K, \mathscr{N}). \tag{4.16}$$

13. The *Human Annotator* corrects the predictions on the tiles supplied to him,
$$BB_{HUM}, CS_{HUM} \leftarrow HUM(\bigcup_n T_n^{HR}, \mathscr{N}). \tag{4.17}$$

14. Finally, the decision is taken based on the predictions from the fine-tuned deep learning model corrected by the human annotator,
$$Decision \leftarrow BB_{HUM}, CS_{HUM}, BB_{DL}, CS_{DL}. \tag{4.18}$$

4.2.1 Budget Calculator

The *Budget Calculator* is the first block called in the routine. It takes as inputs the scenario-specific constraints to determine the highest resolution level (LR) at which the entire compressed image can be sent to the ground station for neural network annotation. Algorithm 1 describes the functioning of the block.

To determine the outputs, it uses the constraints imposed by the channel condition, the data rate available $data\_rate$ and the needs of the application, the transmission time limit $t_{TRlimit}$. The transmission time limit is defined as the time interval from the acquisition to the human annotation, when all data has been transmitted.

Based on these two constraints, the bandwidth budget can be calculated, and is given by
$$BW = data\_rate\ [kbit/s] \cdot t_{TRlimit}\ [s], \tag{4.19}$$
quantifying the amount of data that can be transmitted.

Then, the *Budget Calculator*, starting from the highest resolution and working down to the lowest, checks each time whether the entire image can be sent at the considered resolution. The condition checked is
$$size(I^R) \leq BW, \tag{4.20}$$

where $size(I^R)$ is the size of the image at a given resolution R. The resolution that respects this condition is then denoted LR.

If the condition is respected, the block computes the number of tiles that can be sent at the highest resolution HR for human annotation with the remaining bandwidth budget available. This number is calculated as

$$|\mathscr{N}| = (BW - size(I^R))/size(T^{HR}), \tag{4.21}$$

where $size(T^{HR})$ is the size of a tile at high-resolution.

When the lowest resolution LR has been found and the annotator budget has been calculated, the process ends. But if the condition is never respected, that means that the constraints are too restrictive and transmission respecting the limitation is not possible, even at low-resolution.

Algorithm 1 Budget calculator

Require: $I^{HR}, size(T^{HR}), data\_rate, t_{TRlimit}, Rlvls$
  $LR \leftarrow -1$
  $|\mathscr{N}|, |\mathscr{M}| \leftarrow 0$
  $BW \leftarrow data\_rate.t_{TRlimit}$
  $HR \leftarrow max(Rlvls)$
  for $R \leftarrow HR$ **to** 1 **do**
    if $size(I^R) \leq BW$ **then**
      $LR \leftarrow R$
      $|\mathscr{N}| \leftarrow (BW - size(I^R))/size(T^{HR})$
      break
    end if
  end for
  if $LR = -1$ **then**
    Insufficient BW budget, must relax constraints
  end if
  $|\mathscr{M}| \leftarrow size(I^{HR})/size(T^{HR})$
  return $LR, HR, |\mathscr{N}|, |\mathscr{M}|$

Being in good transmission condition allows to send the whole image at higher resolution facilitating the model's work and thus gives better results, which implies that experts are less asked to correct the bad predictions.

In contrary, having a deteriorate channel does not allow to transmit the image at useful resolution, reducing the model performance. Then, the human annotators have to be more requested to correct the predictions but it is not possible due to the low remaining bandwidth budget. This can delay the decision-making or even lead to missed detection.

4.2.2 JPEG 2000 Extractor

The *J2K Extractor* block is responsible for extracting a part of the JPEG 2000 codestream corresponding to desired characteristics. Algorithm 2 shows the block's behavior.

Initially, it extracts the part of the codestream corresponding to the entire low-resolution compressed image. Then, based on the output of the *Tile Indexer* block, it extracts the part of the codestream corresponding to the selected tiles in high-resolution.

It takes as input a JPEG 2000 compressed image $\bigcup_x T_x^{HR}.J2K$, the list of tiles to be extracted $\mathscr{Y}$, and the resolution to be extracted R. The block outputs the set of desired tiles at the specified resolution $\bigcup_y T_y^R.J2K$.

To do that, it takes advantage of the progressive and hierarchical arrangement of the JPEG 2000 codestream, and based on header information, it reaches the desired parts without having to decompress the whole codestream, saving both time and computing resources.

Algorithm 2 J2K Extractor

Require: $\bigcup_x T_x^{HR}.J2K, \mathscr{Y}, R$
  for each $y \in \mathscr{Y}$ **do**
    Extract T_y^R from $\bigcup_x T_x^{HR}.J2K$ at resolution R
  end for
  return $\bigcup_y T_y^R.J2K$

Within the proposed framework, the block is situated on the UAV side, reducing the amount of data transmitted, since only the effectively useful parts are sent.

4.2.3 Fine-Tuned Deep Learning Annotator

The *Fine-Tuned Deep Learning Annotator* block is responsible for detecting objects on input tiles $\bigcup_m T_m^{LR}$, which are in low-resolution.

As compression can significantly impact image quality [12]. The performance of the DL-based model can be also impacted since it has not been trained on poor-quality instances.

To overcome this problem, five models are trained, each responsible for a specific resolution level. The model in charge of the highest resolution is initialized with the pre-trained weights. These weights are trained on COCO dataset [30], a huge dataset composed of 80 classes including planes and helicopters.

The lower levels are initialized with the weights from training at the higher level and then specialize through training for the resolution level. In other words, the resolution 4 model starts with the weights from the training of the model responsible for resolution 5 and so on, until the model handling resolution 1 images is initialized with the weights from resolution 2. Figure 4.3 shows the cascade fine-tuning approach considered.

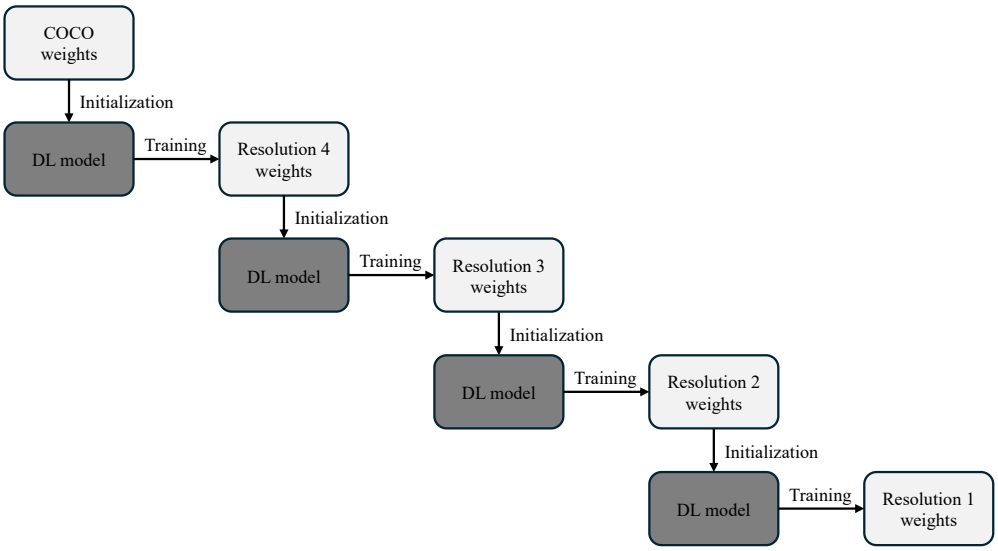

Figure 4.3: Cascade fine-tuning approach with 5 resolution levels.

Fine-tuning the deep learning models on low-resolution images allows to maintain optimal detection performance. The aim of fine-tuning is therefore to specialize the model in object detection at lower resolutions, allowing the model to adapt to compression artifacts.

4.2.4 Tile Indexer

The *Tile Indexer* block is responsible for selecting the tiles which will be sent to experts for annotation, based on a chosen criterion. It has been chosen to select the tiles based on the prediction uncertainty of the model [51].

At inference time, only the predictions with a high enough confidence score are kept and output by the model. The threshold that determines which predictions are kept is a hyper-parameter of YOLOv8 models. In this work, the confidence score threshold is set low, $CS_{threshold} = 0.25$, allowing non-certain predictions to be output, increasing the false positive rate. The model thus tends to predict much more objects. This choice has been taken to be sure to maximize the number of predictions, which hopefully reduces the number of missed objects. On the other hand, keeping uncertain predictions reduces the performance of the model but since predictions are refined by experts, it is expected that the overall performance is not as much impacted.

This block plays an essential role in the overall performance of the system. If tiles are not well selected, the human annotators will not be able to improve the performance by correcting errors. On the contrary, having an efficient *Tile Indexer* can help the system to improve the predictions of the DL-based model.

The block starts by sorting the bounding boxes in ascending order according to their confidence scores. Then, the tiles containing the bounding boxes with the lowest confidence scores are selected to be annotated by the experts. The number of selected tiles is determined by the budget calculator block that takes into account the physical constraints imposed by the environment. Algorithm 3 shows the pseudo-code of the block.

Algorithm 3 Tile Indexer

Require: $BB, CS, |\mathscr{Y}|$
  Sort BB w.r.t. CS values in ascending order
  $|\mathscr{X}|, i \leftarrow \emptyset, 0$
  while $i < |\mathscr{Y}|$ **do**
    $\mathscr{X}$.append($BB[i]$)
    $i \leftarrow i + 1$
  end while
  $\mathscr{Y} \leftarrow \mathscr{X}$
  return $\mathscr{Y}$

Even if the $CS_{threshold}$ is set low and the model outputs uncertain predictions, it happens that there is not enough tiles with predicted bounding boxes, meaning that the annotator budget is large enough to select more tiles. The block can be left as it is, thereby reducing the number of queries to annotators, who annotate less than they offer.

Alternatively, the block can be modified to use all the available budget $|\mathscr{N}|$. Therefore, two extensions of the *Tile Indexer* have been considered.

