Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Department of Electrial Engineering
Master in Electrical Engineering – Electronics and Telecommunications

PROCESSING OF MICROSCOPY IMAGES IN THE COMPRESSED DOMAIN

NICOLAS VASCONCELLOS

Leiria, March 2025

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Department of Electrial Engineering
Master in Electrical Engineering – Electronics and Telecommunications

PROCESSING OF MICROSCOPY IMAGES IN THE COMPRESSED DOMAIN

NICOLAS VASCONCELLOS
Number: 2232720

Master's Dissertation supervised by Prof. Luís Távora, Prof. Carlos Grilo, Prof. Rolando Miragaia, and Prof. Lucas Thomaz.

Leiria, March 2025

ACKNOWLEDGMENTS

I would like to express my gratitude to my family, especially to my parents, Freddy Freire and Katherine Vasconcellos, for their unwavering support throughout my academic journey in Portugal. During the most challenging moments, they were the ones who encouraged me to keep going, and thanks to their support, I am here, about to finish my degree.

I want to thank my sister, Giulianna Vasconcellos, who has been a constant source of encouragement and strength since she arrived in Portugal. Her day-to-day support has helped me focus more on becoming better and achieving my goals.

I am grateful to my friends, Carlos Almeida and Daniel Nicolau, for always being there with me, overcoming difficulties together, and motivating each other to improve and "break our limits".

A special thanks goes to Professor Luís Távora for introducing me to the field of Artificial Intelligence and for pushing me to explore increasingly challenging areas of study. I am thankful for the faith the professor had in me, inviting me to work on his projects and recognizing my potential to grow and achieve great things.

I also want to express my appreciation to Professor Lucas Thomaz, who has been by my side every day, guiding my work. His advice and observations have always aimed to bring out the best in me and push me to grow both professionally and personally. At the end of this journey, I feel like I not only gained a professor but also a friend whom I can rely on to learn and grow.

I would like to thank Professor Carlos Grilo, who has always been there to support my crazy ideas and foster my interest in the field long before the beginning of this project.

Additionally, I extend my gratitude to Professor Rolando Miragaia, whose advice provided a much more practical view of applying the algorithms and methods studied, leading to an improvement in the quality of this work.

It has been a pleasure to be guided by all these professors. Each one has shown me what it truly means to be a professional and a teacher who cares about the growth of their students. It has been an honour to work with all of you, and I hope to continue learning from you and making new advancements in this fascinating field of Artificial Intelligence and all its applications.

I want to thank all the personnel at the “Multimedia Signal Processing - Lr” laboratory at IT, including José Filipe, Rui Lourenço, João Parracho, Edgar Paulo, Ruben Susano, and the interns, Rúben Francisco and Nuno Fernandes. Their support throughout the development of this project has ranged from helping me learn to use the lab computers to resolving the most disastrous errors. Their patience and willingness to help have shown me a great sense of camaraderie, and I look forward to continuing to work with them for a long time. By the end of this journey, we have almost become like a family, and I am delighted to have met such wonderful people.

To all who have contributed to my growth and success, thank you for being a part of my journey. Your support means the world to me.

This work was supported by the Instituto de Telecomunicações and the Fundação para a Ciência e a Tecnologia (FCT), Portugal under projects CoMBINNe 2022.09914 .PTDC (DOI:10.54499/2022.09914.PTDC), Programa Operacional Regional do Centro, and by FCT/MCTES through national funds and when applicable co-funded by EU funds under the project UIDB/EEA/50008/2020 (DOI: 10.54499/UIDB/50008/2020) and LA/P/0109/2020 (DOI:10.54499/LA/P/0109/2020).

RESUMO

Este trabalho apresenta uma abordagem inovadora para a deteção de objetos no domínio comprimido, eliminando a necessidade de descompressão completa da imagem enquanto mantém alta precisão na deteção. O sistema proposto, Processamento de Imagens de Microscopia no Domínio Comprimido (ProMIC), integra uma arquitetura de Tradutor de Domínio, permitindo que redes de deteção de objetos processem diretamente representações no domínio comprimido. Esta abordagem reduz significativamente os custos computacionais enquanto preserva informações visuais essenciais para tarefas de deteção. O estudo também incluiu uma análise preliminar da eficiência de compressão do JPEG-AI em comparação com codecs clássicos (HEVC/H.265 e VVC/H.266) em vários conjuntos de dados.

O método proposto baseia-se num modelo fundamental inspirado numa abordagem anterior de deteção de objetos no domínio comprimido. As principais inovações neste trabalho incluem: a utilização de um Bloco Residual Leve (LRB) que melhora a extração de características de representações do domínio comprimido, melhorando a robustez da deteção em diferentes taxas de compressão; um processo de Afinação a partir do Domínio de Pixels (PD-FT) que melhora a adaptação do modelo para processar representações no domínio comprimido; e uma estratégia de Treino Guiado do Tradutor de Domínio (GDTT) que refina o Tradutor de Domínio para extrair informações significativas diretamente das representações comprimidas, minimizando a diferença entre modelos de deteção de objetos nos domínios comprimido e de pixels.

Foram conduzidas várias experiências para comparar o desempenho tanto do modelo Base quanto da arquitetura proposta em relação a um modelo de referência de deteção de objetos que opera em imagens reconstruídas. Os resultados no conjunto de dados *Lucchi++* mostram que o ProMIC supera significativamente o modelo Base. Notavelmente, a arquitetura proposta excede o modelo de referência em precisão média média, alcançando uma melhoria de 2,31 pontos percentuais na maior taxa de bits. Isto destaca o potencial do processamento de imagens no domínio comprimido para capturar características significativas que podem ser perdidas em imagens totalmente reconstruídas. Além disso, a aplicação do método ProMIC reduziu a complexidade do sistema de deteção de objetos em 42,34% em comparação com o modelo de referência.

Estes resultados validam a viabilidade da deteção direta de objetos em representações comprimidas, oferecendo uma alternativa computacionalmente eficiente

às abordagens tradicionais no domínio de pixels. O sistema proposto tem aplicações promissoras em processamento em tempo real, sistemas embebidos e análise de imagens em grande escala baseada em IA, abrindo caminho para modelos de visão mais eficientes e escaláveis que operam diretamente no domínio comprimido.

ABSTRACT

This work presents a novel approach to the detection of mitochondria in microscopy images within the compressed domain, eliminating the need for full image decompression while maintaining high detection accuracy. The proposed architecture, Processing of Microscopy Images in the Compressed Domain (ProMIC), integrates a Domain Translator architecture, enabling object detection networks to operate directly on images' latent representations. This approach significantly reduces computational costs while preserving essential visual information for detection tasks. The study also included a preliminary analysis of JPEG-AI compression efficiency against classical codecs (HEVC/H.265 and VVC/H.266) across multiple datasets.

The proposed method builds upon a Base model inspired by a previous compressed-domain object detection approach. Key innovations in this work include: the use of a Lightweight Residual Block that enhances feature extraction from latent representations, improving detection robustness across different compression rates; a Pixel Domain Fine-Tuning process that enhances the Domain Adaptation of the model; and a Guided Domain Translator Training strategy that refines the Domain Translator to extract meaningful information directly from latent codes, minimising the gap between compressed and pixel-domain object detection models.

Extensive experiments were conducted to compare the performance of both the Base model and the proposed architecture against a reference object detection model that operates on reconstructed images. Results on the *Lucchi++* dataset show that ProMIC significantly outperforms the Base model. Notably, the proposed architecture exceeds the reference model in mean average precision, achieving a 2.31 percentage point improvement at the highest bitrate. This highlights the potential of compressed-domain image processing to capture meaningful features that may be lost in fully reconstructed images. Additionally, applying the ProMIC method reduced the complexity of the object detection system by 42.34% compared to the reference model.

These findings validate the feasibility of direct object detection in latent codes, offering a computationally efficient alternative to traditional pixel-domain approaches. The proposed framework has promising applications in real-time processing, embedded systems, and large-scale AI-driven image analysis, paving the way for more efficient and scalable vision models that operate directly in the compressed domain.

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI	Artificial Intelligence.
AP	Average Precision.
BD	Bjontegaard Delta.
bpp	Bits per Pixel.
CDE	Cluster Density Error.
CD-OD	Compressed Domain Object Detector.
CNN	Convolutional Neural Network.
CT	Computed Tomography.
DNN	Deep Neural Network.
GAN	Generative Adversarial Network.
GDTT	Guided Domain Translator Training.
IoU	Intersection Over Unit.
LIME	Local Interpretable Model-Agnostic Explanations.
LRB	Lightweight Residual Block.
ML	Machine Learning.
MTL	Multi-Task Learning.
PCA	Principal Component Analysis.
PD-FT	Pixel Domain Fine Tuning.
PD-L	Pixel Domain processing Layers.
PD-OD	Pixel Domain Object Detector.

PD-TL	Pixel Domain Transfer Learning.
ProMIC	Processing of Microscopy Images in the Compressed Domain.
PSNR	Peak Signal-to-Noise Ratio.
RD	Rate-Distortion.
TL	Transfer Learning.
VAEs	Variational Autoencoders.
XAI	Explainable Artificial Intelligence.
xDNN	Explainable Deep Neural Networks.

1

INTRODUCTION

Image compression is the process of reducing the size of an image file, preferably without significantly degrading its visual quality. The primary goal is to minimize the amount of data required to represent an image, making it easier to store, transmit, and process. Image coding is essential in various applications, including digital photography [1], video streaming [2], medical imaging [3], and remote sensing [4]. The growing demand for higher image resolution and quality has led to a corresponding increase in uncompressed image sizes, leading to the need for more efficient image coding solutions to facilitate storage and transmission [5].

Recently, learning-based image coding has emerged as a powerful alternative to traditional image compression techniques, leveraging deep learning to achieve superior efficiency and perceptual quality [6]. Classical image coding approaches, such as JPEG and HEVC-based methods, rely on hand-crafted designs and modular codec architectures. In contrast, modern learned compression models rely on autoencoders to create approximately invertible mappings between pixel data and a quantized latent representation. These models incorporate entropy coding, using a probabilistic prior on the latent representation to generate a compressed bitstream compatible with standard arithmetic coding algorithms [7].

Machine Learning (ML) algorithms have shown great effectiveness in performing tasks in which pattern recognition plays an important role [8]. That is the case in medicine, for example in areas such as hematology [9], [10], dermatology [11], [12], oncology [13]–[15], ophthalmology [16], radiography [17], [18], and neurology [19], [20]. Regarding image processing, Deep Neural Network (DNN), a widely used ML type of model, currently represent the state of the art in numerous computer vision tasks, including high-level image understanding, such as image classification [21], semantic segmentation [22], and face recognition [23], as well as low-level image processing tasks, such as image denoising [24], super-resolution [25], enhancement [26], and inpainting [27].

Given that compact representations are widely employed in state-of-the-art vision and processing tasks, learning-based image coding offers the potential to create, for the first time, an efficient compressed representation suitable for both human and machine visual consumption. This concept forms the basis of the JPEG-AI standard (ISO/IEC 6048-1, ITU-T T.840), the first international standard for image coding based on an end-to-end learning-based approach. By leveraging DNNs, JPEG-AI

achieves superior rate-distortion performance and enhanced perceptual visual quality. The primary motivation behind this new standard is to address the growing need for efficient image compression solutions, particularly in applications where images must be interpreted by both humans and machines [5].

Building upon the aforementioned motivation, this work proposes an object detection framework that operates directly on the compressed domain representation of images generated by a learning-based encoder, specifically JPEG-AI VM 6.1. Processing images in the compressed domain differs significantly from traditional methods, which usually require full decompression and pixel-domain analysis to extract content. By eliminating this step, the proposed method reduces resource consumption, including energy, memory, and computational time.

The remainder of this document is structured as follows: Chapter 2 provides an overview of the key concepts and technologies underlying the approach proposed in this work, including learning-based image coding, multi-task learning, and compressed domain image processing. Chapter 3 details the proposed framework for object detection using compressed domain representations and describes the design of the Domain Translator block, a critical component that enables the interpretation of compressed domain information by a modified object detector. Chapter 4 presents a comparative study of the compression efficiency of the new JPEG-AI standard versus commonly used classical codecs, such as HEVC and VVC, in three different microscopy image datasets. This chapter also includes an experimental evaluation of object detection performance using the proposed methods. Finally, Chapter 5 presents the conclusions of this work and suggests directions for future research in this domain.

2

BACKGROUND

This chapter provides a detailed overview of the foundational concepts underlying the approach proposed in this work. The content is organized into three main topics: (1) Learning-based image coding, (2) Multi-Task Learning, and (3) Compressed Domain Image Processing.

Section 2.1 presents an overview of significant developments in Learning-based Image Coding and introduces the new JPEG-AI standard, which plays a central role in this work. Section 2.2 examines Multi-Task Learning strategies in Deep Neural Networks, which are fundamental to both the object detection algorithms and the proposed pipeline for compressed domain-based object detection. Section 2.3 explores Compressed-Domain Image Processing, an emerging field in Deep Learning that enables direct image analysis on compressed representations without requiring full decompression.

Each section examines state-of-the-art methodologies, key innovations, and recent advancements in their respective fields. Section 2.5 presents the metrics used to evaluate both the compression efficiency and object detection performance of the proposed models. Finally, Section 2.6 describes the datasets used in the experimental work and details the data splits employed for training the ML models.

2.1 LEARNING-BASED IMAGE CODING

Traditional image compression techniques, such as JPEG and HEVC-based methods, rely on predefined transform coding approaches. In contrast, learning-based techniques jointly optimize the transformation and quantization processes in a unified framework, achieving superior rate-distortion efficiency. Recent breakthroughs in deep learning, particularly in variational autoencoders (VAEs), generative adversarial networks (GANs), and attention mechanisms, have accelerated advancements in this area [28], [29].

2.1.1 *End-to-End Optimization of Nonlinear Transform Codes*

Traditional image compression schemes rely on hand-crafted transforms, such as the discrete cosine transform (DCT) or the wavelet transform, followed by quantization and entropy coding. These methods, while effective, are limited in their ability to adapt to complex image structures. In contrast, end-to-end optimization frameworks employ deep neural networks to learn highly efficient nonlinear transformations that better capture image features.

Early works in this area explored autoencoder-based architectures, where an encoder network maps images to a compact latent space, followed by quantization and entropy modeling to ensure efficient coding. The decoder then reconstructs the image from the compressed representation, with the entire system trained to minimize a rate-distortion objective. One notable approach is nonlinear transform coding, introduced by Ballé et al., which employs differentiable transforms and learned entropy models to optimize compression performance in an end-to-end manner [30].

2.1.2 *Variational Image Compression*

A key innovation in learning-based image coding is variational image compression, which employs VAEs to model the probability distribution of the latent representation. Unlike traditional coding schemes, which assume independent distributions for transformed coefficients, VAEs leverage probabilistic models to learn complex dependencies in image data.

Variational models introduce a prior distribution on the latent space and use Bayesian inference to optimize compression efficiency. These models achieve state-of-the-art rate-distortion performance by effectively capturing interdependencies within the data. Recent works have enhanced variational compression frameworks with hyperpriors that model spatial dependencies, improving the accuracy of entropy estimation and reducing bits-per-pixel requirements while maintaining high visual quality. Hyperpriors function as an additional latent representation that refines the compression model, leading to more accurate probability estimation and improved coding performance [7].

2.1.3 *Attention-Based and Generative Models for Image Compression*

Attention mechanisms and generative models have been explored to further enhance learned image compression. Transformer-based architectures and self-attention mechanisms enable improved spatial adaptation, allowing networks to allocate bits more efficiently based on image content. These models analyse global dependencies within an image, rather than relying solely on local feature extraction, improving compression efficiency and reconstruction quality [31].

Moreover, generative adversarial networks (GANs) have been employed to refine the perceptual quality of reconstructed images by leveraging adversarial loss functions. GAN-based approaches enhance realism in reconstructed images by minimizing perceptual distortion, making them particularly valuable for low-bitrate applications. Studies have shown [32] that incorporating discretized Gaussian mixture likelihoods and attention layers in learned compression pipelines leads to superior performance in both objective and subjective evaluations. This integration allows compression models to better account for variations in texture, color, and structural details, ensuring more natural-looking reconstruction [32].

2.1.4 *JPEG-AI: First Learning-based Image Coding Standard*

The JPEG-AI standard represents a significant milestone in the evolution of image compression technologies. Developed by the Joint Photographic Experts Group (JPEG) committee, JPEG-AI, ISO/IEC 6048-1 (ITU-T T.840), is the first international standard for image coding based on an end-to-end learning-based approach. This innovative standard leverages DNNs to achieve superior rate-distortion performance offering enhanced perceptual visual quality. The primary motivation behind JPEG-AI is to address the growing demand for efficient image compression solutions, particularly in applications where images are consumed by both humans and machines [5].

2.1.4.1 *Key Features and Design Elements*

JPEG-AI introduces several features that make it different from conventional image coding standards. One of the most notable aspects of JPEG-AI is its ability to generate a single-stream, compact compressed domain representation that targets both human visualization and machine-driven tasks. This multi-task optimization is achieved through a learning-based image coding algorithm that transforms an image into a latent tensor, which is then compressed and transmitted. The decoder is intended to either reconstruct the image or perform other compressed domain

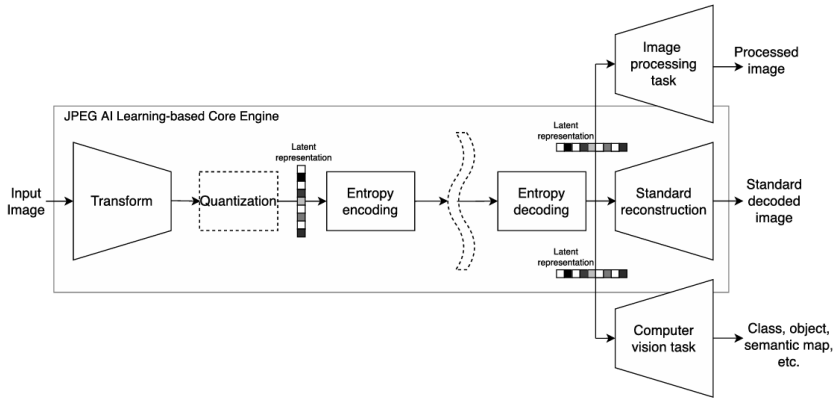


Figure 1: JPEG-AI Multi-Task Pipieline [5].

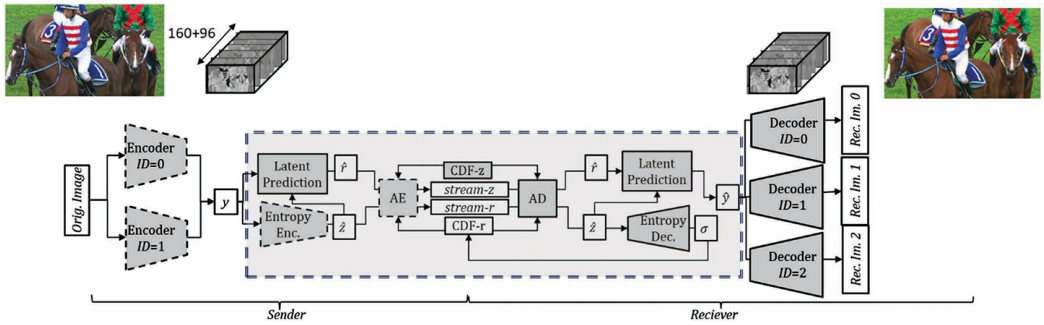


Figure 2: JPEG-AI image reconstruction architecture. The IDs in the encoders and decoders represent different complexity levels available, higher the ID value, the higher the complexity [33].

operations using the latent tensor, making JPEG-AI highly versatile. This multi-task pipeline is illustrated in Figure 1.

The standard supports different color formats, including YUV and RGB, and employs conditional color separation to reduce peak memory usage. This approach allows the primary luminance component to be coded independently with a more powerful neural network, while chrominance components leverage information from luminance as an auxiliary input [33].

JPEG-AI also features a multi-branch coder architecture, where neural-network-based analysis and synthesis transforms are used to compute latent information at the encoder and to reconstruct the image at the decoder. The standard specifies multiple decoders with varying levels of complexity, allowing devices to choose the decoder that best fits their computational capabilities. This flexibility ensures that JPEG-AI can be deployed across a wide range of devices, from low-end mobile phones to high-performance computing systems. In Figure 2, the JPEG-AI architecture for image reconstruction with encoding and decoding options with different complexity levels is presented.

2.1.4.2 *Progressive and Partial Decoding*

One of the key functionalities of JPEG-AI is its support for progressive decoding. The latent tensor channels are ordered such that the most important channels come first, allowing devices with low computational capabilities to decode only a subset of the channels for faster, lower-quality previews. This feature is particularly useful for applications where low-latency image transmission is required, such as in social media or live streaming [5].

Additionally, JPEG-AI supports partial picture decoding, where specific regions of an image can be decoded independently. This is achieved through the use of residual tiles, which correspond to rectangular portions of the image. By parsing only the relevant residual tiles, the decoder can reconstruct specific regions of interest without processing the entire image. This capability is especially beneficial for applications such as virtual reality, where only a portion of the image may be visible to the user at any given time.

2.1.4.3 *Performance and Future Developments*

JPEG-AI has demonstrated significant performance improvements over traditional codecs. In terms of compression efficiency, JPEG-AI achieves up to 27% bit savings compared to VVC/H.266, while offering much faster encoding and decoding times, when run on GPUs [33]. The standard's ability to maintain high perceptual quality at low bitrates makes it particularly suitable for applications such as social media, where images are often transmitted at low resolutions.

Looking ahead, the JPEG committee plans to extend the capabilities of JPEG-AI to support additional computer vision and image processing tasks. Future versions of the standard are expected to focus on enhancing the latent domain representation, enabling tasks such as image classification and super-resolution to be performed directly from the compressed domain. This will further reduce the computational load typically associated with full image decoding and low-level feature extraction.

2.2 MULTI-TASK LEARNING IN DEEP NEURAL NETWORKS

Much of the research in Artificial Intelligence (AI) has focused on developing models that solve different tasks independently, i.e. one network per task. However, much like the human brain, which is capable of performing multiple tasks, learn with minimal supervision, and generalise acquired skills, MTL aims to perform multiple related tasks together, so that the knowledge gained from one task can be used to

improve performance in other tasks, thereby increasing generalisation across all the tasks involved [34].

Inherent relationships and correlations between different tasks can be exploited to improve the performance of models when trained together [34]. When related tasks share complementary information, they can improve performance and act as regularisers for each other. MTL aims to address multiple related tasks by simultaneously optimising the loss functions of these tasks.

MTL has several advantages over single-task approaches. First, it avoids redundant learning of common features across tasks, significantly reducing overall resources consumption such as energy and memory. Second, MTL can learn more general features by averaging the inherent noisy patterns across tasks. Thirdly, it can prioritise critical features that may be difficult to distinguish within a single task framework. Finally, it introduces inductive biases that help mitigate the problem of over-fitting [35].

2.2.1 *Parameter Sharing Strategies*

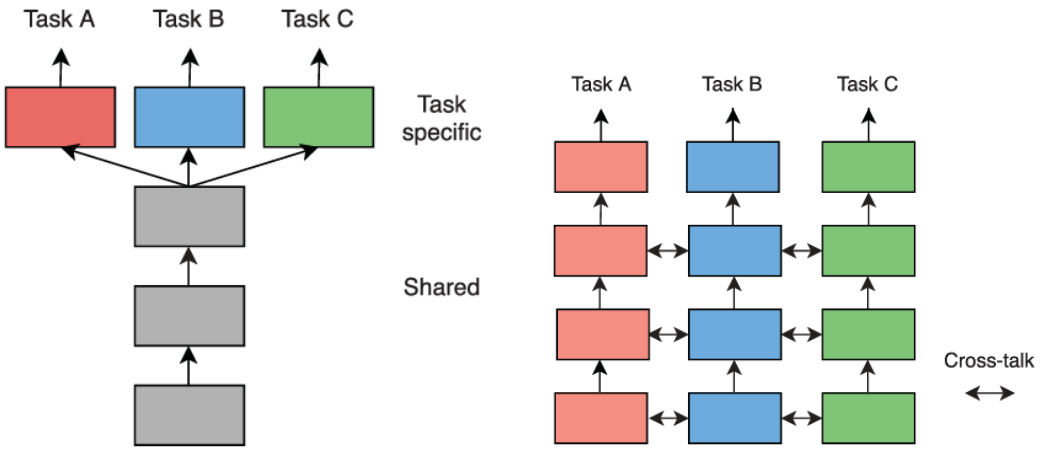
Various attempts have been carried by different authors to establish MTL taxonomies based on the different strategies to share resources when designing network architectures capable to solve multiple problems. Among the most common classifications are the parameter sharing techniques described by [36] and the encoder- and decoder-focused architecture classification used by [37]. This section details these two taxonomies, highlighting the relationships between them and presenting some recent works that fit on these categories.

2.2.1.1 *Parameter Sharing Techniques*

Historically, MTL approaches using DNNs were first classified according to the strategy used to share information within multitask challenges [38]. The methods were then divided into two categories: hard and soft parameter sharing techniques.