Random Approach

The first version corresponds to a naive approach where the tiles are randomly selected. It assumes that the detection made by the DL-based model is good, and almost no tile containing objects will be missed. Since, $CS_{threshold}$ is set very low, large-scale detection is favored. But if the available budget allows more tiles to be annotated than those already selected, the *Tile Indexer* block randomly selects from all the others, hoping to find a tile that has not been correctly predicted.

Majority Voting-Based Approach

The second version selects additional tiles based on a majority voting approach among the adjacent tiles. To determine the order of selection, the idea is to construct a list of adjacent tiles ordered according to a majority voting approach. A tile adjacent to several tiles in which predictions have been made has a greater chance of also containing objects, and these are therefore placed first in the list.

In Figure 4.4 an hypothetical example of the extension based on majority voting is presented. First, one can observe that 4 tiles have been selected for verification by human annotators because the DL model had predicted uncertain objects in these tiles. Then, the *Tile Indexer* extension takes all adjacent tiles and attributes a number to each one, corresponding to the number of already selected neighbors. Finally, the most voted tiles are chosen to be annotated, which led to the recovery of an additional instance.

4.3 Active Hybrid Annotation Framework

The proposed hybrid annotation framework assumes having at its disposal enough training samples to adapt to compression artifacts. But the problem is that at the start of a mission, the model may not have enough training images to ensure a desired level of performance. However, in emergency situations, it is inconceivable

Figure 4.4: Tile Indexer based on a majority voting approach.

to delay the mission. To meet this need, a third approach is envisaged.

Figure 4.5 presents the hybrid annotation framework adapted for active learning. It allows on-the-job reinforcement. The framework can adapt itself to compression artifacts at the beginning of its use.

The active hybrid annotation framework extends the proposed framework essentially by adding a routing stream between the *Human Annotator* block and the *Deep Learning Annotator*. As well as, a *Labeled pool* block responsible for storing the tiles annotated by human annotators.

One can observe that the *Tile Indexer* supplies to the *J2K Decoder* the indices of tiles to be annotated. It is important to note that the block does not have to re-decode the codestream, it can simply retrieve the tiles it decoded earlier in the routine, reducing computational cost.

Query Strategy

The AL query strategy used corresponds to the strategy of the *Tile Indexer*. So, in this case, it is an uncertainty-based approach. The annotated samples correspond to those with which the model has struggled.

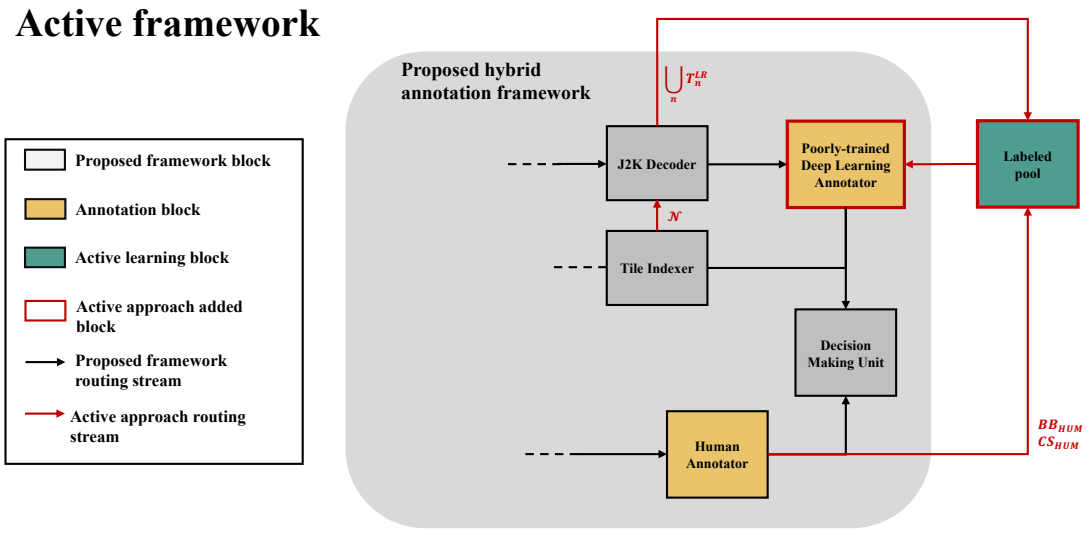

Figure 4.5: Block diagram of the active hybrid annotation framework.

It also uses the majority voting extension to select additional tiles in the prediction area if it is needed, which is often the case at the beginning of missions since the model has not been trained enough. It predicts with less confidence and in smaller quantities.

Labeled Pool

The *Labeled pool* block stores the annotations made by the human annotators and the corresponding tiles at low resolution LR. In fact, it does not store the high-resolution tiles that have been annotated by human experts, as the model must be trained with the low-resolution tiles to adapt to compression. It does not keep all the tiles, it only store the $|\mathcal{N}|$ tiles selected to be annotated.

It does not directly provide the poorly-trained DL model with samples for re-training. Instead, the block waits until it has received several annotated images or tiles before supplying the model. This prevents training too often. The training phase is time and resource consuming, so it has to be done as rarely as possible.

On the other hand, having frequent training allows to adapt to compression artifacts faster, the model quickly converges towards its optimum performance.

Thus, a trade-off must be made to ensure rapid adaptation while reducing resource

consumption.

4.4 Chapter Conclusion

In this chapter, a basic hybrid annotation workflow has been enhanced to obtain a framework designed for emergency situations. The improvements made were:

- Adding a *Budget Calculator* to take into account channel and user constraints;
- Taking advantage of JPEG 2000 codec to extract desired tiles at specified resolution R;
- Fine-tuning a YOLOv8 model to adapt to compression artifacts;
- Transmitting the whole image at low-resolution and only uncertain tiles at high-resolution for human annotation.

Finally, an active learning extension was presented, offering on-the-job reinforcement.

Chapter 5

Experiments

This chapter starts with a description of the data considered in this work to assess the framework, the Pléiades dataset. Then, the evaluation metrics are developed, providing their expressions and the justification for their use. Two experimental setups are then presented, which give the conditions under which the proposed framework is evaluated and what is expected to be observed. Finally, another experimental setup for testing the active learning functioning is presented.

5.1 Dataset

The dataset used in this work is the Pléiades one[1]. It consists of satellite views of airports from different locations throughout the world. There are two different classes of objects in the images: airplanes and helicopters. However, not all instances within a class are necessarily identical, as there exist distinct models (subclasses). Therefore, within the same class, objects can vary in dimension, color, and shape.

Pléiades Data

Specifically, the dataset comprises of 90 synthetic images and 48 real images. The real images are satellite views of airports, on which real objects are present, while the synthetic images are real satellite views of airports on which instances of each class, helicopters and planes, have been added digitally, in addition to the few real ones already present.

[1]The Pléiades images were downloaded from the Pléiades 4 Belgium platform, through an agreement with BELSPO (Belgian Science Policy Office). The images were labeled by the Institut Scientifique de Service Public (ISSeP).

Synthetic objects are added to the scene, taking into account real images metadata such as the position of the sun, to generate shadows and reflections. The instances are added by modifying orientation, location, exposure, etc., offering to the model more object diversity. So, using these synthetic images acts like data augmentation. It allows to have at disposal more samples to train, validate, and test the model, which is rather interesting, as images of airports with lots of airplanes and helicopters are rare.

The Pléiades images are very large, generally composed of several million pixels and their dimensions (height and width) are not constant. They vary greatly within the dataset, the smallest images being around 40 million pixels, while the largest exceed 170 million pixels.

However, the spatial resolution does not change between image acquisitions from different airports. The spatial resolution corresponds to the level of detail or granularity at which satellite sensors can distinguish objects on the Earth's surface, and it is measured in meters per pixel. For example, a spatial resolution of 1 meter per pixel means that one pixel within an image corresponds to a square ground surface of 1 meter by 1 meter in size. In this way, high-resolution enables the capture of greater details of objects on the surface. For Pléiades images, the spatial resolution of panchromatic sensors is about 0.5 meters per pixel. Naturally, the 3D airplane and helicopter models added to synthetic images are adapted to match the spatial resolution of the images.

JPEG 2000 Compression

In order to reduce the bandwidth consumed during transmission, only a part of the codestream is retained and transmitted. This as a negative impact on the spatial resolution of the images.

The decrease in spatial resolution is due to the downsampling operation applied during the DWT decomposition, as shown in Figure 2.3.

Normally, during reconstruction, an upsampling operation is performed to restore original resolution, but when lower resolution levels are considered, this operation is not realized. Thus, spatial resolution varies from one resolution level to another.

The spatial resolution ratios regarding the original resolution are given by:

- Resolution 5 corresponds to a ratio of 1:1.
- Resolution 4 corresponds to a ratio of 2:1.

- Resolution 3 corresponds to a ratio of 4:1.
- Resolution 2 corresponds to a ratio of 8:1.
- Resolution 1 corresponds to a ratio of 16:1.

Thus, an image with a dimension of 14863×11624 at resolution 5, is of size 929×727 at resolution 1, height and width are divided by a factor 16. The operation has the effect of reducing the image granularity, so objects lose details, and their characteristic features become more blurred.

It was decided to work with these resolutions, as going any lower compression levels would have made the network prediction almost impossible. Indeed, having higher ratios can cause instances to disappear. As the number of pixels representing an object becomes too small. An example is shown in Figure 5.1. It can be noticed that airplanes and helicopters disappear from the image as they are represented by so few pixels.

(a) Tile in resolution level 5. (b) Tile in resolution level 1 (not at scale).

Figure 5.1: Comparison between a cell in its original resolution and highly compressed (16:1).

Pre-Processing

For simulation, each image is sliced into patches because training and predicting on the whole image requires many computing resources. It has been chosen to cut

the images into tiles of size 1024×1024 at their original resolution.

But in order to keep a consistent division, the sizes of the tiles change with the considered resolution level. As it has been explained, lower resolution levels are not composed of the same number of pixels, this number is divided by a factor 2 at each lower resolution level. Thus, a tile in resolution 4 has a size of 512×512, and each time with smaller sizes until resolution 1, where tiles are of size 64×64. Thereby, tiles cover the same land surface area, and objects found at one resolution remain on the same tile, whatever the resolution level. This facilitates the work of the *Tile Indexer*. When the fine-tuned model makes an unreliable prediction at a given resolution about a certain tile, it is selected by the *Tile Indexer*. The block just has to store the identification number of the tile selected since it remains the same at the highest resolution level.

In addition, as the dimensions of the images in the dataset vary, it has been assumed that the test images are made up of 80 tiles. It allows to fairly compare the images, which was necessary to calculate the average performance during experiments. Therefore test images have fictitious size of 10240×8192.

Train/Val/Test Split

The whole dataset is splitted into two smaller sets, the training set and the test set. This model validation procedure allows to observe how the model performs on unseen data. It has been chosen that the test set is exclusively composed of real images, in order to observe the performance on real instances simulating the acquisition by an UAV. It is important to make sure that no images are included in both train set and test set, as it would induce a bias in performance, which would no longer simulate the results obtained on new data.

Typically, the training set is further splitted into the training set and the validation set. The validation set is used during the training phase to assess the performance of the model and adapt model parameters. Almost all real images are in the test set, the few that remain are mixed with the synthetic images and then randomly distributed between the train and validation sets.

Generally, the split ratios are about 70/10/20 for the train, validation, and test sets, respectively. For the Pléiades dataset, it has been chosen to work with ratios of 80/5/15, which corresponds to the ratios used with the COCO dataset to pre-train the weights of YOLO models, finally giving 108/7/21 images in the train, validation, and test sets, respectively. This gives approximately 25000 airplane instances and almost 10000 helicopters for training.

5.2 Evaluation Metrics

This section aims at presenting the evaluation metrics used to assess the performance of the model.

5.2.1 Intersection Over Union

Intersection over Union (IoU), also called Jaccard index, is a popular evaluation metric used for object detection tasks. It measures the similarities between the predicted bounding box and a ground truth. The prediction is obtained at the output of the neural network at inference time while, the ground truth bounding box is known and it results from the manual annotation of the image by human experts.

Figure 5.2: Example of 2D predicted and ground truth bounding boxes.

Intersection over Union is computed using the intersection area I and the union area U between the prediction and the ground truth. So, it is given by

$$IoU = \frac{|I|}{|U|} = \frac{|A \cap B|}{|A \cup B|}, \tag{5.1}$$

where A is the predicted box and B is the ground truth.

When the bounding boxes have no overlap, meaning they do not share any area, the intersection is zero and thus, the IoU is 0 as well. Conversely, the more the boxes overlap, the closer the IoU is to 1. Figure 5.3 shows three examples of Intersection over Union ratio.

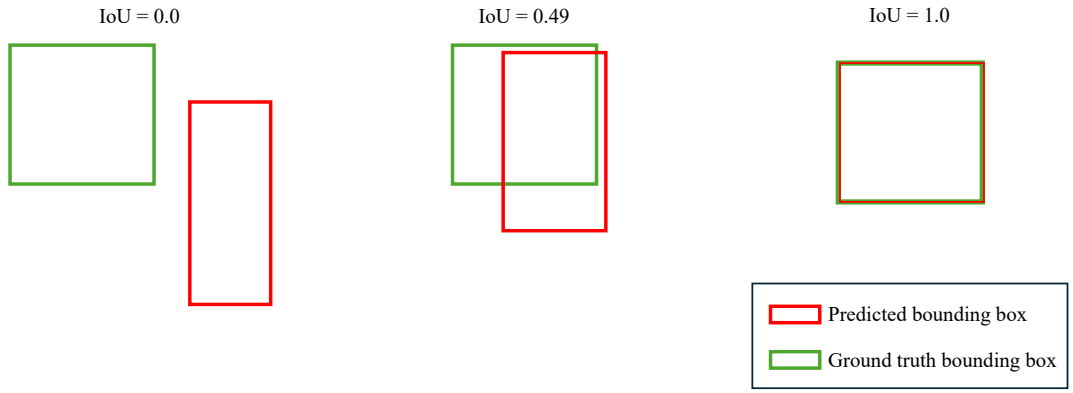

Figure 5.3: Intersection over Union cases.

At the output of the YOLOv8 model, a prediction gives the x and y-coordinates of the center, the height and the width of the bounding box. The same information is available for the ground truth, allowing to compute the intersection and the union areas between them.

But a problem occurs when testing a tile or an entire image because bounding boxes from the output of the neural network and the ground truth boxes do not present direct correspondences. In other words, there is no straight way of identifying which prediction belongs to which ground truth within a tile during the evaluation phase.

To overcome this problem, a matrix between the labels and the predictions is computed, expressing the values of the IoU between each label and all predicted boxes. Then, predictions are assigned to the ground truth that has the highest IoU. A predicted bounding box that does not overlap any ground truths stays unassigned and is automatically identified as false. Only one prediction can be assigned to each human annotated box, the one with the most overlap area.

5.2.2 True Positive, False Positive, And False Negative

First of all, in order to evaluate the system's performance, it is important to be able to distinguish which predictions are correct. To do this, three quantities are used: the True Positive (TP) rate, False Positive (FP) rate, and False Negative (FN) rates.

The true positive rate expresses the number of instances correctly predicted. A prediction is defined as correct when it satisfies two conditions. First, the Intersection over Union between a prediction and its assigned ground truth has to be greater than a chosen IoU threshold fixed at $IoU_{threshold} = 0.5$ to prevent missing TP at low-resolution, objects being represented by only few pixels. Then, the predicted class associated to the bounding box has to correspond to the ground truth label.

Afterwards, two types of errors are considered. Firstly, the neural network may miss objects while not predicting the actual position of an object. This number of missed instances corresponds to the false negative rate. It is computed by looking at the number of ground truths not assigned to a prediction or those assigned but missclassified. Secondly, the false positive rate expresses the number of wrong predictions. It is computed by counting the number of predictions that were made but where no objects were present.

In the context of search and rescue missions, the latter error is less punitive. It is better to overestimate the number of people to be saved than to underestimate and miss persons. For that reason, the model's confidence score threshold and the IoU threshold, two hyper-parameters of YOLOv8, are set at a lower level. The confidence score threshold expresses the limit above which a model prediction is considered. Setting it low allows to increase the number of false positives. The IoU threshold ensures that not too much predicted boxes overlap. Again, when this parameter is set low, more predictions are made in a limited area. As a result, the deep learning model's performance is not as good at first. But this is not a problem, since the predictions are then corrected by the human expert.

5.2.3 Metrics

To evaluate the hybrid annotation frameworks' performance, precision and recall scores are considered as well as the combination of both, the F-score. Generally, these two metrics are not used independently since, it exists a relationship between precision and recall.

Precision

The precision expresses the number of relevant instances among all retrieved predictions and is defined as

$$precision = \frac{TP}{TP + FP}. \tag{5.2}$$

Recall

On the other hand, the recall show the fraction of relevant predictions that were retrieved and is defined as

$$recall = \frac{TP}{TP + FN}. \tag{5.3}$$

These two metrics are related. For example, if the object detector parameters are set such that the model tends to detect more objects than there really are, the recall will increase and the precision will reduce. Conversely, the model could only predict something when it is really sure of itself. This functioning improves the precision but decreases the recall. In other words, having a greater recall increases the chances of not missing instances but surely increases the number of wrong predictions while, a greater precision decreases the chances of making wrong predictions but decreases the number of retrieve objects.

F_1-Score

A good way to synthesize both metrics is the use of the F_1-score which is the harmonic mean of the precision and recall. Having a prefect precision and recall gives the highest value possible, 1.0 while the score is equal to 0.0 when either the precision or the recall are zero. The metric is given by

$$F_1 = \frac{2precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN}. \tag{5.4}$$

Response Time

In order to compare both frameworks, the response time is also considered. It is defined as the time at which the decision is taken. It corresponds to the time interval between the acquisition of the image and the end of the system, when the model's predictions and the expert's corrections are done. The response time t_{RS} is defined as

$$t_{RS} = t_{TR} + t_{HUM}, \tag{5.5}$$

where t_{TR} is the transmission time and t_{HUM} is the human annotator time.

The transmission time corresponds to the time needed for the data to be transmitted through the channel which differs depending on the framework considered. Indeed, the data exchanged are not identical.

The transmission time for the baseline framework is defined as

$$t_{TR} = \frac{size(\bigcup_m T_m^{HR}.J2K)}{data\_rate}, \tag{5.6}$$

where the numerator corresponds to the size of all tiles in high-resolution and $data\_rate$ is the channel data rate available.

The proposed framework transmission time is defined as

$$t_{TR} = \frac{size(\bigcup_m T_m^{LR}.J2K) + size(\bigcup_n T_n^{HR}.J2K)}{data\_rate}, \tag{5.7}$$

where the numerator corresponds to the sum of size of all tiles at low-resolution and the size of transmitted tiles at high-resolution, and $data\_rate$ is the channel data rate available. The numerator is composed of two terms because the channel is used to transmit the entire image at low-resolution and then, the selected high-resolution tiles. As a reminder, the low-resolution level LR is adapted based on the $data\_rate$ and the transmission time limit $t_{TRlimit}$.

It should be noted that, in both frameworks, any metadata transmitted through the channel is neglected. Indeed, it is considered that the size of metadata files is small regarding to an image size and therefore has a negligible impact on the total data volume. Indeed, it can be considered that the metadata file transmitted are text files, and they are typically of size of few bytes. While, on average, a compressed tile of dimension 64×64 (i.e. in resolution 1) has a size of 7600 bytes. Table 5.1 shows the size of compressed tiles on average at each resolution level. It is therefore clear that a few bytes are negligible compared to the size of a compressed tile.