HARD PARAMETER SHARING This technique typically involves sharing some hidden layers between all tasks, with only a few output layers dedicated to each task. Network architectures designed using this technique typically consist of a shared encoder that branches into task-specific heads, as illustrated in Figure 3a.

Among the most notable examples of MTL models using the hard parameter sharing technique are UberNet and Multi-linear Relationship Networks. UberNet, a DNN proposed by [39], simultaneously handles low-, mid-, and high-level vision tasks



(a) Hard Parameter sharing. (b) Soft Parameter sharing.

Figure 3: Hard and Soft Parameter sharing techniques [37].

by introducing techniques that facilitate the training process while using different training sets and handling multiple tasks with a limited memory capacity. This model features a multi-head design across different network layers.

Multi-linear relationship networks, introduced by [40], extend the concept of a shared encoder with task-specific decoders by incorporating tensor priors that model task-specific parameters to learn multi-linear relationships between features, classes and tasks. This approach can mitigate the problems of negative and insufficient knowledge transfer between tasks. However, it has the limitation that it is only suitable for specific task sets due to its predefined architecture [41].

SOFT PARAMETER SHARING In this case, each task is assigned to its own network with a specific architecture and settings, much like in single-task learning. However, a feature sharing mechanism is used to manage cross-talk, where information is transferred via skip connections between the architectures for each task [42]. This technique is illustrated in Figure 3b. Some of the most notable MTL models that use the soft parameter sharing technique include cross-stitch networks [43], gated multi-task networks [44], and sluice networks [45].

Cross-stitch networks, proposed in [43], manage feature sharing by introducing a novel unit designated "cross-stitch unit". These units learn a linear combination of activation maps from multiple networks and propagate this combined information to subsequent layers. This approach allows the network to generalise across multiple tasks, mitigating overfitting and improving the performance of each task. However, this method has two main limitations. First, the performance of the model is significantly influenced by the relationship between the tasks chosen for joint learning. Secondly, the optimal placement of the sharing units within the network to maximise performance is not a simple issue. If the information required to solve

each task is not correlated, or if it is transferred in the wrong place, this can lead to under-sharing or even negative sharing, resulting in performance degradation for each task compared to a single-task approach.

To address the first limitation of cross-stitch networks, [44] proposed a gating mechanism that allows selective feature and evidence sharing across task-specific layers. This is achieved by adding a scalar weight between each pair of tasks. The method allows the selection of the most appropriate tasks for information sharing and regulates the influence of each task, thus preventing negative sharing and enhancing the contribution of beneficial tasks. Networks that implement this mechanism are called gated multi-task networks.

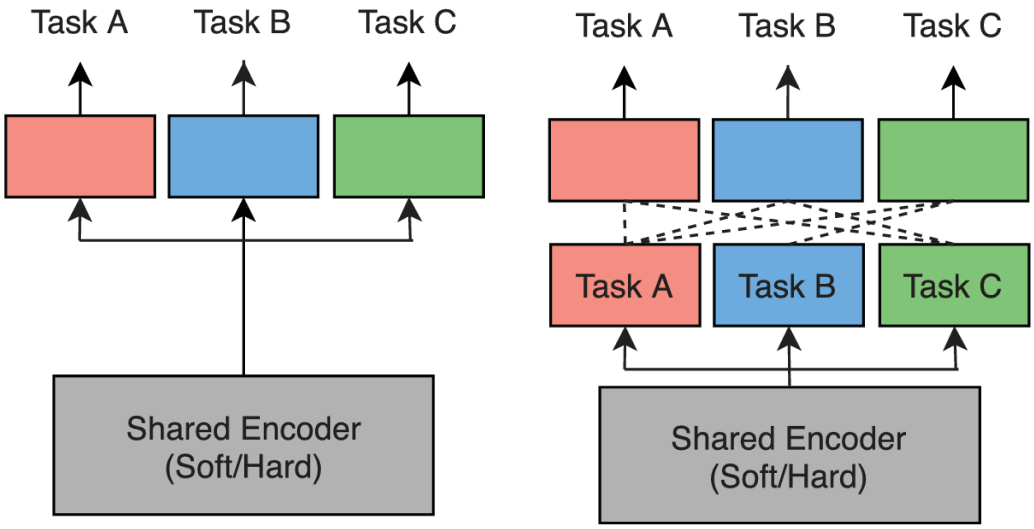
To overcome the second limitation of cross-stitch networks, [45] introduced a framework for learning multi-task architectures known as the "Sluice Network". This model facilitates the sharing of parameters, sub-spaces, and skip connections between multiple neural networks, allowing specific parts of layers to be used for different tasks. By learning to share only the most relevant components, the Sluice Network effectively addresses the challenge of determining the optimal positions for feature sharing within the network, and therefore improves performance across tasks.

2.2.1.2 *Parameter Sharing Focuses*

With the evolution of MTL models, it became debatable whether they can be classified solely based on the parameter-sharing technique adopted. In this context, [37] proposed a novel taxonomy for classifying MTL approaches according to the locus of task interactions. Based on this criterion, MTL models are categorised into either encoder or decoder-focused architectures. These two categories will be described upon in the next sections.

ENCODER-FOCUSED MODELS Similar to the underlying concepts of traditional hard parameter sharing models, encoder-focused architectures share information only within the encoder, leaving task-specific processing to independent decoders. Various works have adopted an ad hoc strategy by integrating an off-the-shelf backbone network with small task-specific heads. Information sharing within the encoder can be achieved by either hard or soft parameter sharing mechanisms. This sharing scheme is illustrated in Figure 4a.

A notable example of this type of architecture is the 2C-Net proposed by [46]. The 2C-Net is a collaborative compression and classification network based on a deep neural network (DNN)-based image compression model introduced by [7]. This model exploits the information present in the compressed image representation



(a) Encoder-focused. (b) Decoder-focused.

Figure 4: Encoder and Decoder-focused architectures [37].

generated by the encoder to perform classification tasks. To achieve this goal, a feature-analytic classifier is used to exploit the shared latent representations generated by the standard compression model.

The compression and classification models are trained using a three-stage strategy. First, the compression model is trained individually until it is close to convergence. Then, the parameters of the common encoder and compression decoder modules are fixed and the classification decoder is trained for the classification task. Finally, when the feature-analytic classifier is close to convergence, the entire 2C-Net architecture is jointly trained, with different weights assigned to each task to balance overall performance.

DECODER-FOCUSED MODELS The encoder-focused models described in the previous section have a significant limitation: they often fail to capture multi-modal information and task-specific differences that could be beneficial to one another. To address this problem, recent research has explored the use of multi-task networks to perform preliminary tasks and feed their results into a multi-modal network, which then combines them to improve performance on another task [37]. Network architectures that share inter-task information at the task-specific heads are referred to as decoder-focused models. The concept of this category is illustrated in Figure 4b.

A notable example of this type of architecture is the PAD-Net proposed by [47]. As shown in Figure 5, in the PAD-Net framework, general features are extracted by a backbone encoder and then passed through task-specific heads to generate initial task predictions. These predictions are then combined using a multi-modal distillation unit to produce the final predictions. It is worth noting that all the initial auxiliary tasks are closely related to the final desired tasks, ensuring that

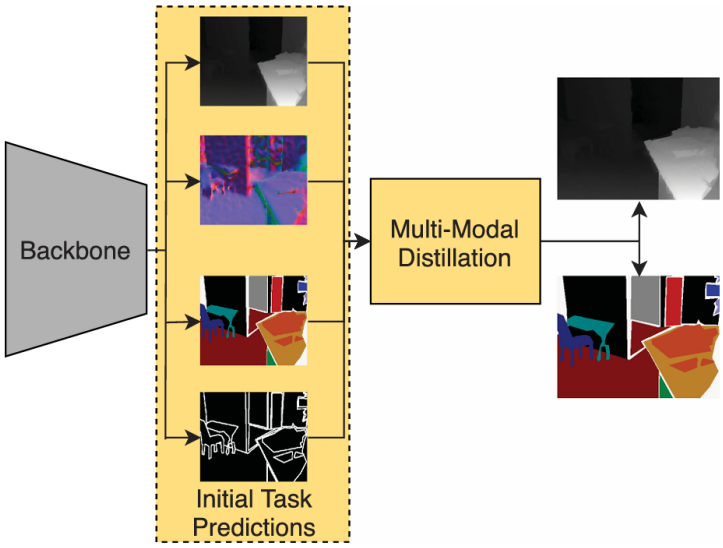


Figure 5: PAD-Net architecture [47].

the information from them is useful for the final predictions. However, this type of architecture only works well if the selected tasks are related, as discussed in Section 2.2.1.1, highlighting the importance of optimal task selection when designing multi-task systems.

2.2.2 *Task Relationship-Oriented Network Architectures*

The architecture design of MTL models is intricately linked with the relationships between the selected tasks. Consequently, [35] proposed a new taxonomy that focuses on four popular MTL embodiments and their relation to the task relationships. These architecture categories are cascaded, parallel, interacted, and hybrid. This section explains each one of these categories as well as how they can guide the design of MTL networks to efficiently address a specific set of tasks.

2.2.2.1 *Cascaded Networks*

In the cascaded architecture, all tasks are performed sequentially, with the output of a preceding task feeding into the sub-network of subsequent tasks. In these networks, there is no parameter sharing between the task-specific networks. These networks are closer to single-task approaches than other MTL models, with the key difference being that each subsequent task is strongly dependent on the results of the previous ones, as illustrated in Figure 6(a).

Cascaded architectures can be trained end-to-end to optimise all tasks simultaneously, or they can be trained in multiple stages to handle computational limitations such as limited memory. This concept was applied by [48] to address the biomarker

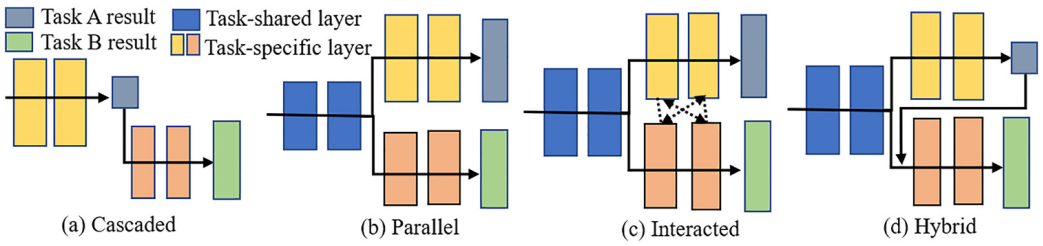


Figure 6: Tasks relationship-related Network architectures' categories for MTL [35].

localisation problem by combining a Convolutional Neural Network (CNN) classifier with a Generative Adversarial Network (GAN). In this case, the classifier and discriminator process images generated by the encoder-decoder module to effectively remove biomarkers from the input images.

2.2.2.2 *Parallel Networks*

In the parallel architecture, there are independent task-specific layers that learn particular features for each one of the different tasks in parallel. It has been observed that DNN tend to learn hierarchical image representations [49], starting with low-level features in the early layers and progressing to high-level task-specific information in the deeper layers. Following this trend, parallel architectures use a backbone network to learn hierarchical encoding structures that are common to the selected tasks, as illustrated in Figure 6(b).

It is important to note that in parallel architectures, parameter sharing occurs only within the backbone network. This type of network is particularly suitable for task sets that share essential knowledge relationships but differ in complexity, such as segmentation and classification tasks. An example of this type of architecture is the 2C-Net model described in Section 2.2.1.2.

2.2.2.3 *Interacted Networks*

As shown in Figure 6(c), in interacted network architectures, the task-specific subnetworks described in Section 2.2.2.2 have a parameter-sharing mechanism that allows information exchange between them. This type of architecture is particularly suitable for task combinations that are are closely related, so that deep features remain useful for improving the performance of other tasks. Typically, well-suited task combinations include those with similar levels of complexity.

An illustrative example of interacted architectures is the multi-task segmentation network for prostate Computed Tomography (CT) images presented by [50]. The backbone U-Net network delineates the bladder and rectum, which are more easily distinguishable structures in prostate CT images. At the same time, an attention

sub-network is constructed to hierarchically transfer the backbone features and adaptively learn discriminative representations for prostate bed segmentation, a more difficult structure to identify [35]. In this case, both tasks involve segmentation, illustrating the degree of task relatedness required to efficiently exploit the potential of this architecture.

2.2.2.4 *Hybrid Networks*

This category includes all MTL models that do not fit into the three categories described above. Hybrid architectures use at least two of the Cascaded, Parallel, or Interacting architectural design strategies, exploiting the strengths of each. This type of architecture may be suitable for more complex task combinations than the other three categories. Figure 6(d) shows an example of a hybrid network architecture using both parallel and cascaded schemes. In this example, both tasks share the feature extractor layers while maintaining independent task specific heads, similar to parallel networks. However, while task A depends only on the network input, task B also depends on the result of task A, similar to cascaded networks.

An example of this architecture is proposed by [51]. This work presents a joint deep learning model for 3D lesion segmentation and classification for interpretable COVID-19 diagnosis. The framework consists of three sub-networks: a cross-talk feature extractor, a 3D lesion segmentation head, and a disease diagnosis classification head. The interaction between these three modules corresponds to the three categories described in this section. The presence of a feature extractor and multiple heads exemplifies a parallel interaction between tasks. At the end of the tasks, there is a regulariser connection, which implies interactive behaviour. In addition, the classification task not only depends on the information provided by the feature extractor, but also requires the result of the lesion segmentation, demonstrating a cascaded task relationship.

2.2.3 *Summary*

An overview of MTL models, taxonomies and classification strategies has been presented. In order to present a global perspective, the different parameter sharing techniques, parameter sharing focuses, and network architectures are now summarised.

Parameter Sharing Techniques:

- Hard Parameter Sharing: shares hidden layers across all tasks while maintaining separate output layers for each task. Examples include UberNet and Multi-linear Relationship Networks. This method is memory efficient but may suffer from insufficient knowledge transfer between tasks.
- Soft Parameter Sharing: Assigns to each task a dedicated network, with information shared via connections between them. Notable models include cross-stitch networks, gated multi-task networks, and sluice networks. This method provides flexibility but requires careful management to avoid negative task interactions.

Parameter Sharing Focuses:

- Encoder-Focused Models: Information is shared only within the encoder, with separate decoders for each task. An example is the 2C-Net, which combines compression and classification tasks using a shared encoder.
- Decoder-Focused Models: Task-specific information is shared at the decoding stage, allowing for multi-modal integration. PAD-Net is an example that combines predictions from multiple tasks for final output, requiring related tasks for optimal performance.

Task Relationship-Oriented Network Architectures:

- Cascaded Networks: Tasks are processed sequentially, with each task's output feeding into the next. This architecture does not share parameters and is closer to single-task approaches but with dependencies between tasks.
- Parallel Networks: Independent task-specific layers work in parallel, with parameter sharing in the common backbone. Suitable for tasks with shared low-level features but different complexities, like segmentation and classification.
- Interacted Networks: Task-specific networks share parameters and information to improve performance on related tasks. This architecture is effective for tasks with similar levels of complexity, such as a segmentation network that uses shared features to improve results for closely related tasks.
- Hybrid Networks: Combines elements of cascaded, parallel, and interacted architectures to handle complex task combinations. An example is a model for COVID-19 diagnosis that integrates feature extraction, lesion segmentation, and classification tasks, showing parallel, interacted, and cascaded characteristics.

Table 1: Literature MTL models classified into the three primary taxonomies described, parameter sharing techniques and focus and network architectures.

Covered Tasks	Parameter Sharing		Network Architecture
	Technique	Focus	
SLO image generation and image classification. [52].	-	-	Cascaded
FNC generation and disease classification. [53].	-	-	Cascaded
Lung nodule malignancy classification and nodule features characterisation. [54].	-	-	Cascaded
Segmentations of left atrium and atrial scars. [55]	-	-	Cascaded
CT-volumes segmentation and synthetisation. Aortic dissection detection. [56]	-	-	Cascaded
Subcutaneous and visceral fat maps prediction. [57].	Hard	Encoder	Parallel
Pancreas segmentation and its skeleton extraction. [58]	Hard	Encoder	Parallel
Skin lesion detection, classification, and segmentation. [59]	Hard	Encoder	Parallel
Classify sperm's head, vacuole, and acrosome as either normal or abnormal. [60].	Hard	Encoder	Parallel
Semantic segmentation, depth estimation, and surface normal estimation on indoor images. [61].	Soft	Encoder	Parallel
Vertebral segmentation and landmark localization. [62].	Soft	Decoder	Interacted
Chromosome joint detection, chromosome type and polarity classification. [63].	Soft	Decoder	Interacted
Infarction area segmentation and quantification. [64].	Soft	Decoder	Interacted
Lung lesion segmentation and COVID-19 infected or uninfected classification. [65].	Soft	Decoder	Interacted
Semantic Segmentation And Change Detection. [66].	Soft	Decoder	Interacted
Breast Ultrasound Image Classification and Segmentation. [67].	Hard	Encoder	Hybrid (Parallel, Cascaded)
Disease Grading, lesion segmentation, and image super-resolution. [68].	Soft	-	Hybrid (Cascaded, Interacted)
LV cavity and Myo segmentation, full LV quantification, and phase classification. [69]	Hard	Encoder	Hybrid (Cascaded, Parallel)
Vertebral localization, identification, and segmentation. [70].	Hard	Encoder	Hybrid (Parallel, Cascaded)
Hepatocellular carcinoma segmentation and classification. [71].	Hard	Decoder	Hybrid (Cascaded, Parallel, Interlaced)

This section elaborates on how different MTL architectures can be tailored to specific task sets, enhancing performance and resource efficiency. In Table 1, some recent MTL works are present and classified in terms of the discussed taxonomies.

In summary, MTL represents a compelling paradigm that exploits shared knowledge across related tasks to improve the generalisation and performance of deep neural networks. Through various architectures such as cascaded, parallel, interacting, and hybrid networks, MTL optimises the learning process by exploiting shared features and relationships inherent in different tasks. The use of parameter sharing techniques, both hard and soft, underlines MTL's ability to effectively manage and distribute computational resources, resulting in reduced memory consumption and improved learning efficiency. Encoder- and decoder-focused models, in particular, highlight the flexibility of MTL frameworks in accommodating different task requirements and promoting cross-task synergy. Despite these advantages, MTL

faces challenges such as negative transfer, task interference, and the complexity of balancing shared and task-specific components, which require ongoing research and refinement of MTL strategies.

The exploration of MTL methods reveals significant potential for advancing AI applications, particularly in domains requiring simultaneous task processing, such as computer vision, natural language processing, and multi-modal data analysis. Future research directions include improving the scalability of MTL frameworks, improving the modelling of task relationships, and developing adaptive mechanisms for dynamic task allocation. By addressing these challenges, MTL can evolve to support more complex and interdependent task scenarios, paving the way for more robust and efficient AI systems capable of handling a wide range of applications with minimal resource consumption. Overall, the integration of MTL into deep neural networks is a promising way to achieve more intelligent, adaptive and resource-efficient AI solutions.

2.3 COMPRESSED-DOMAIN IMAGE PROCESSING

Compressed domain image processing refers to the ability to perform image analysis and understanding tasks directly on their compressed representations, without the need for full decompression into the pixel domain. This approach has gained significant attention in recent years, particularly with the advent of deep learning-based image compression methods. Traditional image compression techniques, such as JPEG and JPEG2000, were primarily designed for human visual perception, and their compressed domains were not optimized for machine vision tasks. However, with the rise of DNNs, new opportunities have emerged to integrate image compression and computer vision tasks within a unified framework.

2.3.1 *Leveraging Deep Learning for Compressed Domain Processing*

The paper [72] marks a significant milestone in the exploration of compressed domain processing. The authors propose a method where the compressed representation of an image, generated by a DNN-based encoder, is directly fed into a modified version of image understanding networks, such as ResNet [73] for classification and DeepLab [74] for segmentation. The key insight is that the compressed representation, which is typically a lower-dimensional feature map, already contains relevant information for machine vision tasks. By avoiding the decompression step, the method reduces both the computational complexity and the memory footprint of the image processing pipeline.

In [72] it is shown that image classification and segmentation from compressed representations can be performed with comparable accuracy to traditional methods that operate on fully decompressed images, while significantly reducing the number of operations required. For example, their experiments show that classification from compressed representations can achieve similar accuracy to case decompressed images had been used, but with 1.5× to 2× fewer operations. Furthermore, the results show that semantic segmentation from compressed representations can be more accurate than segmentation from decompressed images, especially at aggressive compression rates. In resume, evidence is presented for this underlying idea that the compressed representations may capture semantic features that are beneficial for certain vision tasks.

2.3.2 *Joint Training for Compression and Vision Tasks*

The concept of joint training for compression and vision tasks is also explored in [72]. This approach allows the compression network to learn features that are not only useful for reconstructing the image but also for performing machine vision tasks. This methodology leads to synergistic improvements in both compression quality and task performance. For instance, [72] show that joint training can lead to better compression quality (as measured by SSIM and MS-SSIM), as well as improved classification and segmentation accuracy.

2.3.3 *Unified Frameworks for Compression and Understanding*

The authors of [75] take the idea of joint training a step further by proposing a unified framework called CodedVision. This approach uses an eight-layer deep residual network (ResNet) to extract compact feature maps (fMaps) from images, which are then used for both image compression and understanding tasks, such as classification.

The CodedVision framework consists of two main components: a Compression Engine (CE) and a Vision Task Engine (VTE). The CE compresses the fMaps using a scalar quantizer and entropy coding, while the VTE performs vision tasks, such as classification, directly on the compressed fMaps. The authors demonstrate that this approach achieves a visible improvement in compression efficiency, with a 7.8% BD-Rate gain over the state-of-the-art HEVC intra-based image compression. Additionally, the framework allows for image classification directly on the compressed domain, achieving reasonable accuracy without the need for full image reconstruction.

The CodedVision framework highlights the potential of using deep learning to create a unified system that can simultaneously optimize for both compression and understanding tasks. By sharing the feature extraction process between compression and vision tasks, the system reduces redundancy and improves efficiency.

2.3.4 *Shared Latent Representations for Compression and Classification*

Building on the ideas presented by [72] and [75], the paper [76] further explores the use of compressed domain representations for specific computer vision tasks, particularly material and texture recognition. The authors use the HyperMS-SSIM model, a variant of the variational image compression with a scale hyperprior available in [77], to generate compressed representations of images. These representations are then fed into a modified ResNet-50 classifier, referred to as cResNet-39, which is specifically designed to operate on the compressed domain.

The authors compare the performance of compressed domain classification with traditional methods that operate on fully decoded images. Their results show that the compressed domain classification achieves competitive performance in terms of Top-1 and Top-5 accuracy, while using a smaller and less complex classification model. This demonstrates the potential of compressed domain processing for efficient and effective machine vision tasks.

In [46], it is introduced a versatile framework called 2C-Net, which integrates image compression and classification within a shared deep neural network. The key idea behind 2C-Net is to extract a shared latent representation that can be used for both tasks, thereby reducing computational costs and improving efficiency.

The 2C-Net framework consists of four main modules:

- General Feature Extractor (GF-Extr): This module extracts a compact and generalized latent representation that is shared between compression and classification tasks.
- Rate Reduction (R-Red): This module reduces the bitrate of the latent representation using a hyperprior model for efficient entropy coding.
- General Feature Application (GF-App): This module contains two branches—one for image reconstruction and another for image classification using a feature-analytic classifier.
- Rate-multi-Distortion Optimization (RmD-Opt): This module optimizes the trade-off between compression ratio, reconstruction quality, and classification accuracy.

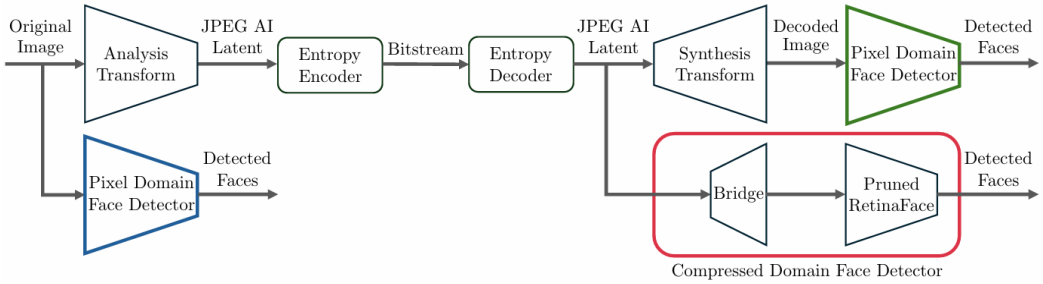


Figure 7: General compressed domain face detection framework [78]. The pixel domain detector can process faces through two pathways: directly in the pixel domain using the original image (blue block) or using a fully decoded image (green block). The proposed compressed domain architecture [78] operates in the latent space (red block), utilizing a bridge network that replaces the initial layers of the face detector before applying the synthesis transform for image decoding.