Finally, the human annotator time t_{HUM} corresponds to the time needed for experts to annotate the tiles of uncertainty, that means the tiles selected by the *Tile Indexer*. This time is defined as

$$t_{HUM} = \mu_{HUM} \times |\mathscr{N}|, \tag{5.8}$$

where μ_{HUM} is the time required to annotate a single tile and $|\mathscr{N}|$ is the number of tiles to annotate.

	Resolution 1	Resolution 2	Resolution 3	Resolution 4	Resolution 5
Tile dimension [in pixels]	64 x 64	128 x 128	256 x 256	512 x 512	1024 x 1024
Tile size on average [in kilobytes]	7.58	29.32	110.66	411.34	1440.34

Table 5.1: JPEG2000 compressed tile size on average at each resolution level measured on all tiles from the dataset.

Comparison Metrics

In short, the two metrics used to compare the baseline with the proposed approach are the F_1-score and response time t_{RS}. To highlight clearly the difference, the response time ratio is used and defined as

$$t_{RS\_ratio} = \frac{t_{RS}(Baseline)}{t_{RS}(Proposed)}, \tag{5.9}$$

together with the F_1 difference defined as

$$F_1\_diff = F_1(t_{RS}(Baseline)) - F_1(t_{RS}(Proposed)). \tag{5.10}$$

5.3 Experimental Setups

In order to assess the performance of the proposed hybrid annotation framework, multiple scenarios are considered. It has been chosen to adapt two parameters to constraint the system, the data rate $data\_rate$ and the transmission time limit $t_{TRlimit}$.

Experimental Setup 1: Basic Approach

First, there are three transmission time limits fixed at 3 minutes, 10 minutes, and 30 minutes. These limits represent three levels of emergency. Then, there are three values for the data rate that are considered: 22, 88, and 176 kbps. The minimum-to-median range available for the Iridium Certus satellite network[2] is used to choose these rates. This network corresponds to the standard for low-bandwidth UAV communications.

Higher rates were not considered since it would be equivalent to the baseline. By constraining the data rate to these values, the goal is to emulate an imperfect network, which is often the case in search and rescue missions, where infrastructures are generally damaged or where the network coverage is not sufficient. But

[2]https://www.iridium.com/iridiumcertus/

it is assumed that the transmission between the UAV and the ground station is perfect, meaning that the transmitted data is not affected by losses or other kind of errors. So, the transmitted codestream arrives to the ground station as it was sent.

It is considered that the time needed for annotator experts to check a tile is $\mu_{HUM} = 1$ minute/tile. It is also assumed that the experts perfectly annotate tiles. To simulate the human annotation, predictions made by the fine-tuned model are replaced by ground truths. In real use cases, the time needed to annotate a tile may vary and the human may occasionally do errors, impacting the performance improvements of the overall system.

In addition, the human experts are only available 10 minutes for annotating tiles, then they can only correct a maximum of 10 tiles when the channel allows transmission of a large number of tiles in high-resolution.

Table 5.2 gives the results obtained by the *Budget Calculator* block depending the scenarios studied and considering the average number of tiles of 80 and the tile sizes from Table 5.1.

Data rate	22 kbps			88 kbps			176 kbps		
Transmission time	3 min	10 min	30 min	3 min	10 min	30 min	3 min	10 min	30 min
Bandwidth available [in kB]	495.0	1,650.0	4,950.0	1,980.0	6,600.0	19,800.0	3,960.0	13,200.0	39,600.0
Low-resolution	-1	1	2	1	2	3	2	3	4
Budget annotator	N/A	0	1	0	2	7	1	3	4

Table 5.2: Results obtained with the Budget Calculator block for the 9 scenarios considered for images composed of 80 tiles.

In this case, the budget allocated to image correction, to send tiles to the annotators, is often very low, or even non-existent. As a result, predictions are mostly based on the output of the fine-tuned object detection model, taking almost no advantage of the hybrid approach, and human experts predictions.

Experimental Setup 2: Human Correction Oriented Approach

The objective of this second experiment is to observe the performance of the proposed framework when human annotators are more requested. Indeed, it has been observed that the remaining budget for sending high-resolution tiles is very often limited. In some cases, it is even zero.

This is why, the *Budget Calculator* block is slightly modified to favor human annotation. Now, it intentionally reduces the resolution LR by 1 if the annotator budget is not higher than what they offer. Obviously, unless it is already at minimum resolution level.

After all, it may be preferable to have slightly poorer results with the neural network model at initial prediction, and then have more budget available for human correction.

For the other constraints, the same transmission time limits and data rates are considered. In addition, identical human annotator constraints are used, meaning that $\mu_{HUM} = 1$ minute/tile and they can review a maximum of 10 tiles. By doing so, Table 5.3 is obtained.

Data rate	22 kbps			88 kbps			176 kbps		
Transmission time	3 min	10 min	30 min	3 min	10 min	30 min	3 min	10 min	30 min
Bandwidth available [in kB]	495.0	1,650.0	4,950.0	1,980.0	6,600.0	19,800.0	3,960.0	13,200.0	39,600.0
Low-resolution	-1	1	**1**	1	**1**	**2**	**1**	**2**	**3**
Budget annotator	N/A	0	**3**	0	**4**	**12**	**2**	**7**	**21**

Table 5.3: Results obtained with the Budget Calculator block for the 9 scenarios considered reducing the low-resolution by 1 when the annotator budget is not fully used.

Experimental Setup 3: Active Learning Framework

The last experiment aims at observing the performance evolution of the proposed framework in an active learning context. So, the primary objective is to study the improvement of the DL model during training iterations.

In this case, transmission time limits and data rates are not really used. In this third experiment, the detection is performed at resolution 1, corresponding to a very low-bandwidth scenario. It has been chosen to work at resolution 1 to observe the evolution of the system behavior in the most limited scenario.

Then, to simulate the AL loop, synthetic and real images are gathered in a single dataset, and constitute the unlabeled pool of data. At every AL iteration, 10 images are randomly selected from the pool, considered as acquired by the UAV. These still images are annotated by the DL model. For each of the 10 automatically annotated images, the 5 least confident tiles predicted are corrected by the annotators at

resolution 5.

In practice, repeating model training for every new image annotated is not feasible. For that reason, it has been decided to retrain the model on a bunch of new samples, every 10 images annotated, and thus every 50 tiles annotated.

In addition, the number of training epochs is fixed to 15. Again, in practice, the model has to adapt quickly, so the number of epochs has to be small to avoid long training phase.

5.4 Chapter Conclusion

In this chapter, two comparison metrics were developed: the F_1_diff and the t_{RS_ratio}. They are used to highlight the performance difference and the time gain between the proposed hybrid annotation framework and the baseline one.

Three ways of testing the proposed framework were explained.

- The basic approach highlights normal system functioning.
- It was noted that the annotation budget was not fully exploited. Therefore, the second approach tests the system by giving more tiles for human annotation. To do this, it reduces LR by 1, giving more budget for sending high-resolution tiles.
- The active learning experience studies the evolution of the proposed framework at the beginning of a mission when labeled samples were not available to prepare the model.

Chapter 6

Results And Discussion

In this chapter, the results obtained by emulating the emergency scenarios are presented and then discussed. The Pléiades dataset has been used to assess the proposed hybrid annotation framework and its active learning alternative.

6.1 Experience 1: Basic Approach

Firstly, the proposed framework is tested according to the constraint values from Table 5.2, obtained by the *Budget Calculator* for the nine scenarios.

By using these results, Figure 6.1 is obtained, plotting the F_1-score of the proposed framework and the baseline respect to the time.

Data Rate Of 176 Kbps

The results for the least constrained scenario, with a data rate of 176 kbps, can be observed in Figure 6.1a. Table 6.1 gives the numerical values for the two testing metrics.

$t_{TRlimit}$	3 min	10 min	30 min
F_1_diff	0.245	0.052	0.027
t_{RS_ratio}	24.250	7.462	2.853

Table 6.1: Comparison metrics obtained for the basic approach with a data rate of 176 kbps.

A first observation that can be made is that it exists a trade-off between the achievable performance and the response time when comparing the baseline with

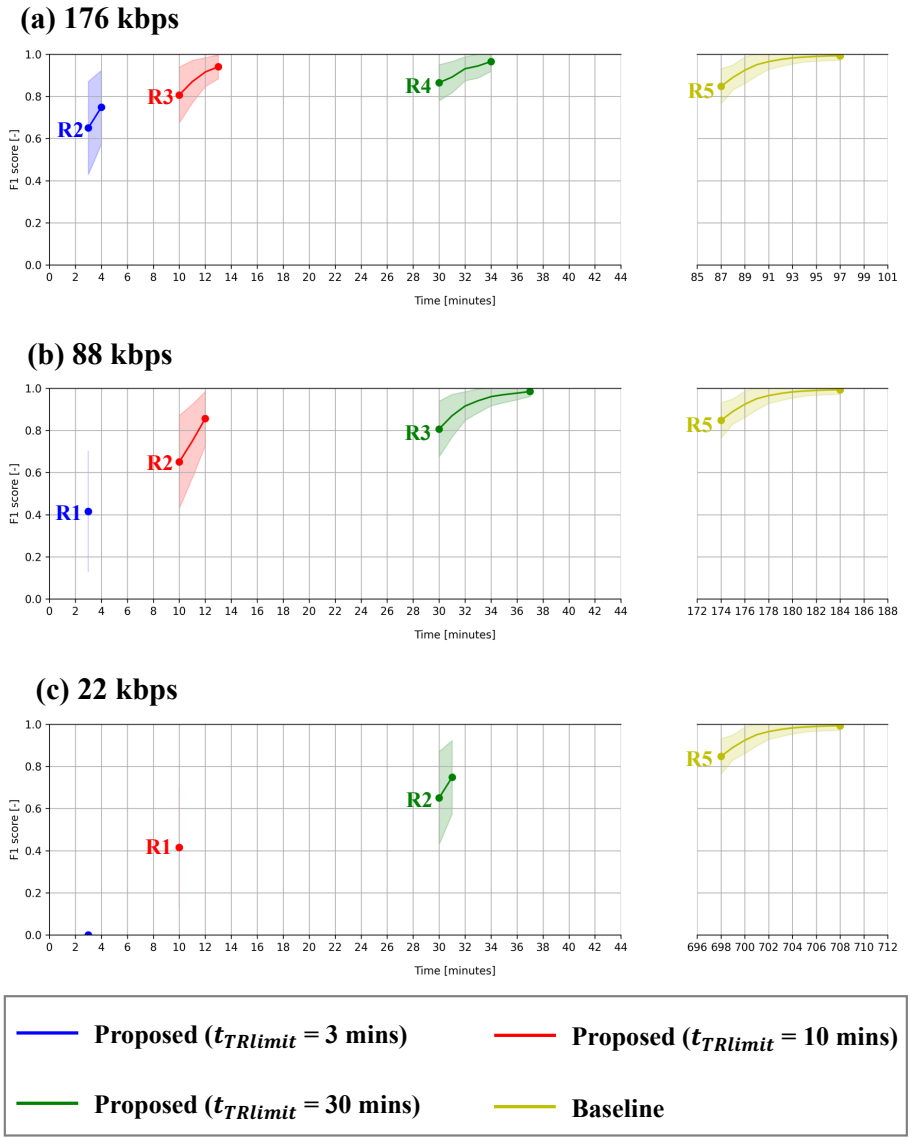

Figure 6.1: F_1-score versus time plot for the baseline and the proposed framework for high-resolution images from the test set under a data rate limit of (a) 176 kbps, (b) 88 kbps, and (c) 22 kbps, with a human annotator time of 10 minutes and considering scenarios with a transmission time limit of 3, 10, and 30 minutes.

the proposed approach.

Indeed, when the response time t_{RS} is shorter, the observed F_1-score is lower. Conversely, having more time available leaves room to the proposed framework for

improvements, resulting in higher performance.

Favor one or the other depends on the application, the right balance between speeding up the response time and maintaining adequate detection capability has to be chosen by users. In emergency situations, the response time is extremely important. But preserving high performance to ensure not missing objects of interest is just as essential.

One can observe that, with a transmission time limit of 30 minutes, the proposed framework divides the response time by 3 while achieving almost identical performance. Indeed, the performance is only reduced by 0.027.

For the other transmission time limits, the response time is greatly reduced while maintaining an acceptable F_1-score. In the most urgent case, the obtained F_1-score is about 0.749 on average even though the constraints allow to correct only 1 tile. The human annotation results in a 9% performance gain.

The baseline gives an almost perfect F_1-score of 0.992 on average, but takes 87 minutes to transmit the 80 tiles forming the image at the highest resolution $HR = 5$. Clearly, the baseline is not suited for emergency situations.

In this case, it is better to use the proposed framework instead of the baseline since almost the same performance is achieved in 30 minutes. Experts are less requested and the system enables faster decision-making.

However, it should be noticed that it is the most favorable scenario, so it will not be always possible to send the whole image at such a high-resolution and with so many tiles to the annotator for correction.

Data Rate Of 88 Kbps

The results for the intermediate scenario, with a data rate of 88 kbps, are plotted in Figure 6.1b. The values of the evaluation metrics are given in Table 6.2.

$t_{TRlimit}$	3 min	10 min	30 min
F_1_diff	0.578	0.137	0.008
t_{RS_ratio}	61.333	15.333	4.973

Table 6.2: Results obtained for the basic approach with a data rate of 88 kbps.

First of all, it can be noted that a data rate of 88 kbps is more restrictive for the system. For each transmission time limit $t_{TRlimit}$, the highest resolution for full transmission LR is reduced by 1 level.

For the case with the transmission time limit restricted to 30 minutes, one can observe that the resolution for the model is reduced to $LR = 3$ but the budget available for experts is larger, allowing up to 7 tiles for annotators. As a result, more corrections are made and the final performance is better, gaining from 0.965 to 0.985.

This result is encouraging for the next experimental setup in which the system voluntarily reduces the resolution to be able to correct more tiles.

When $t_{TRlimit} = 10$ minutes, the proposed framework allows to reduce the response time by a factor 15 while losing approximately 10% performance.

Finally, when the emergency situation allows 3 minutes of transmission time limit, the image can only be sent at the lowest resolution and without being able to use the annotator, giving a lower result. The obtained F_1-score is 0.415 on average with a standard deviation of 0.287.

Without taking into account this last case, the proposed framework enables to greatly reduce response time compared to the baseline, without compromising performance.

Data Rate Of 22 Kbps

The most restrictive scenario, with a data rate of 22 kbps, is shown in Figure 6.1c. Table 6.3 summarizes the performance obtained. This scenario allows to test the limit of the proposed approach.

$t_{TRlimit}$	3 min	10 min	30 min
F_1_diff	\	0.578	0.245
t_{RS_ratio}	\	70.800	22.839

Table 6.3: Results obtained for the basic approach with a data rate of 22 kbps.

A first observation is that with only 22 kbps, the available bandwidth is very low. For the less urgent case, the initial model annotation is performed on $LR = 2$ images with only 1 tile to be annotated. The decrease in performance is greater than for the two previous cases, with a F_1_diff decrease of 0.245 compared to

decreases of less than 0.050 before.

Then, for the 10 minutes case, only the image at $LR = 1$ can be sent for model annotation, giving low performance. No tile can be human corrected.

Finally, one can observe that with the transmission time limit of 3 minutes, the budget available does not even allow transmitting the whole image at low resolution, the constraints are too restrictive. Indeed, with a $data\_rate = 22$ kbps and a $t_{TRlimit} = 3$ minutes, the available bandwidth budget is 495 kB. But for a compressed image composed of 80 tiles at resolution $LR = 1$, the size is approximately 606 kB, which is far superior to what is allowed by the channel.

Inconsistent Predictions

By looking on Figure 6.1, one can observe that lower is the resolution considered, greater is the standard deviation.

The high standard deviation reflects a wide variation in performance between test samples. This is due to the fact that some images or tiles are very poorly predicted, while for others the prediction is relatively good. Figure 6.2 shows four predictions obtained on two tiles of the test set, for the airports of Severomorsk (a,b) and Lattaquie (c,d), at resolution levels 5 (a,c) and 1 (b,d).

Firstly, on Figure 6.2a, the model correctly predicts all the present entities in the image. The predicted bounding boxes are perfectly aligned with ground truth boxes while, in Figure 6.2c, not every airplanes within the tile are detected, certainly due to their size.

Despite the fact that both tiles are in resolution 5, the sizes of the airplanes do not remain the same, there are different aircraft models. Fortunately, thanks to the high-resolution of the image, the fine-tuned model predictions are satisfactory, so that performance is not significantly compromised.

But the observation is not so encouraging at the lowest resolution $LR = 1$. Indeed, when looking at Figure 6.2d, one can see that only one prediction has been made, and happily it is correct.

While in Figure 6.2b, despite the reduction in resolution, predictions remain good, the model predicts a false positive and misses a part of an aircraft.

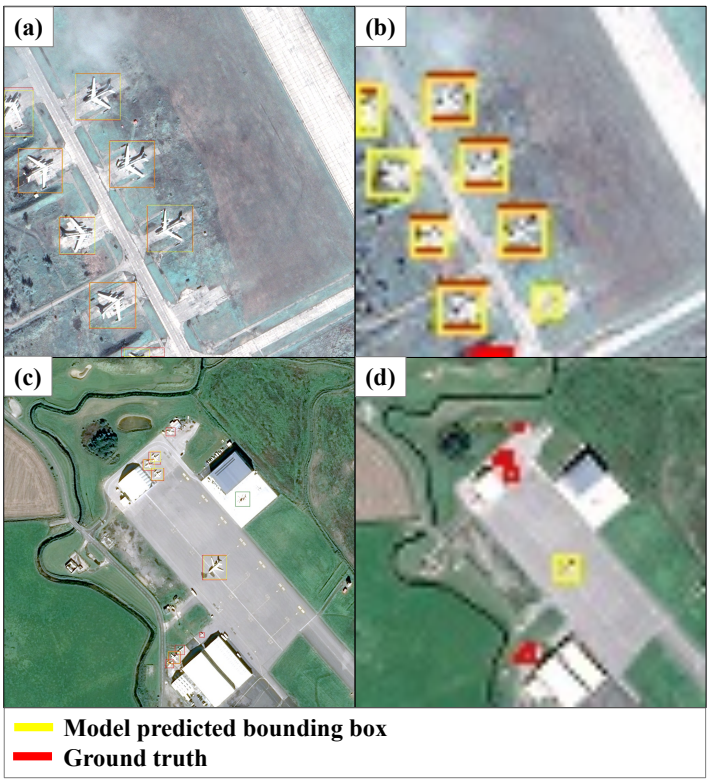

Figure 6.2: Fine-tuned DL model predictions for two different locations and resolutions.

Within the dataset, there are different models of airplanes and helicopters with different dimensions, some of which are significantly smaller. The problem is that, at low-resolution, these tend to lose all their distinctive features, becoming undetectable to the model, resulting in poor performance.

Tile Indexer Strategies

In Subsection 4.2.4, it has been suggested to use different *Tile Indexer* approaches that select additional tiles either randomly or based on a majority voting approach, but this did not yield conclusive results.

Indeed, generally, when an image is divided into tiles, the number of tiles with objects does not exceed 10, corresponding to the number of tiles to be annotated. Object appearances are usually concentrated on a few tiles. As a result, whatever the approach considered, the extra tiles selected contain no instances and make no difference to the overall performance.