The authors demonstrate that 2C-Net outperforms conventional codecs like BPG and JPEG2000 in compression efficiency while achieving acceptable accuracy on image classification tasks without the need for full image reconstruction. For example, 2C-Net achieves 80.4% top-1 accuracy on the Caltech101 dataset and 75.1% mAP on the Pascal VOC 2012 dataset, which is close to the accuracy achieved in the pixel domain. The framework also shows strong generalization capabilities, achieving 62.8% top-1 accuracy on the large-scale ImageNet dataset.

The 2C-Net framework represents a significant step forward in the integration of image compression and classification, demonstrating that it is possible to achieve competitive performance in both tasks within a single unified framework. By sharing the latent representation between compression and classification, 2C-Net reduces redundancy and improves efficiency, making it a promising approach for real-world applications.

2.3.5 *JPEG-AI Compressed Domain Face Detection*

The authors of [78] present an innovative approach to perform face detection directly in the compressed domain, leveraging the latent representation generated by the JPEG-AI architecture. This methodology capitalises on the fact that deep learning-based image compression inherently extracts meaningful visual features, making it possible to conduct high-level computer vision tasks without requiring full image reconstruction.

Traditionally, face detection models, operate in the pixel domain, requiring computationally expensive decoding of compressed images before applying detection algorithms. The compressed domain face detection model proposed in [78] adapts the face detection model RetinaFace to work with JPEG-AI's latent features. Instead of processing decoded images, the model receives as input the feature maps produced by

the JPEG-AI encoder's analysis transform. A key component of this approach is the introduction of a "bridge" network, which aligns the JPEG AI latent representation with the feature maps expected by the RetinaFace model. By pruning the initial convolutional layers of RetinaFace, responsible for low-level feature extraction, the model efficiently processes compressed-domain representations without significant loss of accuracy. The general pipeline of the compressed domain face detection framework proposed by the authors is presented in Figure 7.

The proposed method ensures that only the essential high-level visual features extracted during the compression process are used for face detection. This optimizes both computational efficiency and detection accuracy, demonstrating that a well-designed compressed domain approach can rival conventional methods without the overhead of reconstructing the image.

Experimental evaluations demonstrate that the compressed domain RetinaFace-based detector achieves comparable accuracy to its pixel-domain counterpart while significantly reducing computational costs. The model is tested across various bitrates, showing robust performance in detecting faces without the need for full image decoding.

The evaluation consists of running the detection model on both uncompressed and compressed domain inputs and comparing the results in terms of detection accuracy, processing speed, and computational efficiency. The findings reveal that while there is a slight drop in performance at extremely low bitrates, the overall detection accuracy remains highly competitive. More importantly, the computational savings achieved by bypassing the decoding step make the method highly suitable for real-time and resource-constrained applications such as surveillance, mobile vision, and cloud-based analytics.

These results highlight the potential of learning-based compression standards, such as JPEG AI, in facilitating efficient and accurate computer vision tasks within the compressed domain. The success of this method underscores the feasibility of extending compressed domain processing beyond face detection to other object detection and recognition tasks.

2.4 OBJECT DETECTION

Object detection is a fundamental task in computer vision that involves identifying the presence and location of multiple objects of various categories within an image or video [79]. Unlike image classification, which focuses on determining the dominant object category in an image, object detection provides a more detailed understanding of a scene by localizing each object instance, typically using bounding boxes [79].

This capability is crucial for a wide range of applications, including autonomous driving [80], surveillance [81], robotics [82], and medical imaging [83].

The field has seen significant progress, evolving from traditional methods relying on handcrafted features to modern approaches leveraging the power of deep learning [79]. Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized object detection by enabling the automatic learning of complex and hierarchical features directly from image data, leading to substantial improvements in accuracy and efficiency [84]. This section will explore the foundational principles of object detection and delve into the recent advancements driven by deep learning techniques in the last years.

2.4.1 *Principles of Object Detection*

Object detection fundamentally involves two key sub-tasks: object localization and object classification [79]. Object localization aims to determine the spatial extent of each object in an image, usually by predicting bounding boxes that tightly enclose the objects. Object classification, on the other hand, involves assigning a specific category label to each of these localized objects. Together, these tasks provide a comprehensive understanding of what objects are present in an image and where they are located.

Early approaches to object detection often employed the sliding window technique [84]. This method involves systematically scanning an image with a window of a fixed size at various locations and scales. For each window, a classifier is used to determine if it contains an object of interest. While this method is intuitive, it is computationally expensive due to the large number of windows that need to be evaluated. Furthermore, selecting an appropriate window size to accommodate objects of different scales can be challenging.

In conjunction with the sliding window approach, traditional object detection systems relied on handcrafted feature extraction techniques to represent the visual characteristics of objects. Two prominent examples include the Histogram of Oriented Gradients and the Scale-Invariant Feature Transform.

The Histogram of Oriented Gradients (HOG) descriptor, introduced by Dalal and Triggs in [85], focuses on capturing the shape and appearance of objects by analysing the distribution of local gradient orientations. The image is divided into small regions called cells, and for each cell, a histogram of gradient directions is computed. These histograms are then normalized over larger spatial regions called blocks to account for variations in illumination and contrast. The resulting HOG

features, often used with a Support Vector Machine (SVM) classifier, proved effective for tasks like pedestrian detection [86].

The Scale-Invariant Feature Transform (SIFT), presented by Lowe in [87], is another influential feature extraction technique designed to detect and describe local image features that are invariant to scale, rotation, and partially invariant to changes in illumination and viewpoint. The SIFT algorithm identifies keypoints in an image by finding local extrema in a scale-space representation. For each keypoint, a descriptor is computed based on the magnitudes and orientations of gradients in its neighbourhood. SIFT features are highly distinctive and have been widely used for object recognition [88] and matching [89].

2.4.2 *Deep Learning in Object Detection*

The advent of deep learning, particularly CNNs, has brought about a paradigm shift in object detection, leading to significant breakthroughs in accuracy and performance [79]. CNNs have the ability to automatically learn complex features from raw image data through their hierarchical structure, surpassing the limitations of handcrafted features [84].

Recent advancements in deep learning-based object detection over the last five years have largely focused on refining and developing both two-stage and one-stage detectors, as well as exploring novel architectures [79].

Two-stage detectors, such as the R-CNN family (R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN), first generate a set of region proposals and then classify these proposals [90]. Faster R-CNN, which integrates a Region Proposal Network (RPN) to efficiently generate candidate regions, has become a foundational architecture for high-accuracy object detection [91]. Mask R-CNN extends Faster R-CNN by adding a branch for instance segmentation, enabling pixel-level object detection [92].

One-stage detectors, including YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector), aim for faster detection by performing localization and classification in a single forward pass [90]. The YOLO family has seen continuous improvements, with versions like YOLOv5 [93] and YOLOv8 [94] focusing on enhancing both speed and accuracy. SSD utilizes a set of default anchor boxes across different feature maps to detect objects at various scales [95].

These advancements have collectively led to more accurate, faster, and more robust object detection systems capable of handling various challenges such as occlusions, small objects, and diverse environmental conditions [79].

2.4.3 *Object Detection in Biomedical Images*

Object detection techniques have found increasing applications in the field of biomedical image analysis, aiding in tasks such as disease diagnosis and the study of cellular structures [96]. Deep learning-based object detection models are being utilized across various medical imaging modalities, including MRI, CT scans, pathology images, and microscopy [96], [97]. These methods help in identifying and localizing regions of interest [98], such as lesions [99], tumors [100], and other clinically relevant objects [101].

One specific area of focus is the detection of subcellular organelles, such as mitochondria, in microscopy images [83]. Mitochondria, dynamic organelles crucial for cellular function, are often studied to understand various diseases [102]. Automated detection and segmentation of mitochondria in microscopy images, particularly electron microscopy (EM) and fluorescence microscopy, is a challenging but important task due to the complex shapes and dense arrangements of these organelles [103].

Several deep learning architectures have been employed for mitochondria detection. Faster R-CNN has been used for detecting cellular organelles, including mitochondria, in microscopy images [83]. However, some studies have focused on segmentation rather than pure object detection, using architectures like U-Net with different backbones (e.g., ResNet18) to accurately identify mitochondrial networks [104]. MitoSegNet, a pre-trained deep learning segmentation model, has shown superior performance compared to traditional feature-based methods for quantifying mitochondrial morphology in fluorescence microscopy images [102]. These advancements in object detection and segmentation techniques are significantly enhancing our ability to analyse mitochondrial morphology and function, providing valuable insights into various biological and pathological processes [105].

2.5 METRICS

To assess the performance of the models presented in this work, metrics are needed to evaluate both object detection performance and image compression efficiency. The following sections present established metrics from the literature that are used in this work to evaluate these aspects.

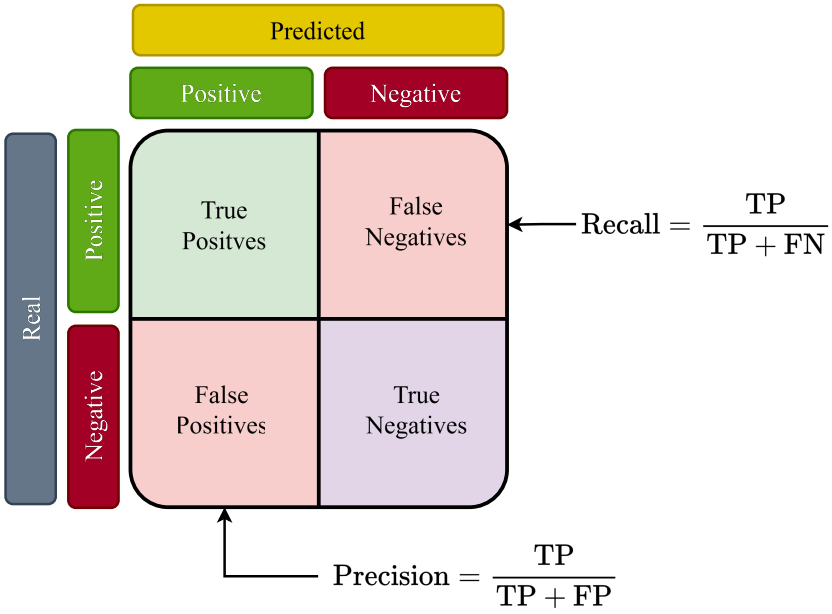


Figure 8: Precision and Recall metrics defined using the Confusion Matrix values.

2.5.1 *Object Detection Metrics*

The assessment and benchmarking of models used in this work follow standard practices in the field of object detection. As described by [106], the Average Precision (AP) score is the predominant metric for measuring detection accuracy in object detection algorithms. To understand this metric, it is essential to first establish several fundamental object detection concepts.

2.5.1.1 *Classification Metrics*

To evaluate detection quality, model outputs must be classified as either correct or incorrect using three key metrics: True Positives (TP), False Positives (FP), and False Negatives (FN). True Positives represent the number of existing objects correctly detected by the model. False Positives indicate the number of detections that do not correspond to any object of interest. False Negatives count the number of objects of interest present in the image that the model failed to detect. While True Negatives (TN) are commonly used in classification problems, they are not considered in object detection since there are infinite positions in an image where objects of interest are not present.

As noted by [106], because TN is not considered in object detection evaluation, metrics that incorporate this indicator (such as the True Positive Rate, the False Positive Rate, and the Receiver-operating characteristic curve) are not utilized. Instead, model classification is evaluated using Precision and Recall. As illustrated in Figure 8, Precision represents the ratio between correctly detected objects and

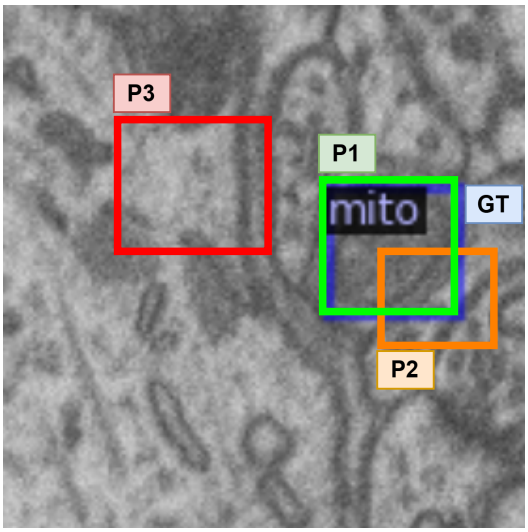


Figure 9: Image patch took from the *Lucchi++* dataset [107] with three mitochondria detection prediction examples.

all detections made by the model, effectively measuring detection accuracy. Recall, conversely, represents the proportion of correct detections among all objects of interest present in the image. The mathematical definitions of Precision (p) and Recall (r) in terms of TP, FP, and FN are presented in Equations (1) and (2), respectively.

$$p = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{1}$$

$$r = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

2.5.1.2 *Intersection Over Union*

While the previous section defined metrics for assessing detection classification, it is crucial to establish how detections are categorized as correct or incorrect. In Figure 9, three mitochondria detection predictions are represented by bounding boxes P1, P2, and P3, with the actual mitochondria location represented by the GT (Ground Truth). This example raises an important question: How do we determine which detections are correct?

The case of prediction P3 is straightforward, as it is completely removed from the mitochondria position. However, P1 and P2 require more nuanced analysis. While P1 almost perfectly aligns with GT, P2 partially overlaps with the mitochondria. To objectively determine which detections should be considered correct, the Intersection Over Unit (IoU) metric is employed.

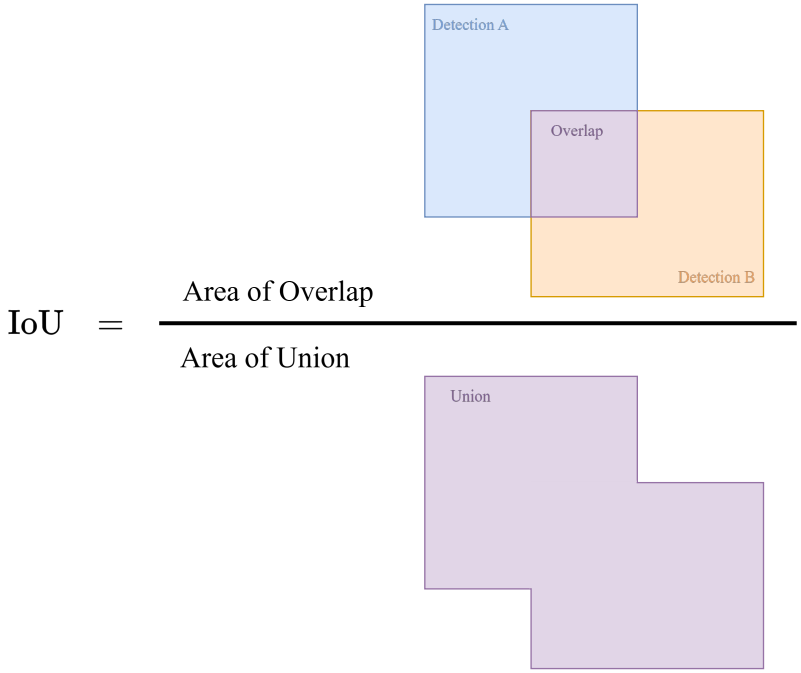
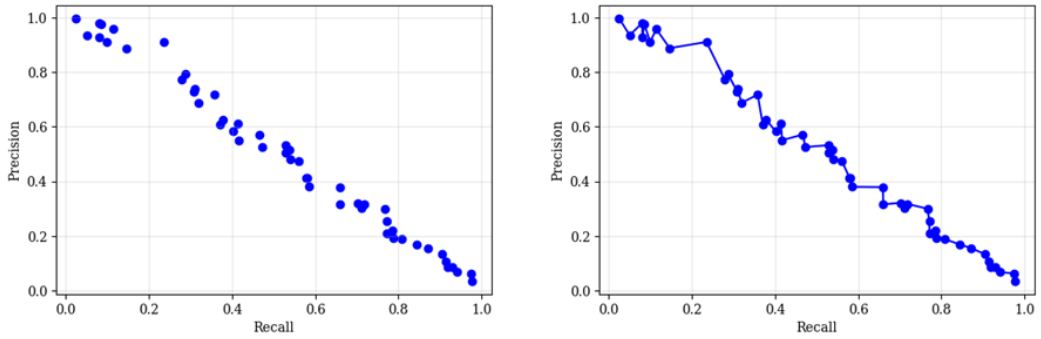


Figure 10: Visual representation of IoU metric.

The IoU metric quantifies the overlap between two bounding boxes relative to their total combined area. As illustrated in Figure 10, the IoU metric is calculated as the ratio between the overlapping area of two bounding boxes and their union area. Mathematically IoU can be presented as in Equation (3), where D_A and D_B represent the areas of bounding boxes A and B, respectively.

$$\text{IoU} = \begin{cases} \frac{D_A \cap D_B}{D_A \cup D_B}, & D_A \cup D_B \neq 0 \\ 0, & D_A \cup D_B = 0 \end{cases} \tag{3}$$

To classify detections, each prediction is compared with the GT using the IoU metric. Detections achieving an IoU value above a predetermined threshold τ_{IoU} are classified as correct detections and contribute to the TP count, while others are deemed incorrect and count as FP. Any GT detections not matched with correct predictions are counted as FN. In the example from Figure 9, the approximate IoU values when compared with GT are: IoU$P1 \approx 0.75$, IoU$P2 \approx 0.3$, and IoU$P3 = 0$. The conventional τIoU value is 0.5. Consequently, P1 is classified as a TP, while P2 and P3 are classified as FP. For each object detection problem, users can select the appropriate τIoU threshold to define the confidence level required for correct detection classification.



(a) Precision-Recall points at different class thresholds τ_{Class}.

(b) Precision-Recall curve for points at different class thresholds τ_{Class}.

Figure 11: Precision-Recall example curve of an object detection model using different class thresholds τ_{Class}.

2.5.1.3 *Average Precision Score*

With the concepts of classification metrics and IoU established, we can now introduce the AP score. Precision-Recall curves are one of the most common methods for assessing object detection model performance, as they combine both Precision and Recall metrics. Figure 11a presents example Precision-Recall points from a hypothetical object detection model. Each point represents Precision and Recall values obtained at a specific class threshold τ_{Class}. This threshold determines which detections are considered valid based on the model's confidence that the detected object belongs to the target class. In this case, τ_{Class} establishes the required confidence level for classifying an image patch within a bounding box as a mitochondria.

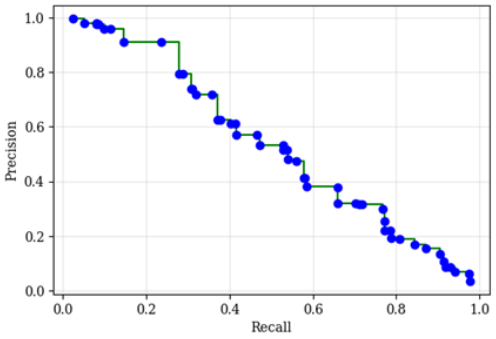
Figure 11b shows the Precision-Recall curve created from the points in Figure 11a. The curve exhibits a zig-zag pattern that complicates analysis. To address this, the curve is smoothed using the 11-point interpolation technique described by [106]. The first step involves filtering and interpolating the Precision-Recall curve using the following expression:

$$p_{\text{Filtered}}(r) = \max_{\hat{r}:\hat{r}>r} \{p(\hat{r})\}, \tag{4}$$

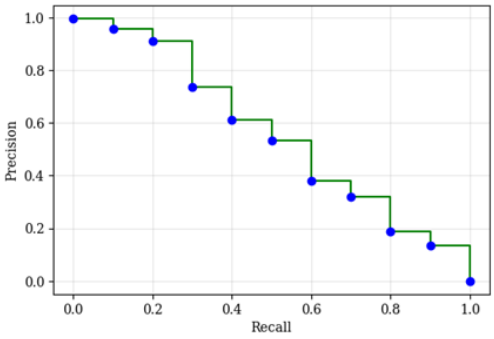
where $p(r)$ represents the original Precision value at Recall value r, $p_{\text{Filtered}}(r)$ corresponds to the filtered and interpolated Precision value at the same Recall value r, and $\hat{r}$ represents any Recall value greater than r. In Figure 12a, a curve created with the values of $p_{\text{Filtered}}(r)$ is presented.

The second step samples 11 equally spaced points from the filtered curve and creates a step curve by maintaining each value until the next point. The resulting filtered and sampled curve is shown in Figure 12b.

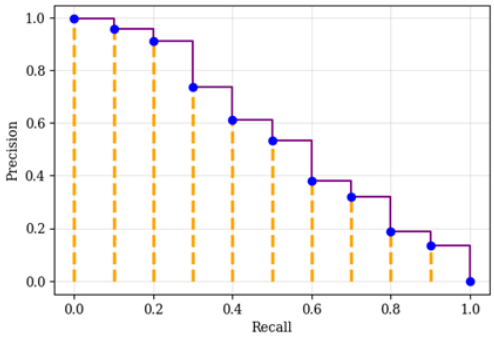
The AP score is defined as the weighted average of Precision values in the Precision-Recall curve. Following [106], this work calculates the AP score by considering



(a) Filtered Precision-Recall curve for points at different class thresholds τ_{Class}.



(b) Filtered and sampled Precision-Recall curve for points at different class thresholds τ_{Class}.



(c) Precision-Recall points used to calculated the AP score.

Figure 12: Filtered and sampled Precision-Recall example curves of an object detection model using different class thresholds τ_{Class} used to calculated the AP score.

the maximum precision value $p_{\text{Filtered}}(r)$ whose Recall value exceeds r, using the precision values from the filtered and sampled curve. These values are illustrated in Figure 12c. Mathematically, the AP score can be defined by

$$\text{AP} = \frac{1}{n_S} \sum_{r \in S} p_{\text{Filtered}}(r), \tag{5}$$

where S corresponds the recall points at which $p_{\text{Filtered}}(r)$ is sampled and n_S corresponds to the number of elements in S. Following [106], in this work $S = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

Since the AP score depends directly on the model's Precision and Recall values, which vary according to the chosen IoU threshold τ_{IoU}, different AP scores can be obtained for different τ_{IoU} values. This work evaluates model performance using the following AP score variations:

- **AP50:** Computes the AP score using a $\tau_{\text{IoU}} = 0.50$.
- **AP75:** Computes the AP score using a $\tau_{\text{IoU}} = 0.75$.
- **AP:** Also known as mean AP (mAP). Calculated using multiple τ_{IoU} values. Following the COCO evaluation methodology [108], this work uses 10 thresholds from 0.5 to 0.95 with 0.05 intervals.

- **APs:** Computes the AP score for objects with an area lower than 32^2 pixels.
- **APm:** Computes the AP score for objects with an area greater than 32^2 and lower than 96^2 pixels.
- **APl:** Computes the AP score for objects with an area greater than 96^2 pixels.

2.5.1.4 *Bjontegaard Delta-based Metrics*

When comparing different compression algorithms over Bits per Pixel (bpp) ranges, Rate-Distortion (RD) curves serve as a valuable tool [109]. These representations illustrate the trade-off between bitrate and distortion, offering insight into the overall performance of different encoding settings.

Two commonly used metrics for assessing the efficiency of compression algorithms through RD curves are introduced in [110]. These metrics condense rate-distortion performance into a single, interpretable value: the Bjontegaard Delta (BD)-Rate and the BD-PSNR. The BD-Rate, depicted in Figure 13a, quantifies the average bitrate difference between two encoders while maintaining the same level of distortion. Mathematically, it is expressed as:

$$\text{BD-Rate} = \frac{1}{D_{\max} - D_{\min}} \int_{D_{\min}}^{D_{\max}} [R_A(\delta) - R_B(\delta)] d\delta, \tag{6}$$

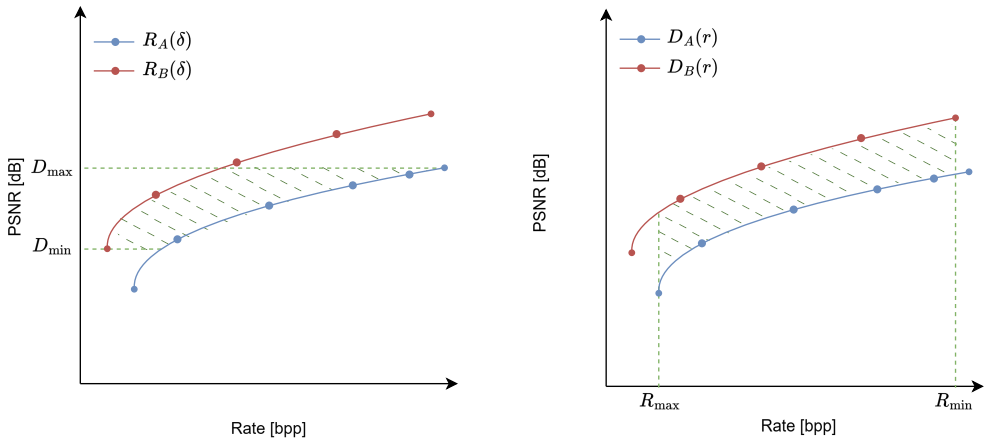
where $R_A(\delta)$ and $R_B(\delta)$ denote the bitrates of encoders A and B at a given distortion level δ. The overlap of the two RD curves is illustrated by the dashed lines in Figure 13a, with $D_{\max}$ and $D_{\min}$ indicating the upper and lower bounds of distortion, respectively. These values are determined as $D_{\min} = \max(\min D_A, \min D_B)$ and $D_{\max} = \min(\max D_A, \max D_B)$, with δ acting as the integration variable. A negative BD-Rate value signifies that encoder B is more efficient than encoder A, as it requires a lower bitrate to achieve the same quality.