This result highlights the fact that it is not worth using the proposed extensions for the Pléiades dataset. At least, for datasets with objects concentrated mainly in one area, the proposed framework is efficient. The system allows to restrict the human user's field of view to areas containing objects.

If instead of using 1024×1024 tiles, smaller ones were considered, the approaches could have a greater impact on framework performance.

6.2 Experience 2: Human Correction Oriented Approach

The proposed framework seems to be useful even in conditions where the entire image is sent at very low-resolution. However, it can be seen that if no corrections are made, the general performance is impacted.

This is why the resolution LR of the whole image sent for initial annotation by the fine-tuned model has been voluntary reduced, when the annotator was not fully used. Based on the values from Table 5.3, Figure 6.3 is obtained.

Data Rate Of 176 Kbps

Firstly, for a data rate of 176 kbps, one can observe that all annotator budgets have changed. The new metric values are given in Table 6.4.

$t_{TRlimit}$	3 min	10 min	30 min
F_1_diff	0.307	0.019	0.001
t_{RS_ratio}	19.400	5.706	2.425

Table 6.4: Modified results considering the human correction oriented approach with a data rate of 176 kbps.

There is an improvement in the F_1-score compared to the basic approach, for the transmission time limits of 10 and 30 minutes. But this comes at the expense of a slight increase in response time. Indeed, as more budget is available to annotators, they take longer to complete their task, reducing the speed of the whole system.

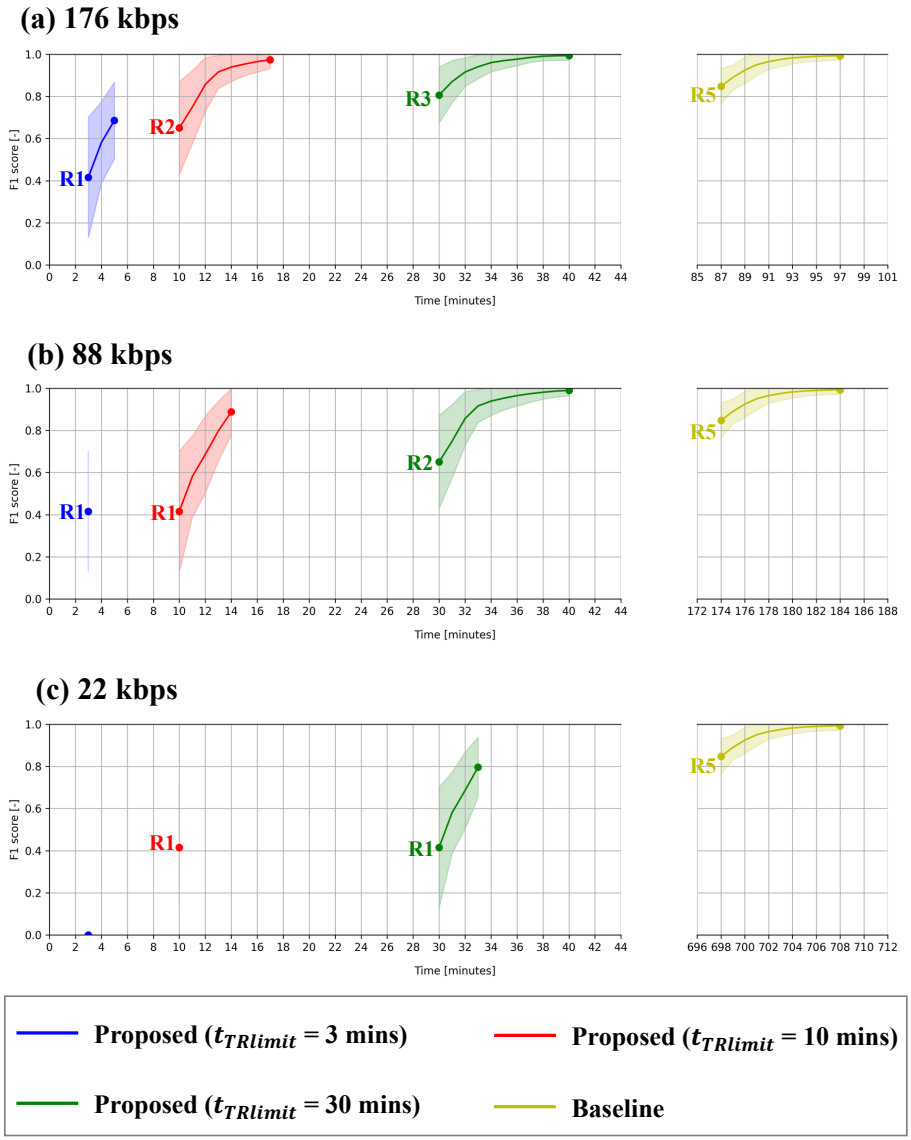

Figure 6.3: F_1-score versus time plot by intentionally reducing the resolution level by 1, allowing more budget for annotator experts.

But for the $t_{TRlimit} = 3$ minutes, the F_1-score is decreased and the response time is larger. The gain in annotator budget does not compensate for the loss in prediction capability on very low-resolution image.

Data Rate Of 88 Kbps

For the 88 kbps case, the modifications are given in Table 6.5. The observations that can be made are identical to those for the 176 kbps case. There is an improvement in the F_1-score but with an increase in response time.

$t_{TRlimit}$	10 min	30 min
F_1_diff	0.105	0.003
t_{RS_ratio}	13.143	4.600

Table 6.5: Modified results considering the human correction oriented approach with a data rate of 88 kbps.

Data Rate Of 22 Kbps

For the case with a data rate of 22 kbps, one can see that only one budget has changed, Table 6.6 highlights this modification.

Once again, reducing the resolution LR has a positive impact on final performance. The decision is taken later, as the annotator experts have to correct 3 tiles instead of 1, but it gives a slightly better F_1-score.

$t_{TRlimit}$	30 min
F_1_diff	0.197
t_{RS_ratio}	21.454

Table 6.6: Modified results considering the human correction oriented approach with a data rate of 22 kbps.

During the second experiment, the resolution available for the fine-tuned model was reduced by 1 to favor the annotator budget. In most cases, this has improved performance. However, response time is longer, as the annotator has more work to do.

6.3 Experience 3: Active Learning Framework

The objective of the third experiment is to assess the performance evolution of the proposed framework at the beginning of a mission.

Figure 6.4 shows the results obtained for the active learning framework. Each loop iteration corresponds to a re-training of the model on the previously annotated tiles

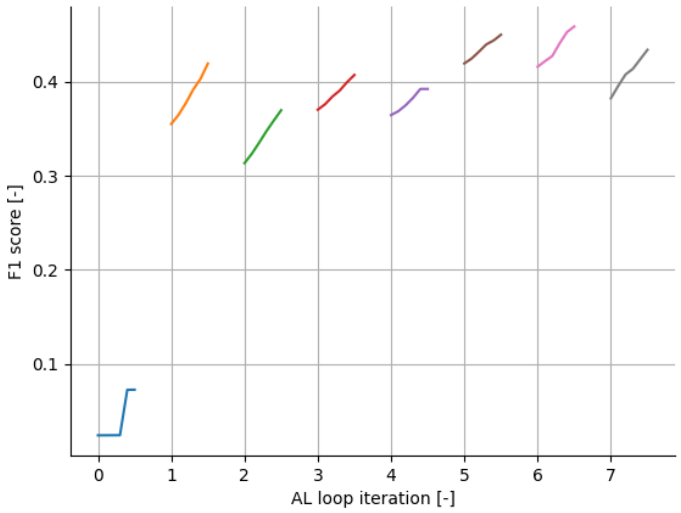

Figure 6.4: F_1-score vs AL loop iteration plot. Fine-tuning of DL-based model for resolution level 1 images in a context of AL.

Firstly, when the model did not have the opportunity to train on resolution 1 images, it has not learned low-resolution features that are different from higher-resolution images. The iteration 0 shows that the model is thus incapable of correctly predicting. The weights used were pre-trained on the COCO dataset, so they were already capable of detecting planes and helicopters, but still not able to detect the instances at resolution 1.

Fortunately, thanks to expert corrections, an F_1-score of 0.072 was achieved. A few predictions were made on the tiles, allowing them to be selected. But clearly, the model cannot stay untrained because performance is really too low.

Once the first training session has taken place, with the first 50 tiles corrected, one can already see that the model adapting to compression artifacts. The model only needed around 1000 airplane instances and 200 helicopter instances to reach its near-top performance for resolution 1.

Thereafter, a slight improvement over the iterations has been observed, to finally

reach the best performance after annotator correction at iteration 6, with an F_1-score of 0.459.

However, it can observed that at some iterations, the model's initial performance is not as good. This is probably due to the problem highlighted in Inconsistent Predictions, where some images are made up of smaller class instances, making low resolution prediction difficult.

In conclusion, from this application with very few examples, chosen on the basis of model uncertainties, it is possible for the model to adapt quickly to compression artifacts.

6.4 Chapter Conclusion

In this chapter, the results for the three experiences were presented. These have shown that it is possible to reduce response time with the proposed framework compared to the baseline but it exists a trade-off between response time and performance. So reducing time too much has a significant impact on performance.

It has been also shown that the human annotation approach provides better results but takes more response time. So annotators are more disturbed, wasting more of their valuable time.

Finally, it has been observed that the system does not require a large amount of data to already provide satisfactory performance. It adapts rapidly to compression artifacts.

Chapter 7

Conclusion

In this thesis, the objective was to improve the way high-resolution images are transmitted from UAVs to ground stations in emergency situations with limited bandwidth and transmission time.

Therefore, a hybrid annotation framework designed for bandwidth-restricted UAV object detection has been presented. The framework takes advantage of the JPEG 2000 compression algorithm to reduce the bandwidth-consumption. It relies on a fine-tuned deep learning model to predict initial bounding boxes on low-resolution images. Then, hybrid annotation is used to refine the initial predictions and improve overall performance.