Similarly, BD-PSNR, shown in Figure 13b, measures the average difference in quality, expressed in Peak Signal-to-Noise Ratio (PSNR), between two RD curves at the same bitrate. It is defined as:

$$\text{BD-PSNR} = \frac{1}{R_{\max} - R_{\min}} \int_{R_{\min}}^{R_{\max}} [D_A(r) - D_B(r)] dr, \tag{7}$$

where $D_A(r)$ and $D_B(r)$ represent the distortion values of encoders A and B at a given bitrate r. A positive BD-PSNR value indicates that encoder A delivers superior quality compared to encoder B for the same bitrate, whereas a negative value suggests the opposite.

Beyond BD-Rate and BD-PSNR, additional BD-based metrics have been introduced. In [111], the BD-Top-1 metric was proposed to evaluate model efficiency using Rate-Performance curves. Likewise, [112] introduced BD-based metrics to



(a) BD-Rate. (b) BD-PSNR.

Figure 13: Rate-distortion comparison of two codecs using (a) BD-Rate and (b) BD-PSNR. The curves depict the distortion, D, in PSNR as a function of bitrate, R, where the reference codec is in blue and the evaluated codec in red.

assess bitrate savings for different quality indicators, including PSNR, VMAF, and MOS. Building on these methodologies, this work introduces new BD-based metrics specifically tailored for evaluating detection algorithms under various image compression scenarios.

Following the same rationale, for object detection models, the BD-AP metric and its variants—BD-AP50, BD-AP75, BD-APs, BD-APm, and BD-APl are herein introduced to quantify the average difference in AP score across different bitrates. BD-AP is formulated as:

$$\text{BD-AP} = \frac{1}{R_{\max} - R_{\min}} \int_{R_{\min}}^{R_{\max}} [\text{AP}_A(r) - \text{AP}_B(r)]dr, \tag{8}$$

where APA and APB denote the AP scores achieved by detection models when processing images compressed by algorithms A and B, respectively. Similarly, the BD-based metrics for different AP variations are defined as:

$$\text{BD-AP50} = \frac{1}{R_{\max} - R_{\min}} \int_{R\text{min}}^{R_{\max}} [\text{AP50}_A(r) - \text{AP50}_B(r)]dr, \tag{9}$$

$$\text{BD-AP75} = \frac{1}{R_{\max} - R_{\min}} \int_{R_{\min}}^{R_{\max}} [\text{AP75}_A(r) - \text{AP75}_B(r)]dr, \tag{10}$$

$$\text{BD-APs} = \frac{1}{R_{\max} - R_{\min}} \int_{R_{\min}}^{R_{\max}} [\text{APs}_A(r) - \text{APs}_B(r)]dr, \tag{11}$$

$$\text{BD-APm} = \frac{1}{R_{\max} - R_{\min}} \int_{R_{\min}}^{R_{\max}} [\text{APm}_A(r) - \text{APm}_B(r)]dr, \tag{12}$$

$$\text{BD-APl} = \frac{1}{R_{\max} - R_{\min}} \int_{R_{\min}}^{R_{\max}} [\text{APl}_A(r) - \text{APl}_B(r)]dr, \tag{13}$$

where the notation follows that of the standard AP metric variations. These BD-based metrics provide a more comprehensive evaluation of detection models' performance on given operational range (compression level/condition, in the case).

2.5.2 *Visual Quality Metrics*

To evaluate compression efficiency, objective quality assessment metrics are necessary for comparing compressed images with their original versions at specific target bitrates. This work primarily utilizes the learning-based image codec JPEG-AI VM 6.1, which compresses input images at five default target bitrates: 0.12 bpp, 0.25 bpp, 0.50 bpp, 0.75 bpp, and 1.00 bpp. To benchmark JPEG-AI's compression efficiency against other codecs, VVC and HEVC were also employed using quantization parameters (QPs) that achieve comparable target bitrates.

The "JPEG AI Common Training and Test Conditions" document [113] specifies seven objective quality evaluation metrics for assessing compression performance: 0.12 bpp, 0.25 bpp, 0.50 bpp, 0.75 bpp, and 1.00 bpp. The reference implementation for all objective quality assessment metrics is available at: https://gitlab.com/wg1/jpeg-ai/jpeg-ai-qaf. The following sections introduce each of these metrics in detail.

2.5.2.1 *Multi-Scale Structural Similarity*

Multi-Scale Structural SIMilarity (MS-SSIM) [114] is a leading image quality evaluation algorithm that computes relative quality scores by comparing details across multiple resolutions, making it particularly effective for learning-based image codecs. Compared to single-scale metrics such as SSIM, MS-SSIM offers greater flexibility by accounting for variations in image resolution and viewing conditions. The metric employs an image synthesis-based approach to calibrate parameters that weight the relative importance of different scales. Higher MS-SSIM scores indicate better image quality.

Mathematically, the MS-SSIM index extends the single-scale SSIM by incorporating multiple levels of image resolution. Given two image signals X and Y, MS-SSIM is computed as:

$$\text{MS-SSIM}(X,Y) = [l_M(X,Y)]^{\alpha_M} \prod_{j=1}^{M} [c_j(X,Y)]^{\beta_j} [s_j(X,Y)]^{\gamma_j}, \qquad (14)$$

where:

- $l_M(X,Y)$ is the luminance comparison function at the coarsest scale M,

- $c_j(X,Y)$ and $s_j(X,Y)$ are the contrast and structure comparison functions at scale j,
- α_M, β_j, and γ_j are parameters that control the relative importance of luminance, contrast, and structure across different scales.

The individual comparison functions are defined as:

$$l(X,Y) = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1}, \tag{15}$$

$$c(X,Y) = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2}, \tag{16}$$

$$s(X,Y) = \frac{\sigma_{XY} + C_3}{\sigma_X\sigma_Y + C_3}, \tag{17}$$

where μ_X and μ_Y are the mean intensities, σ_X^2 and σ_Y^2 are the variances, and σ_{XY} is the covariance of images X and Y. The constants C_1, C_2, and C_3 are small stabilizing factors.

To compute MS-SSIM, the input images undergo successive low-pass filtering and downsampling by a factor of 2 at each iteration, producing multiple resolution levels. The final quality score is obtained by aggregating the SSIM components across these scales, providing a more comprehensive assessment of image fidelity.

2.5.2.2 *Information Content Weighted Structural Similarity Measure*

Information Content Weighted Structural Similarity Measure (IW-SSIM) [115] enhances the structural similarity index through information content weighted pooling. This metric operates on the principle that natural image viewing should employ perceptual weights proportional to local information content. It utilizes advanced statistical models of natural images to derive optimal weights, which are then combined with multi-scale structural similarity measures. This approach achieves superior correlation performance with subjective scores from established databases.

Mathematically, the IW-SSIM measure is defined as follows. Given two corresponding i^{th} local image patches at the j^{th} scale from the reference and distorted images, denoted as $X_{i,j}$ and $Y_{i,j}$, the j^{th} scale Information Content Weighted multiscale structural similarity index (IW-SSIM$_j$) is computed as:

$$\text{IW-SSIM}_j(X_j, Y_j) = \frac{\sum_i w_{i,j} c(X_{i,j}, Y_{i,j}) s(X_{i,j}, Y_{i,j})}{\sum_i w_{i,j}} \tag{18}$$

for $j = 1, \ldots, M-1$, and

$$\text{IW-SSIM}_j(X_{i,j}, Y_{i,j}) = \frac{1}{N_j} \sum_i l(X_{i,j}, Y_{i,j}) c(X_{i,j}, Y_{i,j}) s(X_{i,j}, Y_{i,j}) \tag{19}$$

where $l(X_{i,j}, Y_{i,j})$, $c(X_{i,j}, Y_{i,j})$, and $s(X_{i,j}, Y_{i,j})$ are defined in Equations (15) to (17), respectively.

The local information content weight at a spatial location i and scale j is defined as:

$$w_{i,s} = \log\left(1 + \frac{\sigma_X^2}{\sigma_n^2}\right) + \log\left(1 + \frac{\sigma_Y^2}{\sigma_n^2}\right) - \log\left(1 + \frac{\sigma_{XY}^2}{\sigma_n^2}\right) \tag{20}$$

where σ_n^2 represents the variance of the perceptual noise model. The final overall IW-SSIM measure is computed as:

$$\text{IW-SSIM}(X, Y) = \prod_{j=1}^{M} (\text{IW-SSIM}_j(X_j, Y_j))^{\beta_j} \tag{21}$$

where β_j values were obtained through psychovisual experiments, and M represents the number of scales used in the multi-scale SSIM framework.

This formulation ensures that regions with higher information content contribute more significantly to the overall image quality assessment, aligning the metric more closely with human visual perception.

2.5.2.3 *Video Multimethod Assessment Fusion*

Video Multimethod Assessment Fusion (VMAF) [116], developed by Netflix, specifically targets artifacts created by compression and rescaling. The metric estimates quality scores by combining multiple quality assessment algorithms using a support vector machine (SVM). Although primarily designed for video assessment, VMAF version 2.2.1 has demonstrated effective performance in evaluating single images, particularly with learning-based image codecs. The metric requires input images in YUV colour space format; consequently, PNG images (RGB colour space) are converted to YUV 4:4:4 10-bit format using FFMPEG (BT.709 primaries). Higher VMAF scores indicate superior image quality.

Mathematically, the VMAF score is computed as follows. Let X denote a reference video and Y denote a distorted video. VMAF employs a set of n elementary quality metrics, denoted as $M_i(X, Y)$ for $i = 1,2,\ldots,n$, where each metric captures different aspects of perceptual quality. These include:

- Visual Information Fidelity (VIF)
- Detail Loss Metric (DLM)

- Motion feature (temporal difference between adjacent frames)

The final VMAF score $S(X,Y)$ is obtained by applying a trained Support Vector Machine (SVM) regression model:

$$S(X,Y) = \mathbf{w}^T \mathbf{M}(X,Y) + b, \tag{22}$$

where $\mathbf{M}(X,Y) = [M_1(X,Y), M_2(X,Y), \ldots, M_n(X,Y)]^T$ is the feature vector of elementary metrics, $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ represents the learned weights assigned to each metric, and b is the bias term.

The weights $\mathbf{w}$ and bias b are optimized by training on subjective quality scores obtained from human opinion studies, such as those conducted on the NFLX Video Dataset. The optimization follows the standard formulation of SVM regression, minimizing the error between predicted scores and subjective scores while maintaining generalization.

By leveraging multiple quality metrics and fusing them intelligently using machine learning, VMAF aims to provide a more robust and accurate assessment of video quality, closely aligning with human perception.

2.5.2.4 *Visual Information Fidelity*

Visual Information Fidelity (VIF) [117] quantifies the loss of human-perceived information during degradation processes such as image compression. Operating in the wavelet domain, VIF leverages natural scene statistics to evaluate information fidelity and relates to the Shannon mutual information between the degraded and original pristine images. Experimental results demonstrate strong correlation between VIF metric values and human perception, including for learning-based image codecs. Higher VIF scores indicate better image quality.

Mathematically, VIF is defined as the ratio of mutual information retained in the distorted image relative to the reference image. Given a reference image modeled as a stochastic source, let $I(\mathbf{C};\mathbf{E})$ denote the mutual information between the reference image signal $\mathbf{C}$ and the output of the human visual system (HVS) channel $\mathbf{E}$. Similarly, let $I(\mathbf{C};\mathbf{F})$ represent the mutual information between $\mathbf{C}$ and the corresponding degraded image $\mathbf{F}$ processed through the HVS channel. The VIF metric is then given by:

$$VIF = \frac{\sum_j I(\mathbf{C}_j;\mathbf{F}_j)}{\sum_j I(\mathbf{C}_j;\mathbf{E}_j)} \tag{23}$$

where the summation is performed over different wavelet subbands j. Higher VIF values indicate better image quality, as more information from the reference image

is retained in the distorted version. Experimental results demonstrate a strong correlation between VIF metric values and human perception.

2.5.2.5 *PSNR-HVS-M*

PSNR-HVS-M [118] is an efficient quality model based on the human visual system (HVS) that utilizes Discrete Cosine Transform (DCT) basis functions. The model processes images in 8×8 pixel blocks, incorporating both the contrast sensitivity function (CSF) [119] and between-coefficient contrast masking.

For an image block, the weighted energy of DCT coefficients is computed as:

$$E_w(X) = \sum_{i=0}^{7} \sum_{j=0}^{7} C_{ij} X_{ij}^2, \tag{24}$$

where X_{ij} is the DCT coefficient at position (i,j), and C_{ij} is a correcting factor derived from the CSF.

The masking effect for an image block D is then given by:

$$E_m(D) = \frac{E_w(D) \cdot \delta(D)}{16}, \tag{25}$$

where $\delta(D)$ accounts for local variance adjustment based on surrounding blocks.

To incorporate contrast masking into PSNR computation, the visible difference between DCT coefficients of the original and distorted image blocks, X_e and X_d, is determined as:

$$\Delta X_{ij} = \begin{cases} 0, & \text{if } i = j = 0, \\ X_{e,ij} - X_{d,ij}, & \text{if } |X_{e,ij} - X_{d,ij}| \leq \frac{C_{ij} E_{\max}}{64}, \\ X_{e,ij} - X_{d,ij} - \frac{C_{ij} E_{\max}}{64} & \text{otherwise,} \end{cases} \tag{26}$$

where $E_{\max} = \max(E_m(X_e), E_m(X_d))$ represents the maximum masking effect.

Finally, PSNR-HVS-M is computed similarly to PSNR but using a modified Mean Squared Error (MSE) that accounts for contrast masking and CSF:

$$\text{PSNR-HVS-M} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}_H} \right), \tag{27}$$

where MSE_H is the MSE adjusted for HVS characteristics.

This modification improves correlation with human perception by more accurately modeling perceptual distortions in image quality assessment.

2.5.2.6 *Normalized Laplacian Pyramid*

The Normalized Laplacian Pyramid (NLPD) [120] metric addresses two key aspects of the human visual system: local luminance subtraction and local contrast gain control. The metric employs Laplacian pyramid decomposition, where each image x is subjected to local mean removal, followed by local gain control.

First, at each scale k, a local mean estimate $f_L(x_N)$ is subtracted from each pixel x_i, yielding the intermediate representation:

$$z_i = x_i - f_L(x_N) \tag{28}$$

where x_N denotes the local neighborhood of x_i.

Next, a contrast normalization step is performed by dividing each coefficient by a local amplitude estimate $f_C(z_N)$:

$$y_i = \frac{z_i}{f_C(z_N)} \tag{29}$$

where the local amplitude is computed as:

$$f_C^{(k)}(z_N) = \sigma^{(k)} + \sum_{j \in N} p_j^{(k)} |z_j^{(k)}| \tag{30}$$

where $p_j^{(k)}$ are scale-dependent weights optimized for statistical independence, and $\sigma^{(k)}$ ensures numerical stability by preventing division by zero.

The quality assessment is then performed in this transformed domain by computing the root mean squared error (RMSE) between the distorted ($\tilde{y}^{(k)}$) and reference ($y^{(k)}$) images:

$$D(x, \tilde{x}) = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{\sqrt{N_s^{(k)}}} \| y^{(k)} - \tilde{y}^{(k)} \|_2 \tag{31}$$

where $N_s^{(k)}$ is the number of coefficients at scale k. This formulation ensures that lower NLPD scores correspond to better image quality, as it measures the perceptual distortion in the normalized Laplacian domain.

2.5.2.7 *Feature Similarity*

The Feature Similarity (FSIM) metric [121] evaluates image quality by analyzing two complementary low-level features that reflect distinct aspects of the human visual system:

1. Phase Congruency (PC): A dimensionless feature that captures the significance of local structure;

2. Gradient Magnitude (GM): A feature that accounts for contrast information.

Phase congruency is a contrast-invariant measure of local structure. It is based on the principle that perceptually significant image features occur at points where different frequency components of an image are maximally in phase. The phase congruency at a pixel x is computed as:

$$PC(x) = \frac{\sum_n E_n(x)}{\sum_n A_n(x) + \epsilon} \tag{32}$$

where:

- $E_n(x)$ is the local energy at scale n,
- $A_n(x)$ is the amplitude at scale n,
- ϵ is a small constant to avoid division by zero.

The local energy is given by:

$$E_n(x) = \sqrt{F_n^2(x) + H_n^2(x)} \tag{33}$$

where $F_n(x)$ and $H_n(x)$ are the responses of even-symmetric (cosine) and odd-symmetric (sine) wavelets (such as log-Gabor filters) at scale n. The amplitude $A_n(x)$ is computed as:

$$A_n(x) = \sqrt{F_n^2(x) + H_n^2(x)} \tag{34}$$

To ensure accurate feature localization, the 2D phase congruency map is computed by applying filters at multiple orientations and combining the results.

Gradient magnitude is used to capture contrast variations in an image. It is computed as:

$$GM(x) = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{35}$$

where $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are the horizontal and vertical gradients, respectively. These gradients can be obtained using Sobel, Prewitt, or Scharr operators.

Given a reference image I_r and a distorted image I_d, let $PC_r(x)$ and $PC_d(x)$ represent their respective phase congruency maps at pixel x, and $GM_r(x)$ and $GM_d(x)$ their gradient magnitude maps. The similarity functions for PC and GM are defined as:

$$S_{PC}(x) = \frac{2PC_r(x)PC_d(x) + T_1}{PC_r^2(x) + PC_d^2(x) + T_1} \tag{36}$$

$$S_{GM}(x) = \frac{2GM_r(x)GM_d(x) + T_2}{GM_r^2(x) + GM_d^2(x) + T_2} \tag{37}$$

where T_1 and T_2 are small positive constants to improve numerical stability.

The overall similarity at each pixel is computed as:

$$S(x) = S_{PC}(x)^{\alpha} \cdot S_{GM}(x)^{\beta} \tag{38}$$

where α and β control the relative importance of the two features, often set to 1 for simplicity.

To obtain the final FSIM score, a weighted pooling approach is used, employing PC as a weighting function:

$$FSIM = \frac{\sum_x S(x) \cdot \max(PC_r(x), PC_d(x))}{\sum_x \max(PC_r(x), PC_d(x))} \tag{39}$$

This work employs the color version of FSIM (FSIM_C), which extends FSIM by incorporating chromatic information. Higher FSIM values indicate better perceptual image quality.

2.6 DATASETS

This study utilized three distinct electron microscopy image datasets. The first dataset, *Lucchi++*, introduced in [107], is derived from the EPFL Hippocampus dataset, as described in [122]. This dataset consist of images that were taken from a $5 \times 5 \times 5\,\mu m$ section of the hippocampus of mouse brain using a focused ion beam scanning electron microscopy (FIB-SEM) technique. The *Lucchi++* dataset consists of two stacks from the EPFL Hippocampus dataset, each comprising 165 images with a resolution of 1024×768 pixels. These stacks are commonly employed as separate training and testing sets for evaluating mitochondria detection algorithms. In this study, 24.25% of the training stack was employed as validation set. The detailed dataset split is shown in Table 9 of Appendix A. To enhance the consistency of mitochondria membrane annotations and address any misclassifications in the ground truth labels, the dataset underwent a rigorous re-annotation process.

Initially, a senior biologist manually corrected mitochondria membrane annotations using in-house annotation software. Subsequently, two neuroscientists independently reviewed the corrected annotations to assess membrane consistency. Discrepancies between the reviewers were resolved by the biologist, who refined the

Table 2: Dataset bounding box size distribution for each image set of the *Lucchi++* dataset.

Set	Bounding Box Size Distribution		
	Small	Medium	Large
Training	190 (9.11%)	1649 (79.01%)	248 (11.88%)
Validation	60 (8.70%)	545 (78.98%)	85 (12.32%)
Test	175 (6.44%)	2186 (80.52%)	354 (13.04%)

annotations until consensus was achieved. The biologist's annotations were highly precise, requiring only minimal adjustments after the neuroscientists' review.

To address misclassifications, the biologist meticulously examined each image slice within the two Hippocampus stacks for missing or incorrectly labeled mitochondria. These corrections were again reviewed by the neuroscientists, ensuring agreement. In certain instances, identifying structures as partial mitochondria necessitated examining adjacent sections within the image stacks. Figure 14 shows examples of the images taken from this dataset with the annotated mitochondria.

In order to perform object detection, it is necessary to have bounding boxes around each of the objects of interest. For the *Lucchi++* dataset, the annotations consist of binary masks indicating mitochondria presence. To adapt them to the detection task, it is necessary to create bounding boxes around the objects in the mask. However, in [123], the author noticed that the masks of some mitochondria were connected by a few pixels, creating a single object that generates a single bounding box for two different mitochondria. This issue was addressed by the author, who increased the number of annotated mitochondria by 2.9% for the *Lucchi++* dataset. These corrected annotations were used in this work for training the object detection models.

In this study, as a proof of concept, the proposed method was applied only to the *Lucchi++* dataset. This dataset was chosen due to its smaller image size compared to others, which reduces memory requirements for training, enables parallelization, and shortens training time. To analyse the performance the object detection models presented in this study in terms of the object detection metrics presented in Section 2.5, it is necessary to classify the bounding boxes by its size. In Table 2 the classification of the bounding boxes of each set in the *Lucchi++* dataset is presented.

The second dataset used in this study is referred to as *Kasthuri++* [107], which comprises two adjacent volumes designated for training and testing. The training stack images were randomly split in 85% and 15% to create the training and validation sets. The training stack consists of 85 images, each with dimensions of

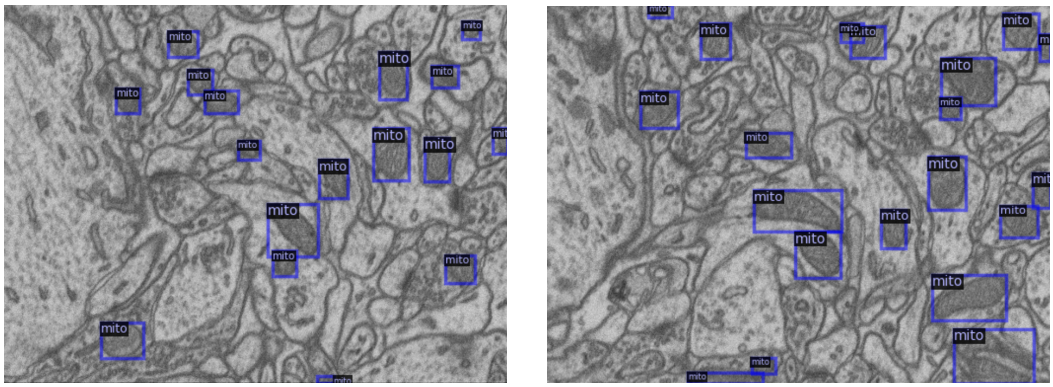


Figure 14: Samples from the *Lucchi++* dataset [107] with annotated mitochondria.

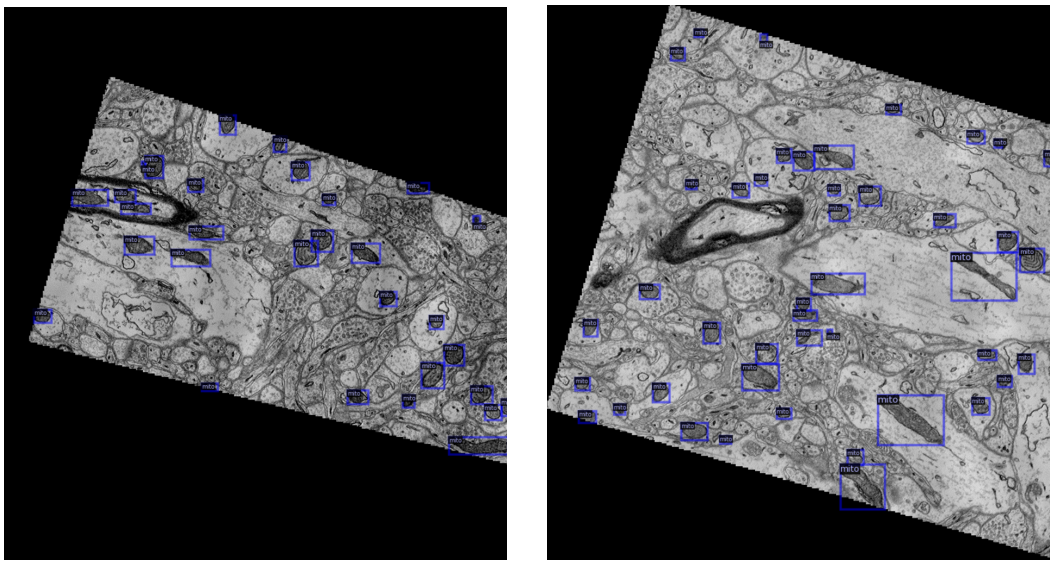


Figure 15: Random samples taken from the *Kasthuri++* Dataset [107] with annotated mitochondria.

1463 × 1613, while the test stack includes 75 images with dimensions of 1334×1553 pixels. Both stacks have a resolution of $3 \times 3 \times 30$ nm per voxel.

The *Kasthuri++* dataset was derived from the 3-cylinder mouse cortex volume described in [124]. It represents dense mammalian neuropil from layers 4 and 5 of the S1 primary somatosensory cortex, imaged using serial section electron microscopy (ssEM). As with the *Lucchi++* dataset, inconsistencies in the mitochondria segmentation masks, particularly in membrane annotations, were identified. To address these issues, the authors engaged experts to re-annotate two neighboring sub-volumes following the same methodology employed for the *Lucchi++* dataset. Figure 15 shows examples of the images taken from this dataset with the annotated mitochondria.

The Fast-EM dataset [125], hereafter referred to as *Delmic*, was provided by Delmic company through a collaborative research initiative with the Multimedia Signal Processing Group at Instituto de Telecomunicações - Leiria Branch. This collection consists of 21 high-resolution images, each measuring 6400 × 6400 pix-

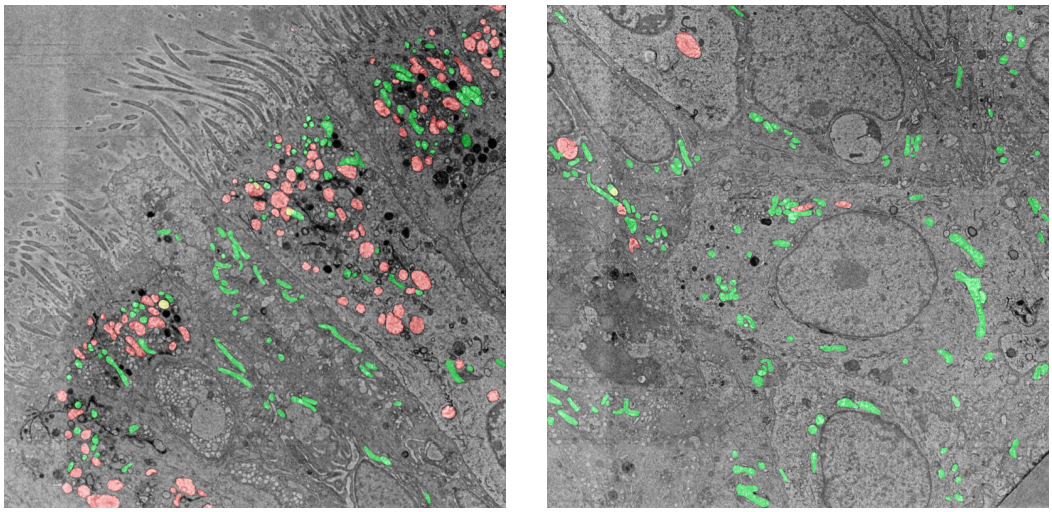


Figure 16: Random samples taken from the *Delmic* Dataset [125] with annotated mitochondria using a different colour for each class.

els, complemented by manually generated segmentation masks that identify two categories of mitochondria: healthy and swollen. The images were captured using the Fast-EM technique, which employs parallel scanning with 64 electron beams to enable high-throughput imaging of larger electron microscopy samples than conventional methods would allow.

Despite its efficiency in acquisition speed, the Fast-EM approach produces a distinctive grid pattern in the resultant images, with each square corresponding to a different beam or lens in the system. The technical specifications of the Fast-EM implementation include a square 8 × 8 multiprobe arrangement totalling 64 beams, 4 nm pixel size, 3.2 μm beamlet pitch, a field of view measuring 25.6 μm × 25.6 μm across all 64 beams, and a typical substrate dimension of 14 mm × 14 mm. Figure 16 illustrates a sample image from the dataset with various mitochondria segmented and colour-coded according to their classification.

3

OBJECT DETECTION FROM IMAGE REPRESENTATION IN THE COMPRESSED DOMAIN

This chapter proposes the Processing of Microscopy Images in the Compressed Domain (ProMIC) architecture, an object detection system that operates on the compressed domain representation of images generated by a learning-based encoder. Processing these representations differs significantly from processing those obtained through classical approaches, from which, extracting information about image content almost always requires decompressing the image and analysing it in the pixel domain. This process decreases resource consumption, including energy, memory, and time.

In the following sections, a detailed explanation of the proposed architecture is provided. This chapter also discusses the design process of the Domain Translator block, an essential part of the proposed framework that enables the translation of information contained in the compressed domain representation to be used by a modified object detector. This chapter is structured as follows: Section 3.1 presents the proposed architecture in detail, including the learning-based image codec used to obtain the compressed domain representations, as well as the object detector adopted and its modifications to process the latent code. Section 3.2 discusses the design process of the Domain Translator block, including the rationale behind each layer's selection. Finally, Section 3.3 presents the adopted training strategy for model fitting.

3.1 PROPOSED ARCHITECTURE

The architecture of the proposed object detection model that operates on compressed domain representations is presented in Figure 17. This model is built upon a generic learning-based image codec, as described in Section 2.1. The choice of a learning-based codec is motivated by its representation format, which projects the image into a feature space that contains all essential information about the image and from which it is possible to directly extract all the information needed by the object detector to localise the objects of interest.

This work proposes feeding the image representation from the learning-based image codec's feature space directly into a learning-based object detector. This

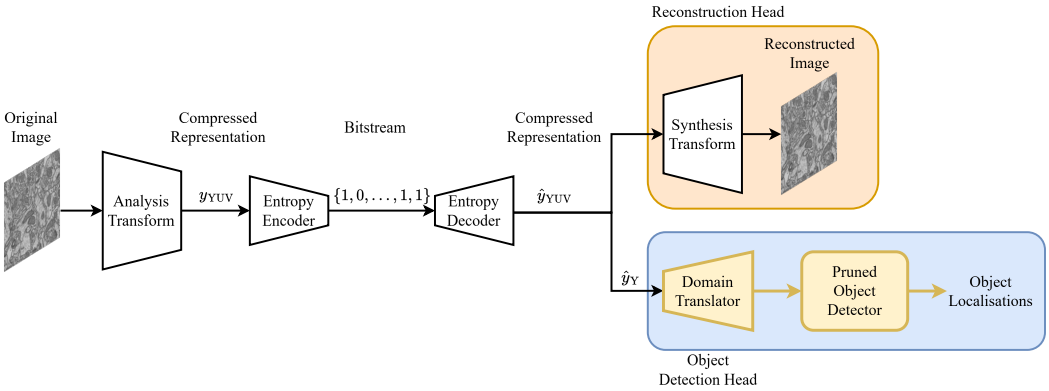


Figure 17: Proposed compressed-domain object detection architecture.

approach aims to perform detection directly on the latent representation of the image, bypassing the decompression process and thereby conserving resources such as memory, energy, and time.

However, the object detection model cannot directly process the latent representation generated by the learning-based image codec. This limitation arises from the fact that the feature space created by the codec differs fundamentally from that used by the object detection model, both in dimensions and in the semantic meaning of its component values. To address this challenge, inspired by the work of [78], we propose using a Domain Translator block that projects the codec-generated latent representation into the feature space used by the learning-based object detector.

3.1.1 *Learning-based Image Codec*

The proposed object detection pipeline requires selecting a learning-based image codec to obtain the latent representations of images. This work uses the ISO/IEC 6048-1 (ITU-T T.840) standard for a learning-based image coding system known as JPEG-AI [126]. This system was chosen due to these organizations' and related researchers' efforts to develop a single representation suitable for both human visualization (image reconstruction) and machine consumption (image processing, object detection, classification, etc.) [5], as well as its state-of-the-art performance in terms of perception metrics, such as MS-SSIM.

Figure 18 shows the general architecture of the JPEG-AI learning-based image codec. The quantized latent image representation ($\hat{y}$) from the Latent Prediction block is not directly included in the JPEG-AI bitstream. Instead, the bitstream contains two elements encoded by the Arithmetic Encoder (AE): the quantized entropy information ($\hat{z}$) and the quantized prediction residue ($\hat{r}$). To recover $\hat{y}$, the decoder first uses $\hat{z}$ to generate a prediction of $\hat{y}$, then applies the residue $\hat{r}$ to complete the reconstruction without loss.

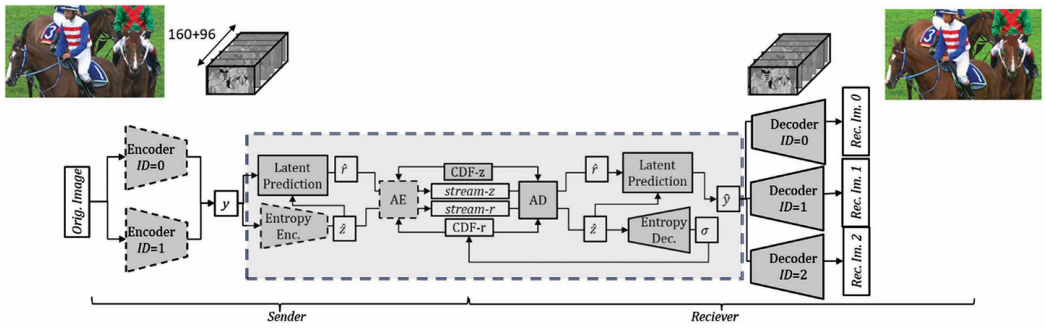


Figure 18: General JPEG-AI architecture. y: latent image representation; $\hat{z}$: quantized entropy information of y; $\hat{r}$: quantized residue of y prediction; σ: $\hat{r}$ entropy information; $\hat{y}$: quantized latent image representation; AE: Arithmetic Encoder; AD: Arithmetic Decoder; stream-z: bitstream of encoded $\hat{z}$, which is transmitted by the JPEG-AI codec; stream-r: bitstream of encoded $\hat{r}$, which is transmitted by the JPEG-AI codec; [33].

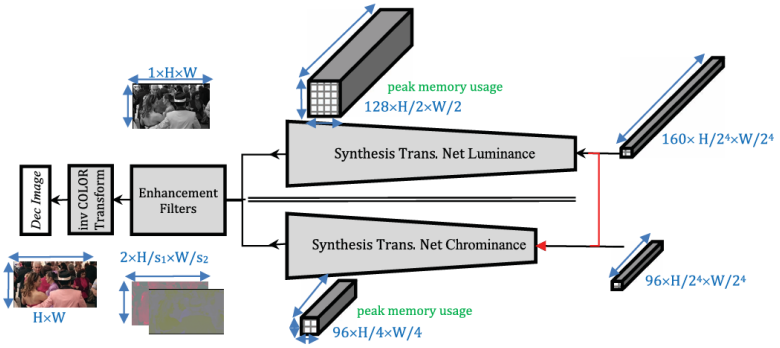


Figure 19: JPEG-AI conditional colours separation in the decoder. H and W represent height and width of the original image [33].

This process is important for understanding why compressed domain object detection cannot operate directly on the JPEG-AI bitstream. As Figure 17 illustrates, the object detection head requires the latent representation $\hat{y}$, not the encoded entropy information or residue found in the bitstream. Therefore, before object detection can occur, components of the JPEG-AI decoder (specifically the entropy decoder and latent predictor) must first process the bitstream to reconstruct the compressed domain representation $\hat{y}$.

Although not explicitly shown in Figure 18, $\hat{y}$ contains two latent representations: one for the luminance component ($\widehat{y_{\mathrm{Y}}}$) and another for the corresponding chrominances ($\widehat{y_{\mathrm{UV}}}$). Both representations are necessary for proper image reconstruction. The JPEG-AI codec employs distinct networks for luminance and chrominance components. Figure 19 illustrates the synthesis transformation separation between these components. Notably, the luminance branch depends solely on the luminance component's latent representation $\widehat{y_{\mathrm{Y}}}$, while the chrominance branch requires both luminance and chrominance latent representations.

Since microscopy images are mono-channel, all image information is contained within the luminance component. This allows discarding the chrominance compo-

nents and processing only the luminance component's latent representation, $\widehat{y_Y}$, thereby simplifying the Domain Translator block and reducing memory consumption.

3.1.2 *Learning-based Object Detector*

Various learning-based object detection models exist, each designed for specific tasks and operating conditions. However, most state-of-the-art object detection models share a common architectural strategy: analyzing input images using a feature extraction network (commonly called a backbone) and processing the extracted features through specific modules to determine object locations. This architecture type integrates seamlessly with the proposed architecture presented in Figure 17, requiring only minimal modifications to the backbone to extract information from the latent representation rather than the original image.

This work employs the *Detectron2* framework [127] as its object detector model, an upgraded version of the *Detectron* [128] model released by Facebook AI Research in October 2019. *Detectron2* supports both object detection and instance segmentation tasks and comprises four main components [129]:

1. the backbone, which extracts features from the input image;
2. the neck, which combines features at different scales from the backbone to create a Multiscale Feature Pyramid, useful for detecting objects at various scales;
3. the Region Proposal Network (RPN), which generates 1000 bounding box proposals indicating potential object locations with different confidence scores;
4. the Box Head, which processes, improves, and filters the RPN-generated proposals, outputting up to 100 high-confidence bounding boxes.

Figure 20 presents the general architecture of *Detectron2*, with the input path for images represented by a dashed line.

The choice of *Detectron2* as the object detection model was influenced by the study in [78], which examined modifications to RetinaNet's ResNet-50 backbone for compressed domain processing. *Detectron2* also uses ResNet-50 as its backbone network and shares a similar architecture with RetinaNet, facilitating result comparison.

To enable processing compressed domain image representations, certain backbone blocks designed for pixel domain processing must be removed and replaced with a Domain Translator. While this modification can be implemented at different points in the backbone, it must ensure that all outputs required by the Neck to create the Multiscale Feature Pyramid are preserved. Following the approach described in [78],

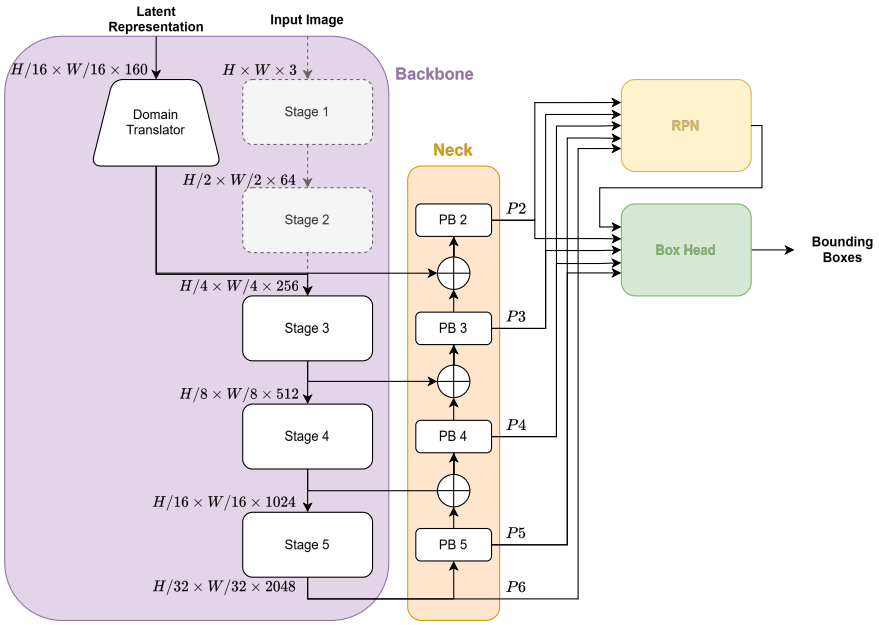


Figure 20: General *Detectron2* Architecture with modifications for latent representation processing. Solid line: Inference path; Dashed line: Image input path.

which showed that the optimal cutting point in ResNet-50 is at the end of Stage 2, the architecture herein proposed involves removing stages 1 and 2 of *Detectron2*'s backbone, replacing them with a Domain Translator. Figure 20 illustrates the modified *Detectron2* Architecture adapted for latent representation processing.

3.2 DOMAIN TRANSLATOR

In the proposed architecture, the latent representation of the input image must be projected onto the feature space of the pruned detector to extract all features needed by the specific modules, as explained in Section 3.1.2. To accomplish this, the use of a Domain Translator block is proposed, which translates the latent representation from the learning-based codec's compressed domain to the pruned detector's feature domain.

The design of an ideal Domain Translator must satisfy a single primary constraint: providing the pruned detector with the same input it would receive when processing an original image. This constraint can be divided into two sub-constraints:

1. The pruned detector's input must maintain identical dimensions whether processing the latent representation or the original image.
2. The information present in the pruned detector's input must be equivalent whether extracted from the image or the latent representation.

Addressing the first sub-constraint requires designing the Domain Translator specifically for each learning-based image codec and object detector pair. This

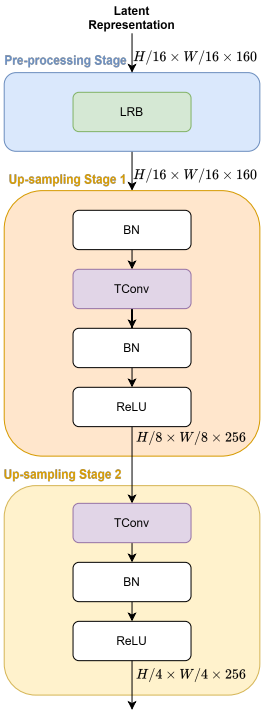


Figure 21: Architecture of the proposed Domain Translator. LRB: Lightweight Residual Block; BN: Batch Normalisation layer; TConv: Transposed Convolutional layer; ReLU: Rectified Linear Unit layer; H: Original Image height; W: Original Image width; Purple colour block: Upsampling block.

specificity is necessary because the Domain Translator must handle the unique requirements of each codec and detector to make their feature space dimensions compatible. For the specific pair used in this work—JPEG-AI and *Detectron2*—the dimensions of each feature space are indicated in Figure 20 at the Domain Translator input and Stage 3. The compressed domain representations have dimensions of $H/16 \times W/16 \times 160$, where H and W represent the original image height and width, respectively, while the *Detectron2*'s feature space dimensions are $H/4 \times W/4 \times 256$.

The JPEG-AI analysis transform spatially downscales images by a factor of 16, while Stages 1 and 2 downscale by a factor of 4. Therefore, to equalize the spatial dimensions between the Domain Translator and Stage 2 outputs, the Domain Translator must upscale the latent representations' spatial dimensions by a factor of 4. In the channel (third) dimension, while Stages 1 and 2 increase the number of channels from 3 to 256, the Domain Translator must transform the input from 160 channels to achieve the same 256-channel output.

In Figure 21, the architecture of the proposed Domain Translator that fulfills the requirements of both sub-constraints is presented. Inspired by the design proposed in [78], the Domain Translator comprises three stages, a pre-processing stage that employs a Lightweight Residual Block (LRB) to enhance the feature extraction and two up-sampling stages that use transposed convolutional layers for up-sampling and Rectified Linear Unit (ReLU) as the activation function. To mitigate overfitting

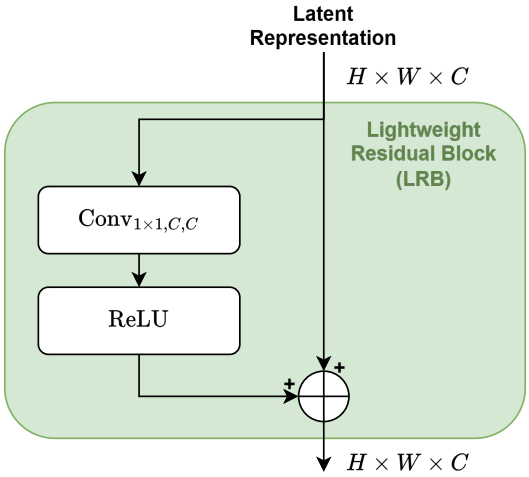


Figure 22: Lightweight Residual Block (LRB) architecture.

effects, Batch Normalization (BN) layers are included at the beginning of the first up-sampling stage and after each transposed convolutional layer. The first up-sampling stage doubles the spatial resolution and increases the number of channels from 160 to 256, while the second stage further doubles the spatial resolution while maintaining the channel count.

The Domain Translator architecture incorporates a pre-processing block to enhance feature extraction from the latent representation. Figure 22 presents the architecture of this pre-processing block, named Lightweight Residual Block (LRB). The LRB consists of a convolutional layer with a ReLU activation function, whose output combines with the block's input through a residual connection. The convolutional layer preserves the number of input channels, maintaining the same channel dimensionality throughout the block. While inspired by the LRB block used in the JPEG-AI decoder's synthesis transform network for the luminance component, this implementation uses a 1×1 kernel in the convolutional layer instead of the original 3×3 kernel to reduce computational complexity. This modification achieves both improved efficiency and enhanced performance in the Domain Translator.

3.3 MIRROR TRAINING STRATEGY

Theoretically, the maximum performance of an object detection model processing latent representations should equal that of a pixel domain processing model using reconstructed images, since all information contained in the compressed representation should be present in the reconstructed image. Therefore, the pixel domain processing detection model can serve as a reference model to guide the training process of the latent representation processing model. This section explains tech-

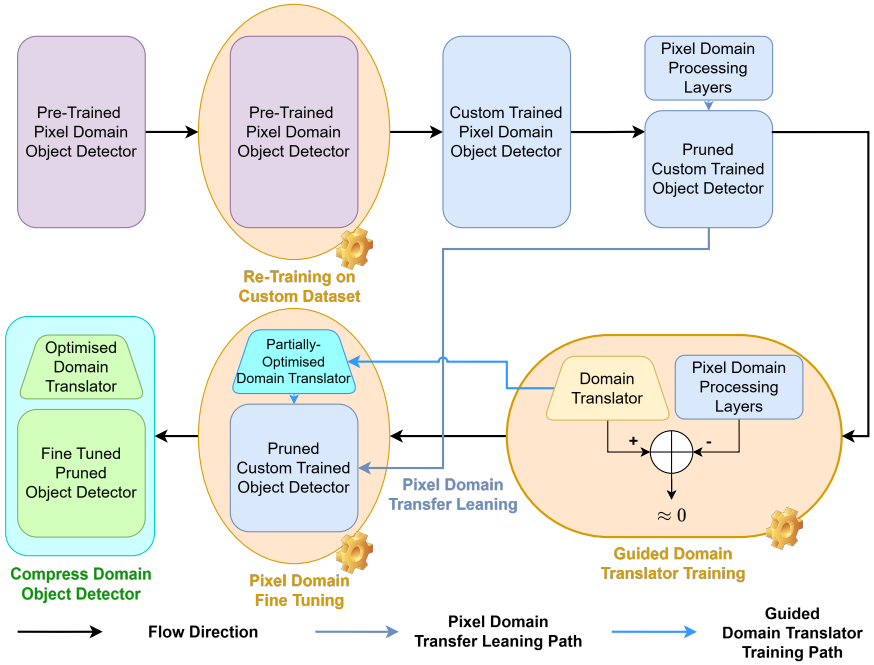


Figure 23: Mirror Training Strategy framework. The strategy comprises three sequential training stages (denoted by orange gear icons): (1) Retraining of a Pixel Domain Object Detector on the target dataset; (2) Training of a Domain Translator to extract equivalent information from latent representations as obtained from images, utilizing the image processing layers from stage 1; (3) Integration and fine-tuning of the Domain Translator with the Pixel Domain Object Detector (excluding image processing layers) to create the final Compressed Domain Object Detector.

niques for leveraging the knowledge learned by the reference model to enhance the performance of the latent representation processing object detection system.

The core idea behind the Mirror Training strategy is to create a model that performs as closely as possible to its pixel domain processing counterpart. To achieve this objective, a procedure is proposed to develop a "Mirror" model that matches the performance of the pixel domain processing model while making the fewest possible modifications to adapt it for latent representation processing.

The general intuition behind the Mirror Training strategy is illustrated in Figure 23. This process comprises three distinct training stages. First, a pre-trained pixel domain object detector (Pixel Domain Object Detector (PD-OD)) is retrained on the dataset of interest. Second, the retrained model's image processing layers (Pixel Domain processing Layers (PD-L)) are used to train the Domain Translator to extract useful features from the latent representation. This training process is referred as Guided Domain Translator Training (GDTT), Finally, the PD-L in the retrained PD-OD are replaced with the Domain Translator, and the entire model is fine-tuned to properly integrate the Domain Translator with the Pruned PD-OD. Ideally, the remaining model components continue to function as if processing a pixel domain image. This training process culminates in a compressed domain

object detector (Compressed Domain Object Detector (CD-OD)) that, in optimal conditions, achieves performance comparable to its pixel domain counterpart.

3.3.1 *Pixel Domain Transfer Learning*

Transfer Learning (TL) is a common technique in ML used to accelerate training and reduce the amount of data needed to train models. This technique involves initially training a model on a large, rich dataset to learn general and useful features, then retraining it on the dataset of interest. The final model's performance is strongly influenced by the similarity between the final and original tasks, as this determines the utility of features learned from the first dataset.