More precisely, the transmission scheme was improved by integrating four blocks: a *Budget Calculator* that allows to take into account scenario constraints, a *JPEG 2000 Extractor* which extracts parts of interest within the JPEG 2000 codestream, a *Fine-Tuned Deep Learning Annotator* that maintains satisfactory detection capability at low-resolution, and the *Tile Indexer* that selects areas of images requiring human annotation based on model uncertainty.

The proposed hybrid annotation framework has shown faster response times while maintaining high F_1-scores. When the response time is divided by a factor of 15, there is only a performance loss of 10%. It has been also demonstrated that the framework does not require a large set of samples to adapt to the compression artifacts of very low-resolution images.

Further Research Directions

The study also highlighted areas for further explorations.

- During framework testing, only one dataset was considered. Thus, in order

to attest its robustness, it would be interesting to consider other datasets, potentially with more classes.

- It was observed that the different *Tile Indexer* approaches did not bring any major performance improvements given the distribution of instances across the images. But for other datasets, studying other approaches would make it possible to improve performance.
- In this work, only five DWT decomposition levels have been considered, it might be worthwhile to explore compression with larger ratios.
- The advantage of the proposed framework is that it can be adapted to different applications. In the future, imposing other constraints on the system is a path that needs exploring (constraining computational resources, for example). Or even, other deep learning models could also be incorporated, depending on the computer vision task.

Publication

A paper has been written as part of this master thesis [25]. Its content is approximately the same as that of the master's thesis, but in a synthesized form and without addressing the active learning part.

Bibliography

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE.

[2] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine learning from theory to algorithms: An overview. 1142:012012.

[3] Nassim Ammour, Haikel Alhichri, Yakoub Bazi, Bilel Benjdira, Naif Alajlan, and Mansour Zuair. Deep learning approach for car detection in UAV imagery. 9(4):312.

[4] W. Budiharto, A. A. S. Gunawan, J. S. Suroso, A. Chowanda, A. Patrik, and G. Utama. Fast object detection for quadcopter drone using deep learning. In *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, pages 192–195, 2018.

[5] Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. 13(1):89.

[6] Lahiru D. Chamain and Zhi Ding. Improving deep learning classification of jpeg2000 images over bandlimited networks. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4062–4066, 2020.

[7] Rene Y Choi, Aaron S Coyner, Jayashree Kalpathy-Cramer, Michael F Chiang, and J Peter Campbell. Introduction to machine learning, neural networks, and deep learning.

[8] Demetris Demetriou, Pavlos Mavromatidis, Ponsian M. Robert, Harris Papadopoulos, Michael F. Petrou, and Demetris Nicolaides. Real-time construction demolition waste detection using state-of-the-art deep learning methods; single–stage vs two-stage detectors. *Waste Management*, 167:194–203, 2023.

[9] .Usha Ruby Dr.A. Binary cross entropy with deep learning technique for image classification. 9(4):5393–5397.

[10] Lixuan Du, Rongyu Zhang, and Xiaotian Wang. Overview of two-stage object detection algorithms. In *Journal of Physics: Conference Series*, volume 1544, page 012033. IOP Publishing, 2020.

[11] Xuedan Du, Yinghao Cai, Shuo Wang, and Leijie Zhang. Overview of deep learning. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 159–164. IEEE.

[12] Max Ehrlich, Larry Davis, Ser-Nam Lim, and Abhinav Shrivastava. Analyzing and mitigating JPEG compression defects in deep learning. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2357–2367. IEEE.

[13] Karim El Khoury, Martin Fockedey, Eliott Brion, and Benoît Macq. Improved 3d u-net robustness against JPEG 2000 compression for male pelvic organ segmentation in radiotherapy. 8(4).

[14] Karim El Khoury, Jonathan Samelson, and Benoît Macq. Deep learning-based object tracking via compressed domain residual frames. 1:765006.

[15] Karim El Khoury, Maxence Wynen, Pietro Maggi, and Benoît Macq. Improving 3d lesion segmentation robustness against image compression in multiple sclerosis.

[16] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang. Tood: Task-aligned one-stage object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3490–3499, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[17] Wendong Gai, Yakun Liu, Jing Zhang, and Gang Jing. An improved tiny YOLOv3 for real-time object detection. 9(1):314–321.

[18] Farhad Ghazvinian Zanjani, Svitlana Zinger, Bastian Piepers, Saeed Mahmoudpour, and Peter Schelkens. Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images. 6(2):1.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[20] Tianxu He, Shukui Zhang, Jie Xin, Pengpeng Zhao, Jian Wu, Xuefeng Xian, Chunhua Li, Zhiming Cui, et al. An active learning approach with uncertainty, representativeness, and diversity. *The Scientific World Journal*, 2014, 2014.

[21] O Hmidani and EM Ismaili Alaoui. A comprehensive survey of the r-cnn family for object detection. In *2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–6. IEEE, 2022.

[22] A Indradjad, A S Nasution, H Gunawan, and A Widipaminto. A comparison of satellite image compression methods in the wavelet domain. 280(1):012031.

[23] W. Jin, J. Yang, Y. Fang, and W. Feng. Research on application and deployment of uav in emergency response. In *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 277–280, 2020.

[24] Benjamin Kellenberger, Diego Marcos, Sylvain Lobry, and Devis Tuia. Half a percent of labels is enough: Efficient animal detection in UAV imagery using deep CNNs and active learning. 57(12):9524–9533.

[25] Karim El Khoury, Tiffanie Godelaine, Simon Delvaux, Sebastien Lugan, and Benoit Macq. Streamlined hybrid annotation framework using scalable codestream for bandwidth-restricted uav object detection. *arXiv preprint arXiv:2402.04673*, 2024.

[26] Jeong-ah Kim, Ju-Yeong Sung, and Se-ho Park. Comparison of faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, pages 1–4. IEEE.

[27] A.S. Lewis and G. Knowles. Image compression using the 2-d wavelet transform. 1(2):244–250.

[28] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in Neural Information Processing Systems*, 33:21002–21012, 2020.

[29] N. Lin, Y. Liu, L. Zhao, D. O. Wu, and Y. Wang. An adaptive uav deployment scheme for emergency networking. *IEEE Transactions on Wireless Communications*, 21(4):2383–2398, 2022.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context.

[31] Mingjie Liu, Xianhao Wang, Anjian Zhou, Xiuyuan Fu, Yiwei Ma, and Changhao Piao. UAV-YOLO: Small object detection on unmanned aerial vehicle perspective. 20(8):2238.

[32] Vladimir Nasteski. An overview of the supervised machine learning methods. 4:51–62.

[33] Misgana Negassi, Rodrigo Suarez-Ibarrola, Simon Hein, Arkadiusz Miernik, and Alexander Reiterer. Application of artificial neural networks for automated analysis of cystoscopic images: a review of the current status and future prospects. 38(10):2349–2358.

[34] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks.

[35] Kirtan Gopal Panda, Shrayan Das, Debarati Sen, and Wasim Arif. Design and deployment of UAV-aided post-disaster emergency network. 7:102985–102999.

[36] Vladan Papić, Petar Šolić, Ante Milan, Sven Gotovac, and Miljenko Polić. High-resolution image transmission from UAV to ground station for search and rescue missions planning. 11(5):2105.

[37] Akshara Preethy Byju, Gencer Sumbul, Begum Demir, and Lorenzo Bruzzone. Remote-sensing image scene classification with deep neural networks in JPEG 2000 compressed domain. 59(4):3458–3472.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE.

[39] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[40] Raul Rojas and Raúl Rojas. The backpropagation algorithm. *Neural networks: a systematic introduction*, pages 149–182, 1996.

[41] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[42] Nida Shahid, Tim Rappon, and Whitney Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. 14(2):e0212356.

[43] Amit K. Shukla, Manvendra Janmaijaya, Ajith Abraham, and Pranab K. Muhuri. Engineering applications of artificial intelligence: A bibliometric analysis of 30 years (1988–2018). 85:517–532.

[44] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. 18(5):36–58.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[46] Norbert Tuśnio and Wojciech Wróblewski. The efficiency of drones usage for safety and rescue operations in an open area: A case from poland. 14(1):327.

[47] Gaoang Wang, Jenq-Neng Hwang, Craig Rose, and Farron Wallace. Uncertainty-based active learning via sparse modeling for image classification. *IEEE Transactions on Image Processing*, 28(1):316–329, 2018.

[48] Xiaoliang Wang, Peng Cheng, Xinchuan Liu, and Benedict Uzochukwu. Fast and accurate, convolutional neural network based approach for object detection from UAV. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3171–3175. IEEE.

[49] Xingxing Xie, Gong Cheng, Jiabao Wang, Xiwen Yao, and Junwei Han. Oriented r-cnn for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3520–3529, 2021.

[50] Zhihua Xu, Lixin Wu, and Zhenxin Zhang. Use of active learning for earthquake damage mapping from uav photogrammetric point clouds. *International journal of remote sensing*, 39(15-16):5568–5595, 2018.

[51] Asma Yamani, Albandari Alyami, Hamzah Luqman, Bernard Ghanem, and Silvio Giancola. Active learning for single-stage object detection in UAV images. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1849–1858. IEEE.

[52] Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao, and Furao Shen. Image data augmentation for deep learning: A survey. *arXiv preprint arXiv:2204.08610*, 2022.

[53] Jiajie Zhang, Jian Weng, Weiqi Luo, Jia-Nan Liu, Anjia Yang, Jiancheng Lin, Zhijun Zhang, and Hailiang Li. REMT: A real-time end-to-end media data transmission mechanism in UAV-aided networks. 32(5):118–123.

[54] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU loss: Faster and better learning for bounding box regression. 34(7):12993–13000.

[55] S. Zhou, C. Deng, B. Zhao, Y. Xia, Q. Li, and Z. Chen. Remote sensing image compression: A review. In *2015 IEEE International Conference on Multimedia Big Data*, pages 406–410, 2015.