Domain Adaptation (DA) is a subfield of Transfer Learning that specifically addresses the challenge of adapting a model trained on a source domain to perform well on a target domain, where the two domains may have different data distributions. The goal of DA is to minimize the distribution discrepancy between the source and target domains, enabling the model to generalize effectively. This is particularly useful in scenarios where labelled data in the target domain is scarce or unavailable, but abundant labelled data exists in the source domain. Techniques in DA often involve aligning the feature spaces of the two domains, either through feature transformation, adversarial training, or other methods that encourage domain-invariant representations [130], [131].

In microscopy image processing, datasets typically contain hundreds or thousands of images—relatively small compared to the millions of natural images available in datasets commonly used to train detection models. For the *Detectron2* detector, pre-trained weights are available from models trained on the *train2017* COCO dataset [108], which contains more than 200,000 images. While these weights served as the starting point for models that process the pixel domain images, it was observed that initialising the latent representation processing detector using weights from the model trained on the pixel domain produced better results than using the pre-trained COCO dataset weights alone. This approach of configuring the compressed domain object detector model with the weights from the pixel domain trained model is termed Pixel Domain Transfer Learning (PD-TL).

3.3.2 *Guided Domain Translator Training (GDTT)*

In order to reduce the detection performance gap between pixel domain and latent representation processing object detection architectures, the GDTT strategy is proposed. The core idea of GDTT is to apply a knowledge distillation approach,

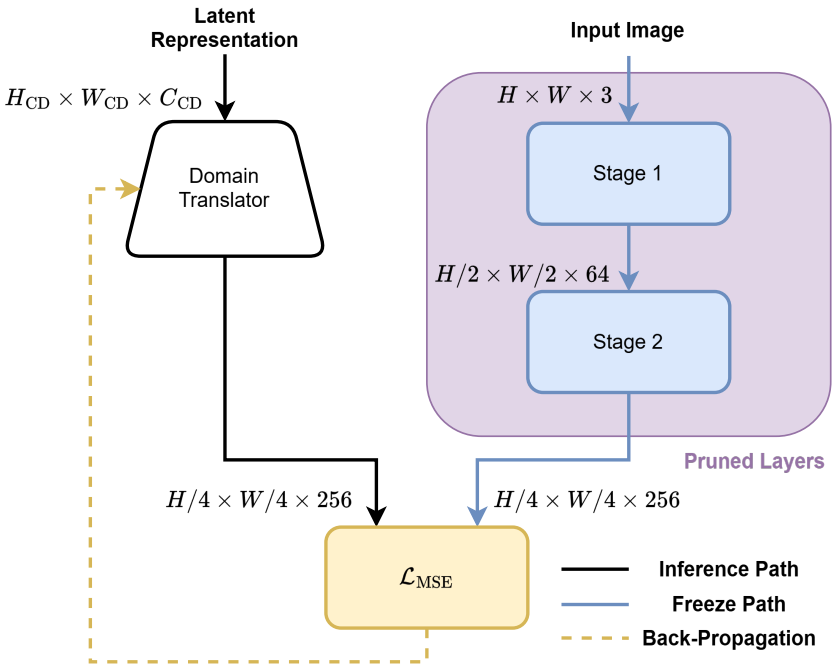


Figure 24: Guided Domain Translator Training architecture. The Domain Translator is trained to generate outputs from latent representations that match those produced by the image processing layers of the retrained Pixel Domain Object Detector, where these layers have been removed and serve as a reference for training.

transferring the knowledge acquired by the pixel domain processing object detection model (the teacher) to its latent representation counterpart (the student). In the proposed architecture, the knowledge distillation process involves approximating the activations of the pruned layers from the pixel domain model and the Domain Translator block.

The GDTT process, involves training the Domain Translator block using the outputs of the pruned layers from the pixel domain object detector model. This optimization is achieved by minimizing the Mean Squared Error (MSE) loss function, $\mathcal{L}_{\text{MSE}}$, between the outputs of the pruned layers and the Domain Translator. The MSE for this application is mathematically defined as:

$$\mathcal{L}_{\text{MSE}}(x, y) = \frac{\sum_i^{H/4} \sum_j^{W/4} \sum_k^{256} \left(x_{i,j,k} - y_{i,j,k}\right)^2}{\frac{H}{4} \times \frac{W}{4} \times 256}, \tag{40}$$

where $x_{i,j,k}$, and $y_{i,j,k}$ correspond to the values of the output features maps of the Domain Translator and the pruned layers, respectively, at the position $\{i, j\}$ in the k^{th} feature map and H and W represent the height and width of the original image. The goal of this process is to train the Domain Translator to effectively replace the pruned layers, providing the model with the same information from the latent representation as it would receive from processing the pixel domain image. The architecture of this training process is illustrated in Figure 24.

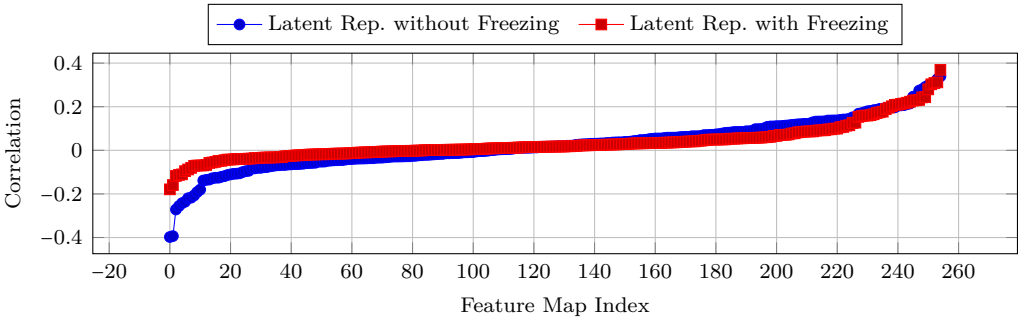
3.3.3 *Pixel Domain Fine Tuning*

Bulding upon the idea behind the PD-TL process, which states that retaining features learned from training the model on pixel domain images enables the model to extract more meaningful information from the compressed domain representations, we decided to go even further and introduced the Pixel Domain Fine Tuning (PD-FT) process. This process aims to fit a Domain Translator and a pruned pixel domain object detector to work together and reduce the performance gap in object detection between models trained on pixel domain images and those trained on latent representations.

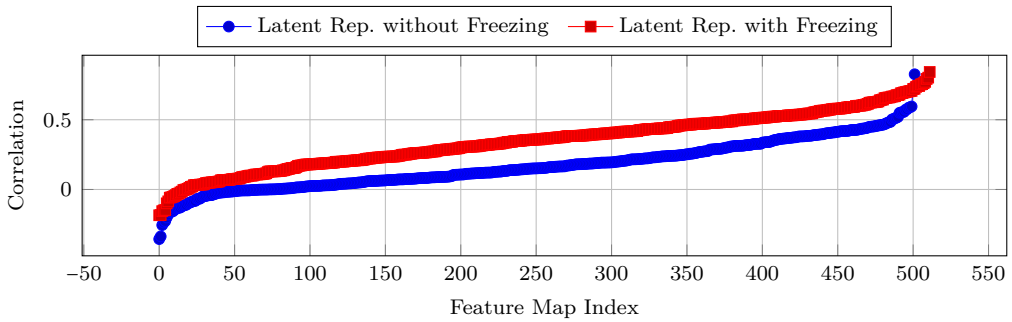
The PD-FT process applies the PD-TL technique and then freezes the backbone stages not replaced by the Domain Translator (Stages 3, 4, and 5) to preserve the features learned during pixel domain training. With the pruned backbone frozen, the remaining components (Domain Translator, neck, and context modules) are trained to improve object detection performance. When the training process approaches convergence, the pruned backbone stages are unfrozen, and the entire model undergoes training to properly integrate the Domain Translator with all other modules.

To support the use of the PD-FT process, evaluations were conducted at two key points: (1) at the final stage, by measuring object detection performance using AP, and (2) at intermediate stages, by analysing the correlation between backbone stage inputs in the reference pixel-domain processing model and the latent representation processing models, with and without stage freezing. In Figures 25a and 25b, the correlation of feature input maps for Stages 3 and 4 is presented, comparing pixel domain and latent representation processing. For clarity, the feature maps in each experiment are sorted from minimum to maximum correlation value, highlighting the overall correlation differences between experiments.

In Figure 25a, the correlation of Stage 3 backbone feature map inputs is shown, comparing pixel domain and latent representation processing. The blue line represents the correlation between reference features (pixel domain processing) and latent representation features when the full model is trained from reference weights. The red line corresponds to the correlation between reference features and latent representation features when only the Domain Translator is trained, initialized from reference model weights. Both correlations are similar and close to zero for a significant portion of the feature maps, indicating weak similarity between the outputs of the Domain Translator (when processing latent representations) and Stage 2 (when processing pixel domain images). However, this behaviour changes when analysing the inputs of Stage 4.



(a) Stage 3.



(b) Stage 4.

Figure 25: Correlation of backbone Stages 3 and 4 input feature maps: Pixel domain vs. latent representation processing. Blue line: Full model training from reference weights. Red line: Domain Translator training, initialized from reference model weights.

In Figure 25b, the correlation of Stage 4 backbone feature map inputs is presented. Unlike Stage 3, there is a clear difference in correlation between the fully retrained model (blue line) and the model with frozen stages (red line). This evaluation supports the use of the PD-FT process by demonstrating that freezing Stages 3, 4, and 5 enhances feature map consistency with the reference model. When these stages are frozen, the feature maps at the input of Stage 4 show a higher correlation with those of the reference pixel-domain processing model compared to when the entire model is trained. This trend continues in later stages, leading to improved detection performance and reducing the gap in AP between latent representation processing and the reference model. Notably, this approach results in a 6.39% increase in AP75 and a 15.8% increase in APl. This improvement is particularly relevant for detecting large objects (area $> 96^2$ pixels), as Stages 4 and 5 contribute significantly to the APl metric. It is possible to notice that the blue line ends earlier than the red one, this is due to 10 of the feature maps at the input of the stage 4 for the fully retrained model have a constant value in the whole map, thus have a variance value of zero and the correlation coefficient is undefined. Additionally, the blue line ends earlier than the red one because, in the fully retrained model, 10 feature maps at the input of Stage 4 contain constant values across the entire map, resulting in a variance of zero and an undefined correlation coefficient.

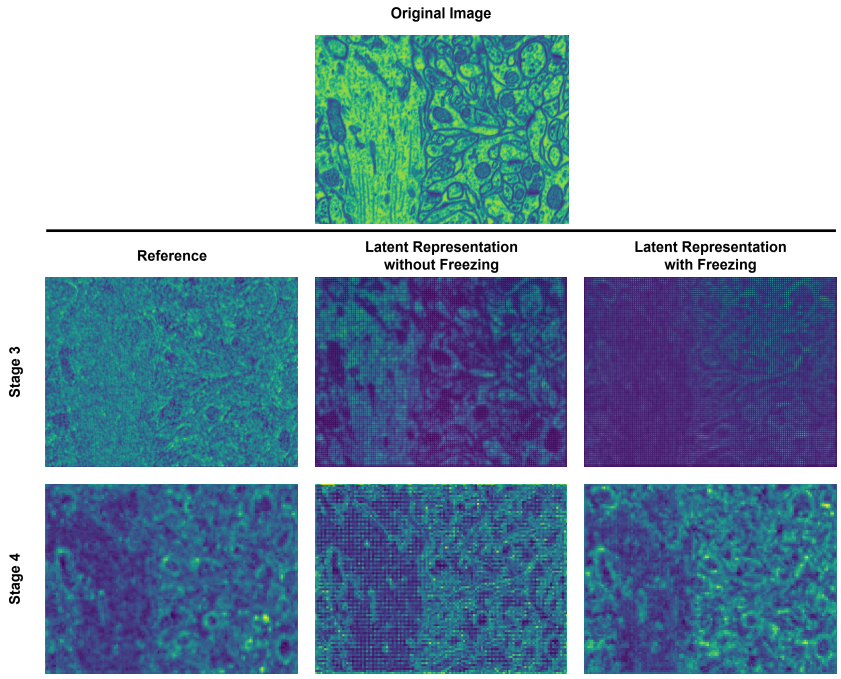


Figure 26: Backbone stages feature inputs for the original image and its latent representations. (Top) Original input image. (Rows) Feature inputs to (1st row) Stage 3 and (2nd row) Stage 4 of the backbone model. (Columns) Left: Reference features from original image processing (Stages 1 and 2); Middle: Latent representation features with full model training from reference weights; Right: Latent representation features with Domain Translator training only, initialized from reference model weights.

To further clarify the comparison of feature inputs for Stages 3 and 4, Figure 26 provides a visual representation of the backbone stage feature inputs for the original image and its latent representations. The multi-channel feature maps were averaged to produce a single-channel representation of each feature input. The top row shows the original input image, while the subsequent rows display the feature inputs for (1st row) Stage 3 and (2nd row) Stage 4 of the backbone model. The columns represent distinct experimental conditions: (Left) Features extracted from the original image processing pipeline (Stages 1 and 2); (Middle) Latent representation features obtained by training the entire model using reference weights; (Right) Latent representation features generated by training only the Domain Translator, while the rest of the model is initialized with reference weights. The Stage 3 feature inputs for latent representation experiments show weak correlation with the reference experiment features, indicating that the same information could not be properly extracted from latent representations as from pixel domain images. In contrast, the Stage 4 feature inputs exhibit stronger similarity to the reference features, particularly when Stages 3, 4, and 5 are frozen. This increased similarity with the reference features leads to improved detection performance.

4

EXPERIMENTAL ASSESSMENT

This chapter presents and discusses the results of the experimental assessment in terms of both the compression efficiency and the object detection performance of the proposed approach. For compression evaluation, the four main settings of the JPEG-AI codec are compared against the HEVC and VVC codecs using the datasets described in Section 2.6. For object detection evaluation, the impact of each component of the proposed method is analyzed through an ablation study.

4.1 EXPERIMENTAL CONDITIONS

In this work, the implementation of the object detection model *Detectron2* available in [128] was used. To meet the requirements of JPEG-AI and the object detector, the images were first converted from grayscale to RGB as a pre-processing step. For all datasets, the images were then compressed and processed as 3-channel images.

To train compressed-domain object detection models, it is essential to obtain the latent representations of the input images. However, integrating the compression process directly into the model's input loading pipeline would significantly increase computational cost and training time, making the process infeasible. To overcome this limitation, all images were first encoded using JPEG-AI VM 6.1 at five bitrate levels: 0.12 bpp, 0.25 bpp, 0.50 bpp, 0.75 bpp, and 1.00 bpp. For each rate, the proposed compressed domain object detector was trained using the same settings and hyper parameters, as those specifided in the configuration file *COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml* available in the GitHub repository of the *Detectron2* [127] with the modifications presented in Table 3. Parameters marked as "Optimized" were fine-tuned using the Optuna framework, as described in Section 4.1.1.

The experiments were conducted on an NVIDIA GeForce RTX 3090 GPU and a CPU Xeon Gold 6336Y CPU @ 2.40GHz. In each training stage, the model was trained for up to 50000 epochs using the Stochastic Gradient Descent (SGD) optimiser. Additionally, early stopping was implemented with a relative threshold of -1 and a patience of 1000 iterations. For the GDTT process, a reduce-on-plateau mechanism was applied to adjust the learning rate, using a factor of 0.5 with a

Table 3: List of *Detectron2* parameters modified or modified for mitochondria detection on the *Lucchi++* dataset.

Parameter	Value
SOLVER.BASE_LR	Optimised
SOLVER.IMS_PER_BATCH	Optimised
SOLVER.MAX_ITER	50000
SOLVER.STEPS	Optimised
SOLVER.GAMMA	Optimised
SOLVER.MOMENTUM	0.949
SOLVER.WEIGHT_DECAY	-05
SOLVER.AMP.ENABLED	TRUE
INPUT.CROP.ENABLED	TRUE
INPUT.CROP.TYPE	relative_range
INPUT.CROP.SIZE	[0,5]
INPUT.RANDOM_FLIP	horizontal
INPUT.MIN_SIZE_TRAIN	[416, 512, 640, 768]
INPUT.MAX_SIZE_TRAIN	999999
TEST.AUG.FLIP	FALSE
TEST.AUG.MIN_SIZES	[416, 618, 768]
TEST.MAX_SIZE	999999
TEST.EVAL_PERIOD	100
MODEL.ANCHOR_GENERATOR.SIZES	[[32, 64, 128, 256, 512]]
MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS	[[0.5, 1.0, 2.0]]
MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE	128
MODEL.ROI_HEADS.NUM_CLASSES	1
MODEL.ROI_HEADS.SCORE_THRESH_TEST	0.5
MODEL.ROI_HEADS.NMS_THRESH_TEST	0.5
MODEL.RPN.IN_FEATURES	[”p2”, ”p3”, ”p4”, ”p5”, ”p6”]
MODEL.Detectron2_config_file_path	COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml
MODEL.Detectron2_checkpoint_url	COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml
DATALOADER.NUM_WORKERS	2

relative threshold of -01 and a patience of 10 epochs. The minimum learning rate was set to 10^{-10}.

4.1.1 *Parameter Optimisation*

In order to find the best configurations for training the models, the parameters marked as “Optimized” in Table 3 were fine-tuned for each rate using the Optuna framework [132] and running around 100 trials for each rate value considered.

In Table 4, the parameters search limits are presented. For the parameters *SOLVER.BASE_LR* and *Final Learning Rate*, the values are sampled from the logarithm domain during the optimisation process, while the rest of the parameters are sampled from the linear domain. The optimised parameter values for all models in this study are provided in Appendix B. The description of the optimised parameters is presented as follows:

Table 4: Parameters optimised using the Optuna framework and its corresponding search limits.

Parameter	Limit	
	Minimum	Maximum
SOLVER.BASE_LR	1×10^{-4}	1
SOLVER.IMS_PER_BATCH	4	12
Delta Learning Rate	1×10^{-6}	1×10^{-1}
Initial Iteration	100	5000
Number of Steps	0	100
Iteration Step	100	1000

- ***SOLVER.BASE_LR*:** Base learning rate from which it is decreased;
- ***SOLVER.IMS_PER_BATCH*:** Actual number of inputs processed in parallel per batch;
- ***Delta Learning Rate*:** Amount of learning rated decreased at the end of the decay process;
- ***Initial Iteration*:** Iteration at which the learning rate decay process begin;
- ***Number of Steps*:** Number of times (decay steps) that the learning rate is decreased to decay *Delta Learning Rate*;
- ***Iteration Step*:** Number of iteration between each decay step.

4.2 COMPRESSION EFFICIENCY

To evaluate the compression performance of different configurations of the JPEG-AI image coding standard and determine the optimal configuration for applying the proposed method, all the datasets introduced in Section 2.6 were compressed using the two JPEG-AI operation points (with and without tools enabled), HEVC/H.265, and VVC/H.266 coding standards. The compression efficiency between datasets was compared using the PSNR-HVS-M and MS-SSIM metrics, both objective quality evaluation metrics specified in the "JPEG AI Common Training and Test Conditions" document [113].

Figure 27 presents the mean PSNR-HVS-M versus rate (bpp) for the *Lucchi++*, *Kasthuri++*, and *Delmic* datasets for the different codecs. The datasets are distinguished by color: red for *Kasthuri++*, green for *Lucchi++*, and blue for *Delmic*. The results show distinct compression efficiency patterns across datasets. The *Kasthuri++* dataset achieves the highest PSNR-HSV-M values, particularly at higher rates, followed by *Lucchi++* and *Delmic*. *Kasthuri++* also exhibits the

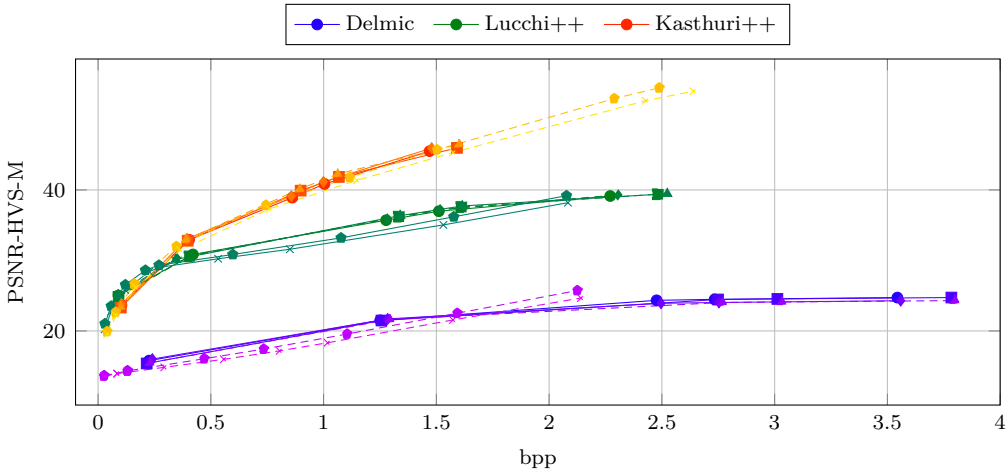


Figure 27: PSNR-HVS-M vs. bits per pixel (bpp) for the *Lucchi++*, *Kasthuri++*, and *Delmic* datasets using different codecs. Each dataset is represented by a distinct color shade: red for *Kasthuri++*, green for *Lucchi++*, and blue for *Delmic*. Different codecs are indicated by unique marker symbols: VVC: Pentagon; HEVC: Cross; JPEG-AI Base Operation Point Tools Off: Circle; JPEG-AI Base Operation Point Tools On: Square; JPEG-AI High Operation Point Tools Off: Diamond; JPEG-AI High Operation Point Tools On: Triangle.

largest quality variation, spanning approximately 35 dB PSNR-HVS-M from around 20 dB to nearly 55dB, over the range 0.04 – 2.49 bpp (with the VVC and HEVC codecs). In contrast, the *Delmic* dataset exhibits a more limited performance, both in terms of quality range and highest performance figures attained.

A similar benchmark was carried out, but involving the MS-SSIM metric, with results shown in Figure 28. The perceptual quality trends mirror those observed in the PSNR-HSV-M analysis. The *Kasthuri++* dataset maintains the highest quality across all rates, likely due to the presence of black regions that aid the compression process. The *Lucchi++* dataset follows closely in compression performance. In general, the performance of the three codecs converges as the bitrate increases. Notably, the most significant differences occur at lower bitrates, where a clear gap in performance is observed between the Delmic dataset and the *Kasthuri++* and *Lucchi++* datasets. The *Delmic* dataset shows the lowest quality, particularly at lower rates, achieving an MS-SSIM value of approximately 0.50 with HEVC and VVC codecs. This performance is significantly lower compared to *Lucchi++* and *Kasthuri++* datasets, which achieve MS-SSIM values of around 0.80 and 0.90, respectively, using the same codecs.

To refine the previous analysis for codec-specific compression efficiency, each dataset was analysed individually. Since the proposed method will be applied only to the *Lucchi++* dataset as a proof-of-concept, the remainder of this section addresses compression efficiency solely for the this dataset. The results for the *Kasthuri++* and *Delmic* datasets are presented in Appendix C. Figure 29 shows the PSNR-HVS-M metric versus bits per pixel for the *Lucchi++* dataset. At lower rates,

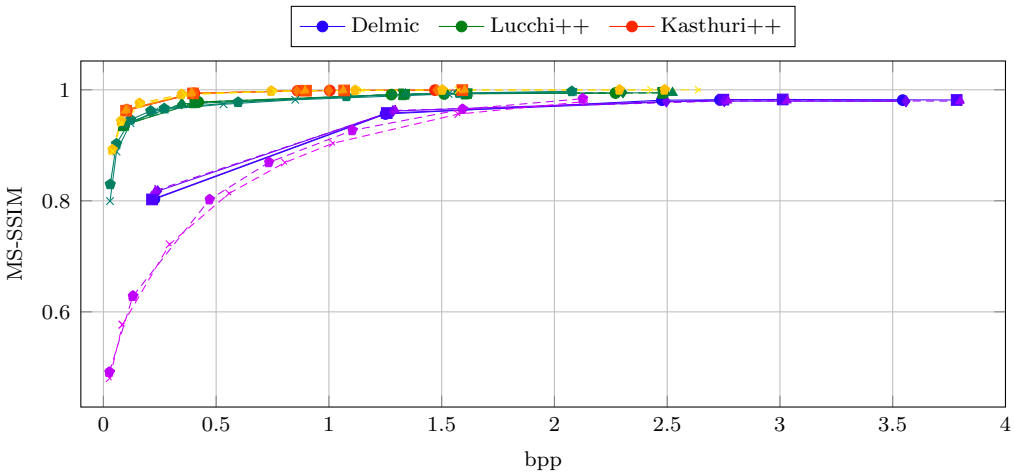


Figure 28: Multi-Scale Structural Similarity (MS-SIM) vs. bits per pixel (bpp) for the *Lucchi++*, *Kasthuri++*, and *Delmic* datasets using different codecs. Each dataset is represented by a distinct color shade: red for *Kasthuri++*, green for *Lucchi++*, and blue for *Delmic*. Different codecs are indicated by unique marker symbols: VVC: Pentagon; HEVC: Cross; JPEG-AI Base Operation Point Tools Off: Circle; JPEG-AI Base Operation Point Tools On: Square; JPEG-AI High Operation Point Tools Off: Diamond; JPEG-AI High Operation Point Tools On: Triangle.

JPEG-AI configurations perform similarly to HEVC and VVC codecs. However, the performance gap between classical and learning-based codecs is more notable at higher rates. Conversely, Figure 30 shows that for MS-SSIM metrics, JPEG-AI configurations more closely match classical codecs' performance at higher rates, with minimal differences at lower rates. This pattern is also evident in Figure 31, which presents the Information Content Weighted Structural Similarity Measure (IW-SSIM) versus bits per pixel.

In Figure 32, we present the Video Multimethod Assessment Fusion (VMAF) versus bits per pixel (bpp) for the *Lucchi++* dataset across different codecs. At intermediate rates, JPEG-AI codec demonstrates superior compression performance in terms of VMAF compared to HEVC and VVC, particularly in configurations with tools enabled (purple and red lines). The JPEG-AI high operation point with tools enabled achieves approximately 8% improvement over the VVC codec at equivalent rates. At higher rates, the performance gap between codecs narrows. In contrast, Figure 33 shows that when comparing codecs using the Visual Information Fidelity (VIF) metric, while the performance trends are similar, the difference between classical and learning-based codecs increases at higher rates. The classical codecs (HEVC and VVC) demonstrate markedly superior VIF performance at higher rates, achieving approximately 20% improvement at equivalent rates.

Figure 34 shows the Normalized Laplacian Pyramid (NLPD) versus bits per pixel (bpp) for the *Lucchi++* dataset. At lower rates, JPEG-AI configurations perform similarly to HEVC and VVC. However, at higher rates, HEVC and VVC demonstrate clear superiority as their NLPD values continue to decrease, while JPEG-AI appears

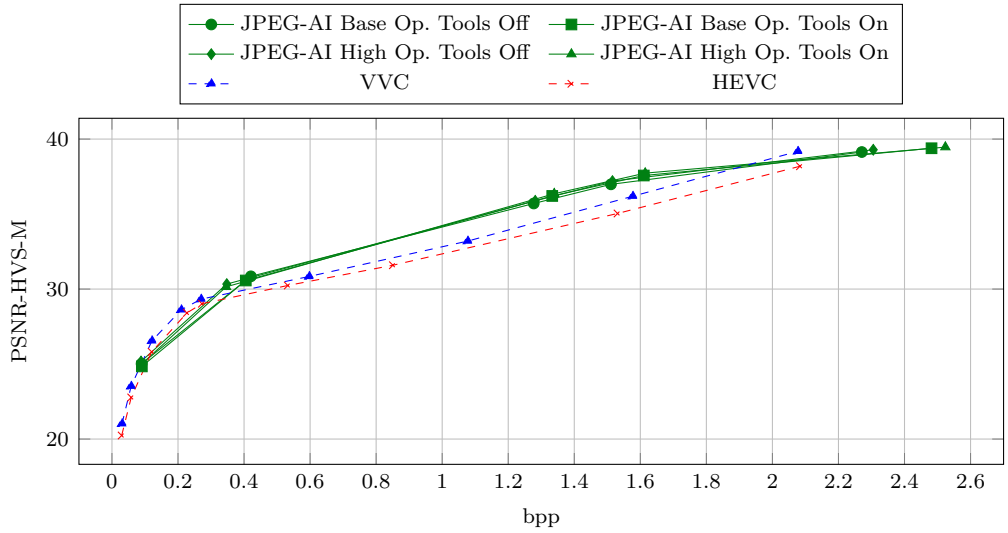


Figure 29: PSNR-HVS-M vs. bits per pixel (bpp) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

to plateau around 0.12. Conversely, when evaluating codec performance using FSIM (shown in Figure 35), the performance gap is more pronounced at lower rates, similar to MS-SSIM and IW-SSIM patterns. At higher rates, the performance across all codecs becomes more uniform.

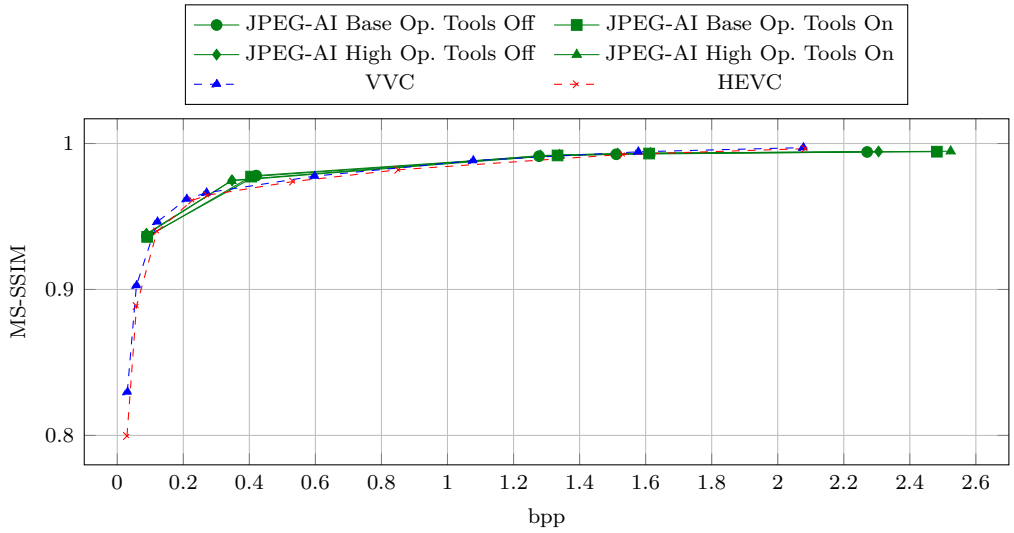


Figure 30: Multi-Scale Structural Similarity (MS-SIM) vs. bits per pixel (bpp) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

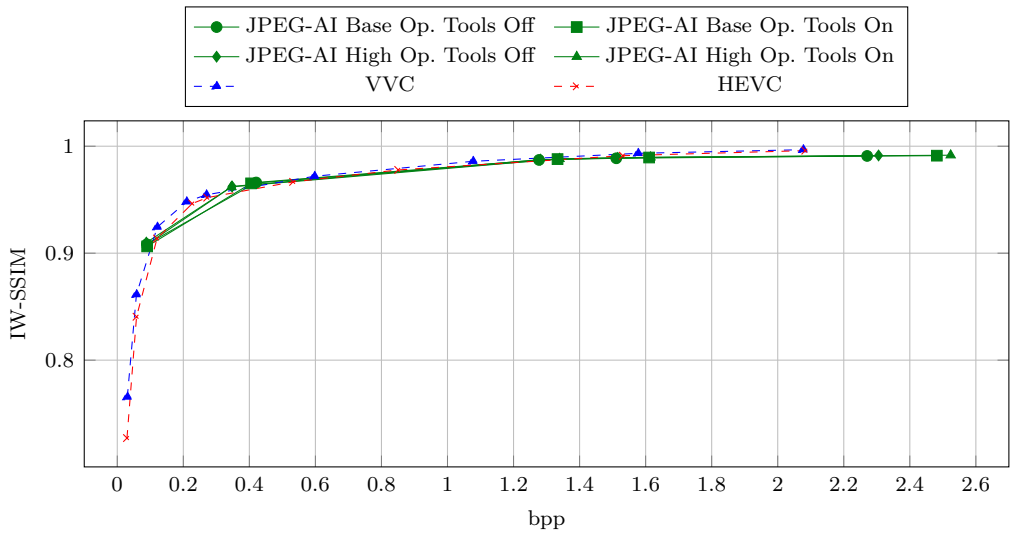


Figure 31: Information Content Weighted Structural Similarity Measure (IW-SSIM) vs. bits per pixel (bpp) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

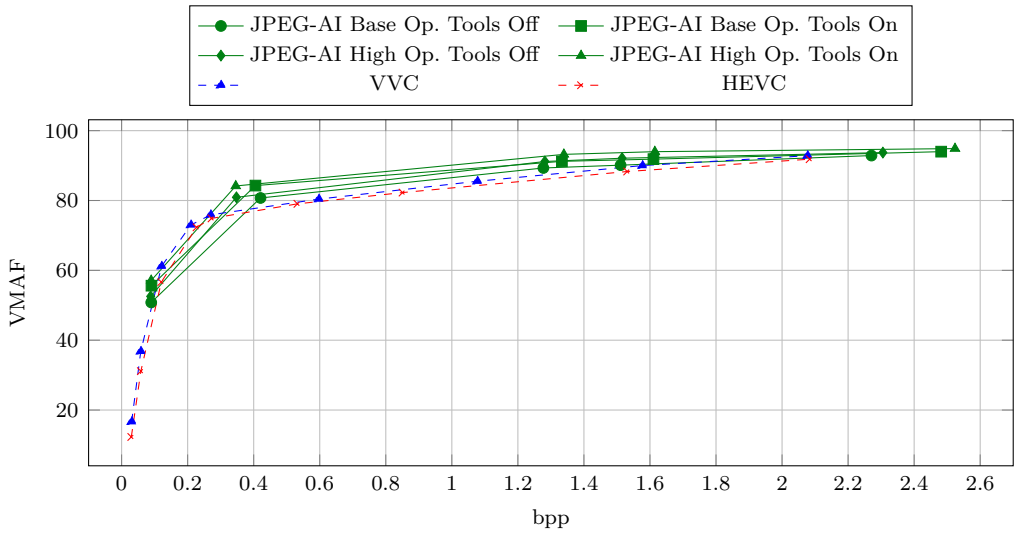


Figure 32: Video Multimethod Assessment Fusion (VMAF) vs. bits per pixel (bpp) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

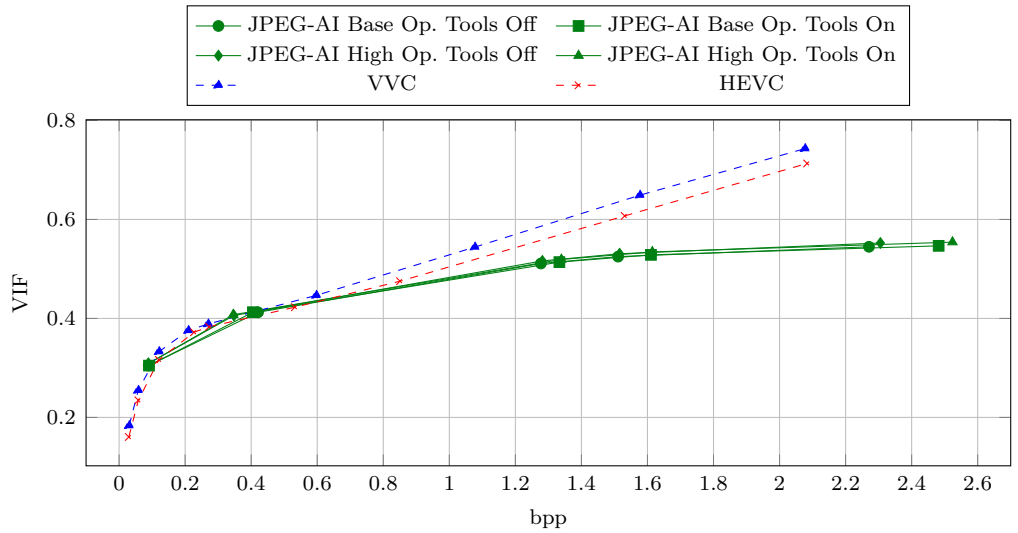


Figure 33: Visual Information Fidelity (VIF) vs. bits per pixel (bpp) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

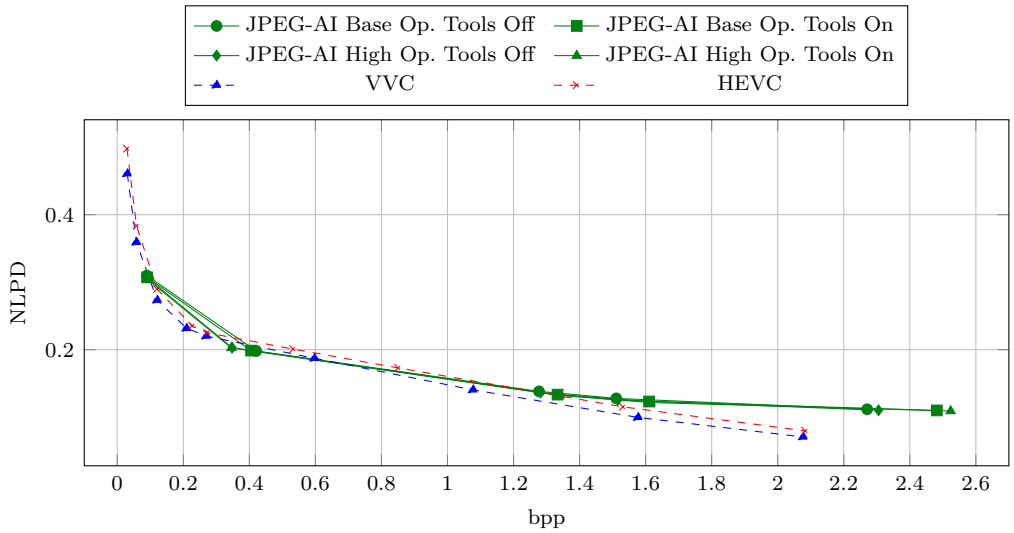


Figure 34: Normalized Laplacian Pyramid (NLPD) vs. bits per pixel (bpp) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

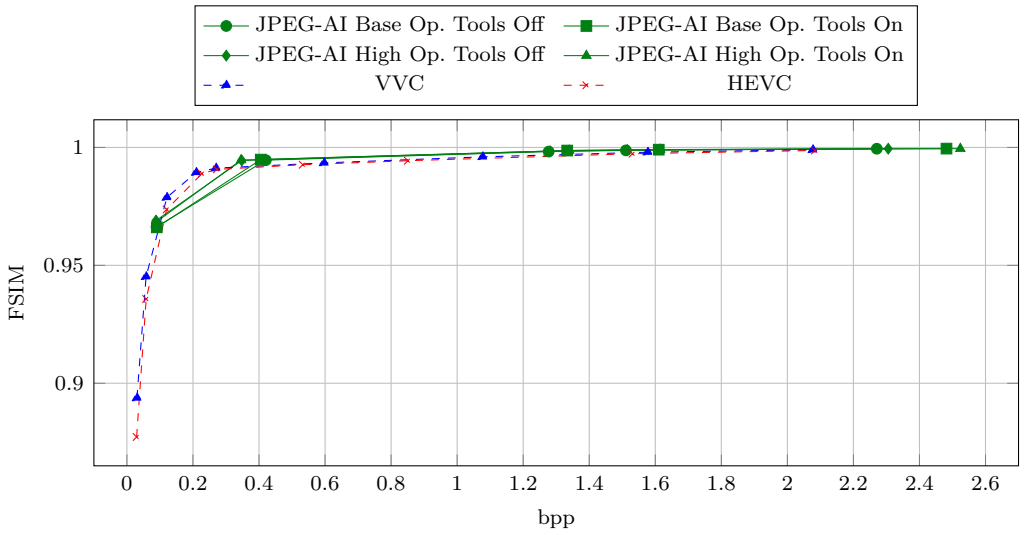


Figure 35: Feature Similarity (FSIM) for the *Lucchi++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

4.3 OBJECT DETECTION EFFICIENCY

Table 5: Comparison of the results, in terms of Bjontegaard Delta levels of the object detection metrics, obtained on test ("unseen") images of the *Lucchi++* dataset between the Base model, inspired on [78], and the proposed ProMIC method.

	Bjontegaard Delta					
Model	AP50 (*pp*)	AP75 (*pp*)	APs (*pp*)	APm (*pp*)	APl (*pp*)	AP (*pp*)
Reconstructed Images (Ref.)	0,00	0,00	0,00	0,00	0,00	0,00
Base [78]	-15,41	-26,94	-21,74	-19,95	-44,84	-23,44
ProMIC	**-0,50**	**-0,78**	**-1,27**	**0,08**	**0,57**	**0,21**

As mentioned before, the proposed method was applied to the *Lucchi++* dataset. To enable fair comparison, we followed the methodology described in [78]. Throughout this document, this baseline model is referred as the "Base" (B) model, and the proposed method as ProMIC.

As shown in Table 5, the performance of the Base model and ProMIC is compared using BD levels for the object detection metrics outlined in Section 2.5. All values in the table use the performance of the original object detector on decoded images as a reference. The BD levels for each model represent the average metric gain across all evaluation rates when using that model instead of the reference. Consequently, a negative BD value indicates a performance loss for that particular metric.

Table 5 shows that the proposed method improves object detection performance in the JPEG-AI compressed domain across all metrics when compared to the Base model. The method achieves a BD-AP50 of $-0.50pp$, representing an average performance loss of $0.50pp$ in AP50, but still this represents a $14.91pp$ BD improvement over the Base model. Similar improvements are observed across other object detection metrics: AP75 improves by $26.16pp$, APs by $20.47pp$, APm by $20.03pp$, APl by $45.41pp$, and AP by $23.65pp$.

Notably, when using the ProMIC model instead of the Base model, the performance gap between AP50 and AP75 significantly decreases from $11.53pp$ to just $0.28pp$. This dramatic reduction indicates that the detection model achieves higher confidence, maintaining consistent performance even under stricter detection criteria (i.e., increasing the IoU threshold from 0.5 to 0.75). Notably, the model excels in APm, APl, and overall AP metrics, outperforming the reference model. This suggests that valuable object detection features exist in the compressed domain, which may not be immediately apparent in reconstructed images.

As detailed in Section 2.5, APm represent the AP metric calculated exclusively for medium-sized bounding boxes, which constitute 80.52% of the *Lucchi++* dataset's test images. This predominance of medium-sized boxes underscores the significance of APm metric in assessing overall model performance for this dataset. The AP

Table 6: Ablation Study: Comparing the impact of individual components in the proposed method on Bjontegaard Delta levels of the object detection metrics on test ("unseen") images of the *Lucchi++* dataset.

	Architecture		Training Process		Bjontegaard Delta					
Model	Compressed Domain	LRB	PD-FT	GDTT	AP50 (*pp*)	AP75 (*pp*)	APs (*pp*)	APm (*pp*)	APl (*pp*)	AP (*pp*)
A (Ref.)					-	-	-	-	-	-
B [78]	✓				-15.41	-26.94	-21.74	-19.95	-44.84	-23.44
C	✓		✓		-9.15	-14.75	-20.25	-13.62	-18.95	-14.83
D	✓			✓	-1.96	-1.89	-3.83	-1.84	-1.10	-1.85
E	✓		✓	✓	-0.72	-0.83	**-1.00**	-0.97	-0.24	-0.98
F	✓	✓			-10.47	-19.22	-22.85	-15.59	-33.37	-18.34
G	✓	✓	✓		-6.17	-9.68	-19.45	-9.74	-14.08	-10.78
H	✓	✓		✓	-1.18	-1.33	-2.19	-1.53	-0.70	-1.39
ProMIC	✓	✓	✓	✓	**-0.50**	**-0.78**	-1.27	**0.08**	**0.57**	**0.21**

metric averages precision values across 10 equally spaced IoU thresholds from 0.5 to 0.95. Achieving a positive BD-AP value indicates that the proposed method outperforms the reference model at certain IoU thresholds above 0.50, further demonstrating its superior confidence compared to the Base model.

4.3.1 *Ablation Study*

The ProMIC method consists of multiple components and processes that refine the Base model into the final model shown in Table 5. The impact of each component and process on the model's performance is assessed through an ablation study, whose results are summarised in Table 6. This study compares model performance based on BD levels for the metrics presented in Section 2.5. The models are identified as follows:

- **A:** Reference model used for calculating BD metrics;
- **B:** Base model built following the method proposed in [78];
- **C - H:** Intermediate models in the ablation study;
- **ProMIC:** Complete proposed model.

The following sections analyse in detail the individual contribution of each component and process to the model's performance. Additionally, they compare these contributions with the two primary pixel-domain reference models: the **PD-OB!** (**PD-OB!**), which operates on both uncompressed images and reconstructed images at all target rates. All models in this study were optimized to maximize the AP50 metric; therefore, the ablation study results are primarily discussed in terms of AP50, with AP75 serving as an auxiliary metric. The results for all the metrics are presented in Appendix D.

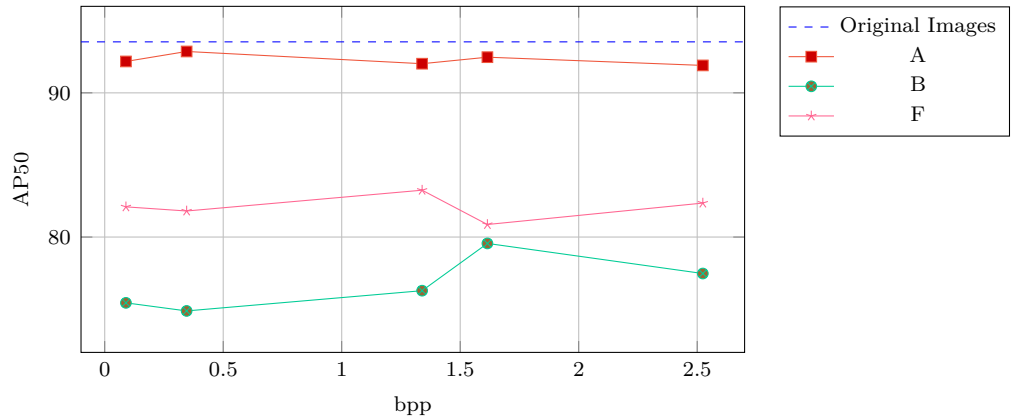


Figure 36: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

4.3.1.1 *Lightweight Residual Block*

To evaluate the impact of the Lightweight Residual Block (LRB) on model performance, two configurations are compared: Model B (the Base model) and Model F (the Base model incorporating the LRB block), using the performance of the PD-OD over the original (Original Images) and reconstructed images (Model A) as reference. Figures 36 and 37 present performance comparisons across different image compression settings, measuring Average Precision at 0.50 IoU threshold (AP50) and 0.75 IoU threshold (AP75), respectively. The graphs plot these metrics against bits per pixel (bpp) for the four models aforementioned.

The results demonstrate that incorporating the LRB block improves object detection performance across all compression rates, achieving AP50 improvements of 6.93*pp* at 0.35 bpp and AP75 improvements of 17.92*pp* at 0.09 bpp. Both metrics show a consistent pattern where improvements are more pronounced at lower bitrates and gradually decrease at medium bitrates.

Model F outperforms Model B across all rates and exhibits remarkable stability across different compression rates. This stability is particularly evident in AP50, where the maximum performance variation is only 2.38*pp* across all scales. At a higher confidence level, AP75, Model F maintains consistent performance, whereas Model B (without the LRB block) shows a clear trend of improved detection performance at higher bitrates.

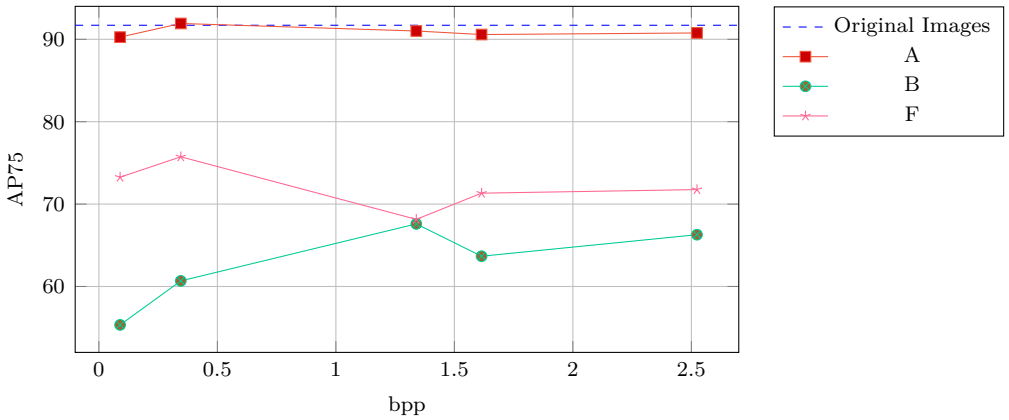


Figure 37: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

4.3.1.2 *Pixel Domain Fine Tuning*

To assess the effectiveness of the PD-FT process on model performance, two comparisons were conducted: first, evaluating its impact on the base Domain Translator architecture, and second, examining its effect on the final Domain Translator architecture incorporating the LRB block. This approach enables evaluation of the PD-FT process under comparable conditions.

Figures 38 and 39 present AP50 and AP75 values plotted against bits per pixel for models using the base Domain Translator architecture. The comparison includes the Base model (Model B), trained through a full training process, and Model C, which incorporates the PD-FT process. For reference, the PD-OD performance on original images (Original Images) and reconstructed images (Model A) is also included. The results demonstrate that fine-tuning features learned by Model A through the PD-FT process improves detection performance in both AP50 and AP75 metrics. This training process enhances CD-OD performance across all rates for AP50, achieving a 7.81*pp* improvement at 1.34 bpp. For AP75, the improvement reaches 19.87*pp* at 0.09 bpp.