[56] Jingbo Zhu, Huizhen Wang, Benjamin K Tsou, and Matthew Ma. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on audio, speech, and language processing*, 18(6):1323–1331, 2009.

Appendix

A.1 Introduction to Machine Learning and Computer vision

Over the past few years, Artificial Intelligence (AI) has become more and more present in various domain and many articles about the subject are published everyday [7, 43]. However, it remains a misunderstanding about the similarities and differences that exist between AI, Machine Learning (ML) [2], and Deep Learning (DL) [11], three inextricably linked but tightly related concepts.

Nowadays, AI-based technologies are becoming increasingly widespread since they are applied in various sectors such as robotic, education, finance, medication, and many more. But concisely, the aim of AI is to automate tasks ordinarily performed by human (e.g. writing, creating art, translating, managing data, ...) using a programmable machine. Whereas, ML and DL are two sub-fields of AI, where tasks are automated but involving the concept of learning. Indeed, an AI machine not necessarily learns how to complete an intellectual task, the machine operation can be exactly defined by the designer, describing to the machine how to interpret its environment and what to do about it. For example, AI system might be used to filter incoming emails; analyzing the context, and interpreting the content, to decide whether it is reliable or spam. In ML and DL, the approach would involve a model learning to differentiate between the two, which can be done by using spam and non-spam labeled emails as examples. The model then learns patterns, finding the characteristics that differentiate legitimate emails from spam.

One of the most commonly used machine learning algorithm to carry out these intellectual tasks is the Artificial Neural Network (ANN) [33, 42] which is inspired by the neural architecture of the brain. Such models are composed of nodes (inspired by neurons) communicating through links (representing the synapses). Figure 7.1 illustrates the analogy between the biological cell and its artificial correspondence. A node receives input signals on which it apply some non-linear function, called

activation function, to generate the output signal. Each link between two nodes is weighted in order to reinforce or attenuate the signal, these weights are determined and adjusted during the training phase.

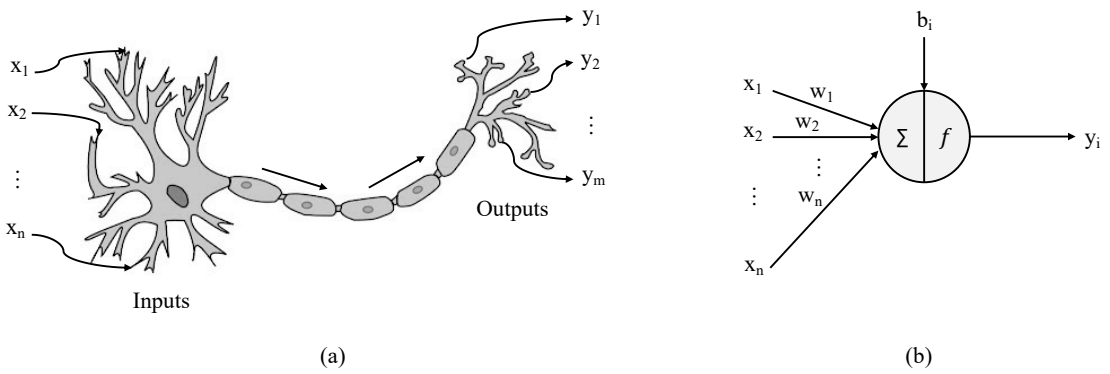

Figure 7.1: Illustration of neural networks' constitutive element. Analogy between (a) a biological neuron and (b) an artificial neuron.

In Figure 7.1b, an artificial neuron is represented. It takes as input the signals $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ weighted by $\mathbf{w} = \{w_1, w_2, ..., w_n\}$ plus a certain bias b_i. The output signal y_i results from the application of a non-linear activation function f on the sum of inputs. Mathematically, the output of the i-th neuron is given by:

$$y_i = f(\mathbf{w}\mathbf{x} + b_i) = f(\sum_{j=1}^{n} w_j x_j + b_i). \tag{7.1}$$

To simplify the notation, the equation can be rewrite as,

$$y_i = f(\sum_{j=0}^{n} w_j x_j), \tag{7.2}$$

with $w_0 = b_i$ and $x_0 = 1$. Thanks to the activation function f, non-linear transformations are introduced within the network, enabling to model and approximate non-linear relationships in the data. There are a multitude of different functions, all with their advantages and disadvantages, but the most commonly used are Sigmoid, Rectified Linear Unit (ReLU), or even Softmax activation functions.

Generally, a node is not used alone, several nodes are aggregated to constitute a layer. Then, neural networks are built using a succession of layers; an input layer, multiple intermediate layers (called hidden layers), and an output layer. Figure 7.2 shows a generic representation of a shallow neural network taking two inputs

and outputting a single result. It is composed of three layers, each composed of artificial neurons. Every neuron from one layer are connected to every neuron in adjacent layers, it is said that the network is fully-connected. The size of these dense models is growing rapidly with the number of layers, since there are weighted connections between all neurons. Typically, when a network has more than two hidden layers, it is considered as a deep neural network.

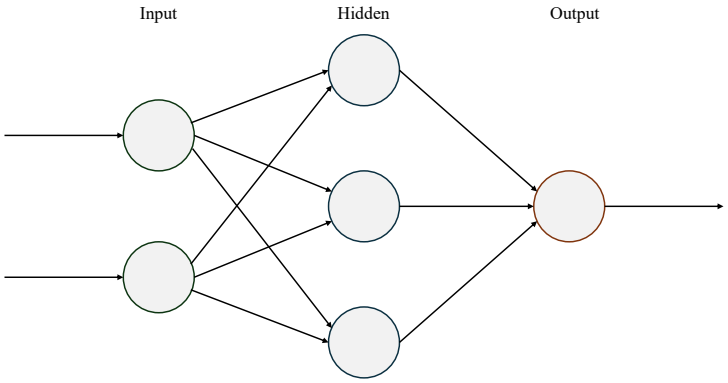

Figure 7.2: Fully-connected shallow neural network composed of three layers.

Afterwards, in order to have a machine learning algorithm able to accomplish a desired task (e.g. classification, segmentation, object detection, etc.), the network's parameters are optimized through a learning phase. In most cases, the learning is performed by minimizing a loss function that measures the error made between the model's predicted output and a known expected output, this approach is known as supervised learning [32]. If the predicted output postpones from what is expected, the network is adapted. The weights are iteratively updated given several training samples until the model reaches satisfactory performance. Learning the parameters in this way allows to then generalize to unseen data, by applying the operations learned to new data.

Supervised learning approach requires having at disposal a set of training samples $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where N is the number of samples available, x_i is a feature vector, and y_i is the associated label (the expected output). The set of labeled examples is provided by human annotators that take of their time to review each input (feature vector) and specify the result that the model must provide.

During training, the model has to learn a function $g : X \rightarrow Y$ that enables it to provide the correct output from an input. To do so, the model operations are

applied on a training samples to predict an output, $g(x_i) = \hat{y}_i$.

Then, a loss function is used to measure the dissimilarities between the predicted value and the expected label, so the loss of a given sample (x_i, y_i) is written as $L(\hat{y}_i, y_i)$. The gradient of the loss function with respect to each weight is calculated to be then back-propagated through the network, adjusting the weights, which minimizes the error [40].

Finally, for object detection tasks, the ANN inputs are the pixels constituting the image on which objects are to be detected. The problem is, with such models, spatial information is lost and ANNs struggle manipulating image data due to their computational complexity. Actually, when dealing with images, pixel locations constitute a meaningful resource. Pixels close to each other within an image are much more likely to be correlated than pixels on opposite sides. This issue is overcome using Convolutional Neural Network (CNN) [1, 34] which preserves the spatial relationship of pixels within the image.

Rather than treating each input independently, multiple nodes are grouped to form convolutional filters, which are able to extract features more suited for image-based tasks. For example, the specific features extracted can be horizontal or vertical lines, image color, corners, ... but also not human interpretable features. Figure 7.3 shows the convolution operation used in CNN. The convolutional filter applies a dot product between the kernel and the values comprise in a moving window that passes through the entire image, to generate the output, called a feature map.

The Figure 7.3 shows that the generated value is influenced by 9 values of pixels in the image, allowing spatial information to be taken into account. In this example, the kernel has an arbitrary size of 3×3, which can be changed depending the network application. Thus, increasing its size enables the model to capture more global context, but also increases the computational cost.

The deeper is the convolutional network, the more abstract the features extracted become, leaving a latent representation of the image. As for ANN, the parameters constituting the filters are learned during the training phase. Layers composed of convolutional filters are interesting because they are composed of fewer parameters than standard fully-connected layers, significantly reducing model complexity and thus facilitating training, and allowing to have a deeper architecture without increasing the number of parameters. Making this type of architecture one of the most widely used for computer vision tasks [48].

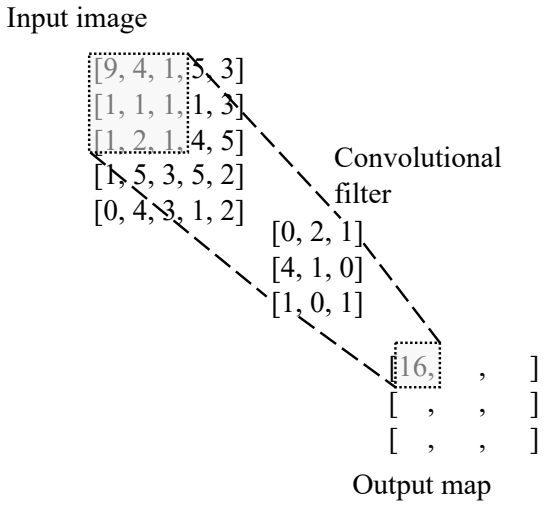

Figure 7.3: Convolution filter operation. Kernel of size 3×3.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**