Notably, similar to the LRB impact analysis, the PD-FT process yields greater performance improvements at lower rates compared to higher ones. This pattern is evident in both metrics, particularly at the highest rate where the performance gap between full training and PD-FT approaches is minimal. For the AP75 metric, the model trained through the PD-FT process actually performs slightly worse than the conventional full training process at the highest rate of 2.52 bpp.

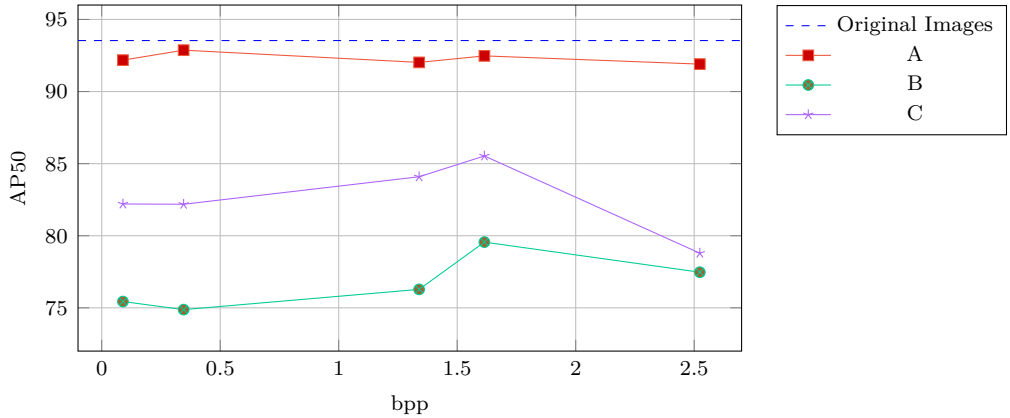


Figure 38: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

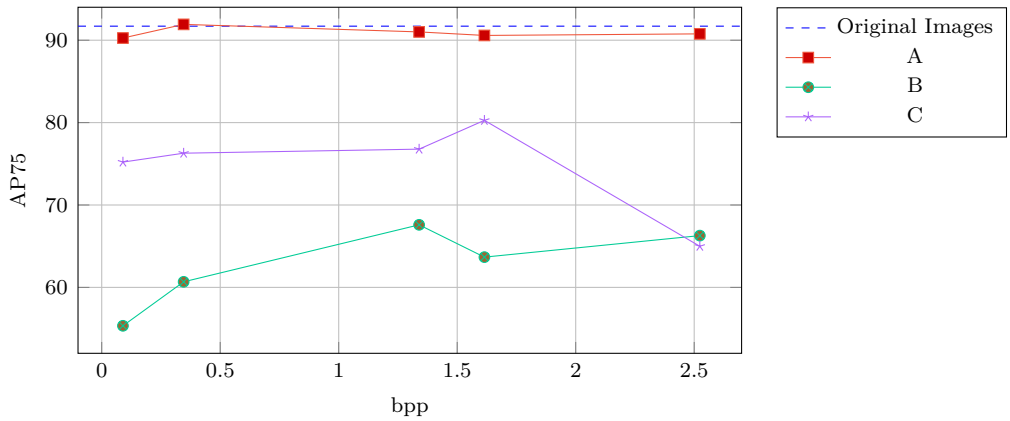


Figure 39: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

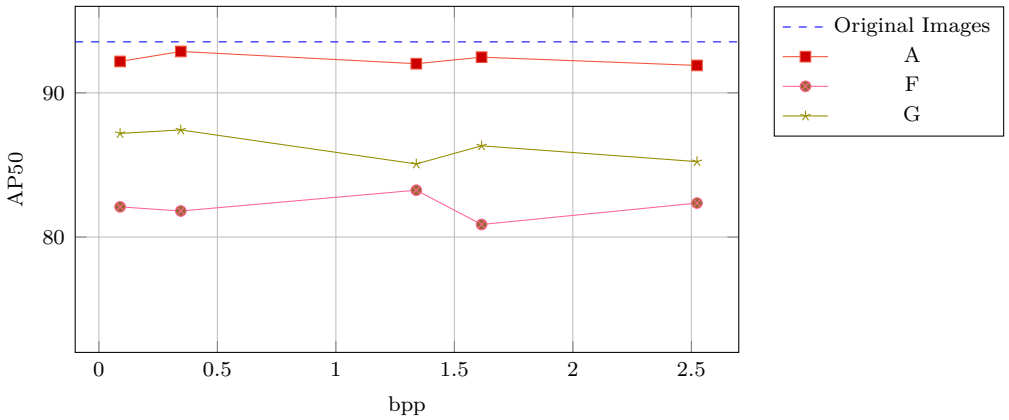


Figure 40: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final Domain Translator architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

The behavior of Model C at the highest rate presents an interesting case. While lower rates demonstrate a clear trend of improved detection performance with increasing rates, the model's performance degrades when processing the highest rate compressed representations (especially in AP75). Analysis of the hyperparameters optimized by the Optuna framework [132] reveals that the initial learning rate (SOLVER.BASE_LR) has the most significant impact on model performance. Experiments show that increasing the initial learning rate improves model performance up to a certain threshold, beyond which further increases lead to model divergence, preventing continued learning.

It is crucial to note that each rate corresponds to a distinct model (set of weights) within the JPEG-AI codec. Consequently, the compressed representations at different rates exist in rather independent domains or latent spaces, with no inherent relationships between them. This independence was verified by attempting to process compressed representations from one rate using a model trained on a different one. The result was a complete absence of detections when processing compressed representations from rates other than the one used for training. This empirical evidence confirms the absence of relationships between compressed domains at different rates. Therefore, it is possible that the Base Domain Translator architecture, when using the PD-FT training process, was insufficient to effectively handle the compressed representations at the highest rate.

The performance instability observed in Models B and C, which incorporate the Base Domain Translator architecture, is overcome when using the LRB preprocessing block. As shown in Figures 40 and 41, implementing the PD-FT training process in conjunction with the LRB block (Model G) enhances detection performance

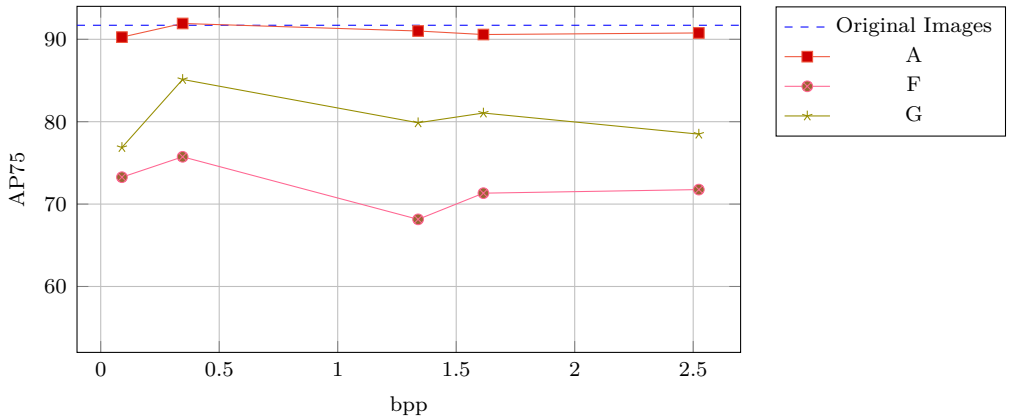


Figure 41: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final compressed domain processing architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

while maintaining stability across all stages. Notably, unlike the Base Domain Translator architecture, models using the LRB block do not experience significant performance degradation at higher rates. Instead, the performance remains stable, showing only a slight decrease of 1.1*pp* from the immediate lower rate (1.61 bpp) - approximately six times less than the decrease observed with the Base Domain Translator architecture. This stability demonstrates the LRB block's effectiveness in extracting useful information from compressed domain representations, even at higher rates.

For the AP50 metric, the most significant performance improvement occurs at lower rates, achieving a 5.62*pp* improvement at 0.35 bpp. The AP75 metric shows a different pattern, with the performance improvements from the PD-FT training process concentrated in the medium rates, reaching an 11.72*pp* improvement at 1.34 bpp. These results demonstrate the effectiveness of preserving and adapting features learned from the pixel domain to the compressed domain context.

4.3.1.3 *Guided Domain Translator Training*

Similarly to the approach adopted in the Section 4.3.1.2, to assess the effectiveness of the GDTT process on model performance, two comparisons were conducted: first, evaluating its impact on the base Domain Translator architecture, and second, examining its effect on the final Domain Translator architecture incorporating the LRB block. This approach enables evaluation of the PD-FT process under comparable conditions.

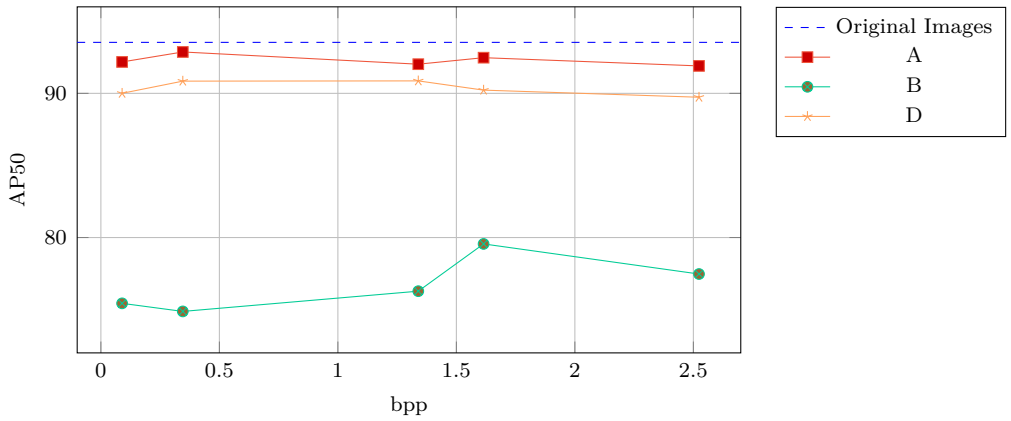


Figure 42: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

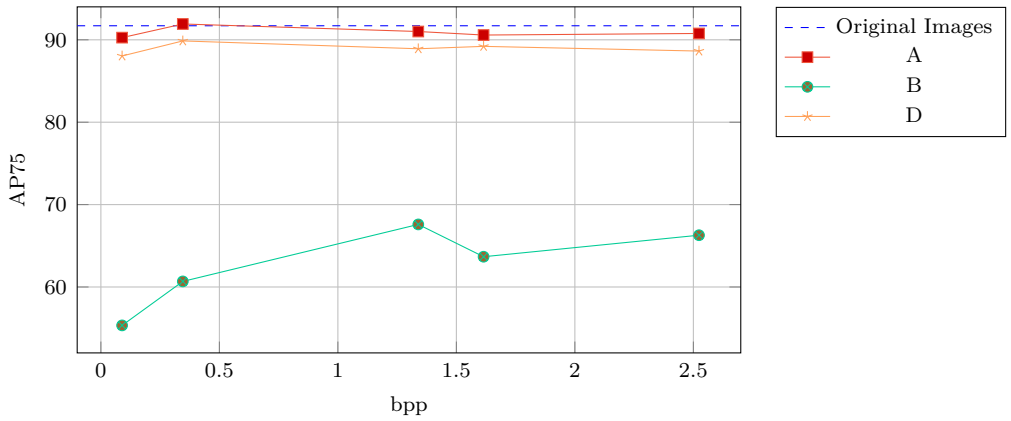


Figure 43: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

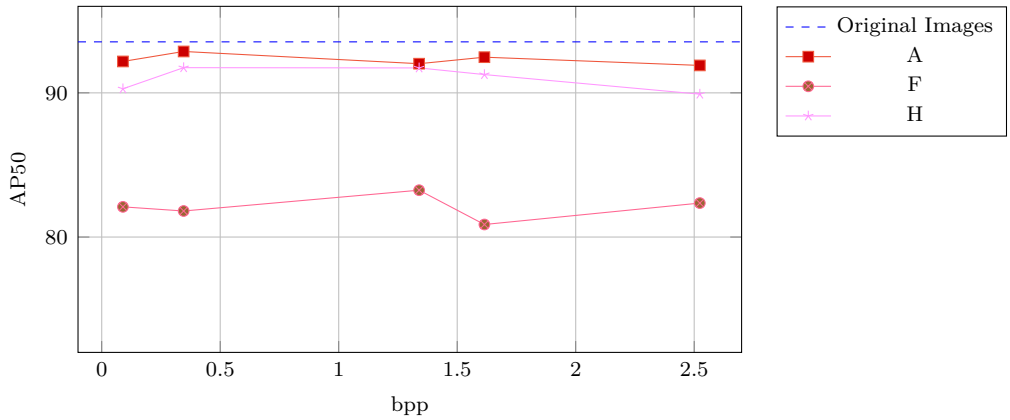


Figure 44: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

Figures 42 and 43 present a performance comparison between the Base model using full training (Model B) and the Base model trained through the GDTT process (Model D), measured in terms of AP50 and AP75, respectively. The adoption of the GDTT process significantly enhances model performance, bringing it remarkably close to the reference model (Model A), which guided the training of the Domain Translator in Model D.

For the AP50 metric, implementing the GDTT process achieved a maximum improvement of 15.97*pp* at 0.35 bpp, representing the highest performance gain for the AP50 metric using the Base Domain Translator Architecture. The AP75 metric showed even more dramatic improvement, peaking at 32.70*pp* at 0.09 bpp. The performance gap between Model D (trained with GDTT) and Model A (the reference model) averages only 1.96*pp* for AP50 and 1.89*pp* for AP75, as detailed in Table 6. The GDTT process proves to be a breakthrough technique, reducing performance loss by 13.45*pp* for AP50 and 25.05*pp* for AP75. These reductions substantially exceed the improvements achieved by individual applications of other ProMIC method components.

One of Model D's most notable characteristics is its consistent performance across all rates. The implementation of the GDTT process reduced the performance standard deviation for AP50 from 1.66*pp* to 0.45*pp*, representing nearly a fourfold improvement in stability. This effect is even more pronounced for the AP75 metric, where the standard deviation decreased from 4.38*pp* to 0.61*pp*, representing more than a sevenfold reduction.

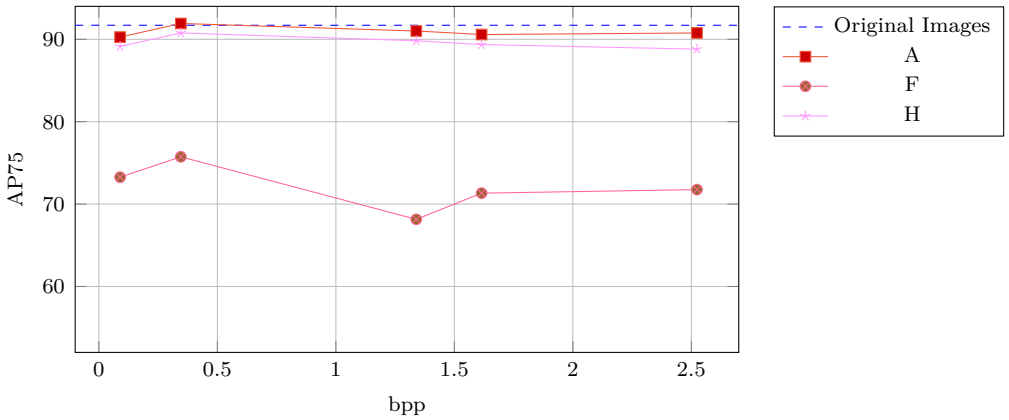


Figure 45: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

When analysing the impact of the GDTT process in conjunction with the LRB block in the Domain Translator architecture, similar patterns emerge as those observed with the Base architecture. As shown in Figures 44 and 45, both AP50 and AP75 metrics demonstrate significant improvement across all rates when applying the GDTT process (Model H), achieving performance levels comparable to the reference model (Model A) when compared to the full training approach (Model F). Although the performance improvements are more modest than those observed with the base Domain Translator architecture, they still reach notable peaks: 10.4*pp* for AP50 at 1.61 bpp and 21.68*pp* for AP75 at 1.34 bpp.

Similar to the base architecture case, implementing the GDTT process reduces the standard deviation in detection performance. For AP50, the standard deviation decreases from 0.77 to 0.75*pp*, while for AP75, it drops from 2.49 to 0.68*pp*. The relatively small reduction in AP50 standard deviation, compared to that observed with the base Domain Translator architecture, can be attributed to Model F's (Domain Translator with LRB block) inherently stable performance across rates. Compared to the base Domain Translator architecture, the model incorporating the LRB block demonstrates significantly better stability, with standard deviations approximately half as large: 2.16 times lower for AP50 and 1.76 times lower for AP75. This pattern underscores the enhanced stability provided by the LRB block in the Domain Translator.

In order to evaluate the combined effectiveness of the PD-FT and *GDTT* training processes, the models using only the Domain Translator trained through the GDTT process are compared against those that additionally employ PD-FT process for full model fine-tuning. This comparison encompasses both Domain Translator

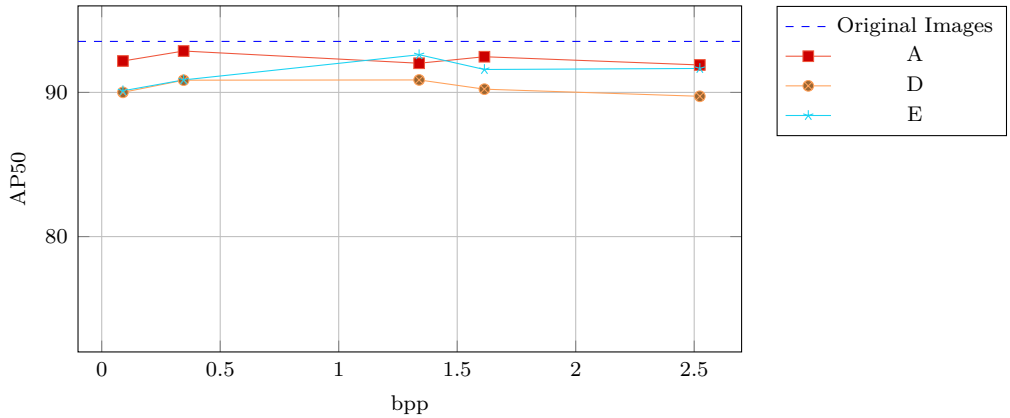


Figure 46: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

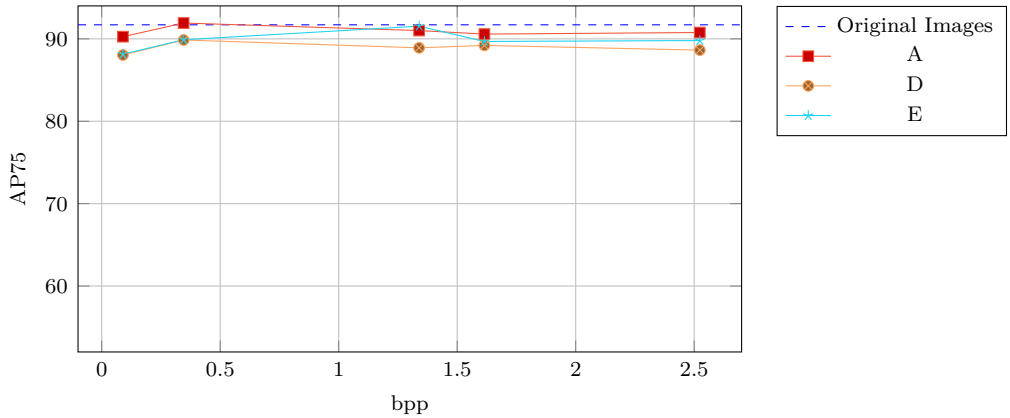


Figure 47: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

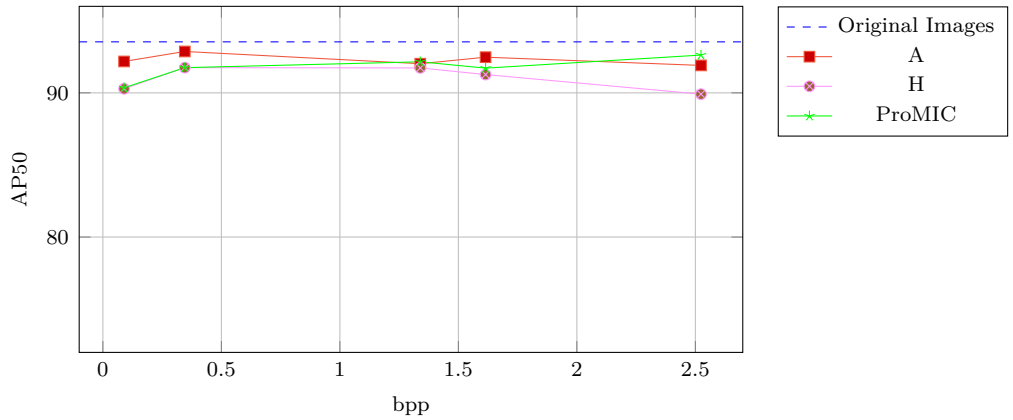


Figure 48: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

architectures: the base architecture and the one integrating the LRB block, with the latter configuration representing the complete proposed ProMIC method.

Figures 46 and 47 present the AP50 and AP75 performance comparison between Model D (base Domain Translator architecture trained through GDTT) and Model E (including additional PD-FT process). The results demonstrate that applying the PD-FT process enhances model performance across all rates. The performance gap between Model E and the reference model diminishes at higher rates, with Model E actually surpassing the reference by $0.59pp$ at 1.34 bpp. This finding indicates that the combination of GDTT and PD-FT processes enables more effective information extraction from compressed domain representations compared to decoded images. Two potential explanations for this behavior are: (1) minor information loss during the decoding process, or (2) the compressed domain representation emphasizes certain useful features that are less apparent in the decoded image, making them more challenging to extract through pixel domain processing layers.

For the Domain Translator architecture incorporating the LRB block, the combined impact of PD-FT and GDTT mirrors the patterns observed in Figures 46 and 47. As shown in Figures 48 and 49, implementing both *GDTT* and PD-FT (Model ProMIC) improves detection performance across all rates for both AP50 and AP75 metrics, with the sole exception of the lowest rate (0.09 bpp), where the ProMIC model shows a slight performance decrease.

Similar to the base Domain Translator architecture case, the ProMIC model surpasses the reference model in both AP50 and AP75 metrics, achieving improve-

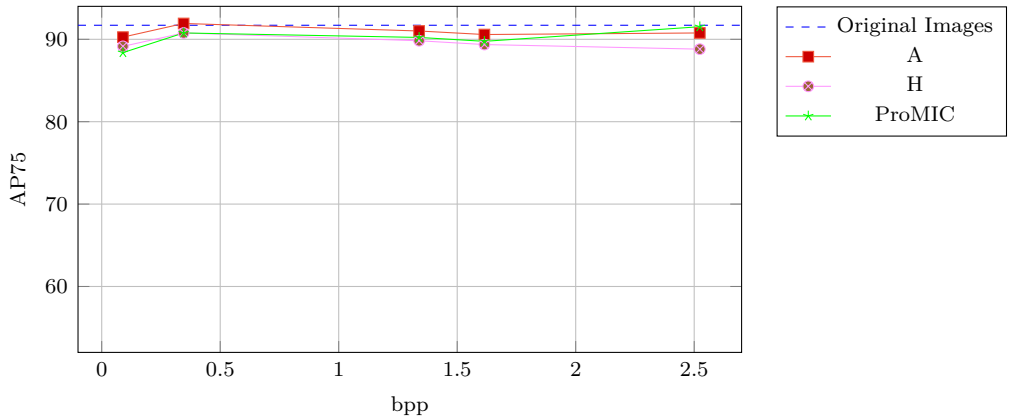


Figure 49: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

ments of 0.72*pp* for AP50 and 0.76*pp* for AP75 at 2.52 bpp. As shown in Table 6, the ProMIC model achieves, on average, a performance loss that is 0.22*pp* lower than Model E for AP50 and 0.05*pp* lower for AP75. These results demonstrate the effectiveness of the proposed method in detecting objects from compressed domain representations.

4.3.2 *Summary*

The ablation study systematically evaluated the contributions of different components and training strategies to the object detection performance of the proposed ProMIC method. The key findings can be summarized as follows:

1. **Lightweight Residual Block (LRB)**: The incorporation of the LRB block improved object detection performance across all compression rates. The hightest enhancements were observed at lower bitrates, where AP50 increased by 6.93*pp* at 0.35 bpp (Figure 36), and AP75 improved by 17.92*pp* at 0.09 bpp (Figure 37). The LRB block also contributed to performance stability, particularly in AP50, where the standard deviation was only 0.77*pp* across all tested bitrates.

2. **Pixel Domain Fine-Tuning (PD-FT)**: The PD-FT process demonstrated effectiveness in preserving and adapting features from the pixel domain to the compressed domain. It significantly improved object detection performance

across various rates, achieving an AP50 improvement of 7.81*pp* at 1.34 bpp (Figure 38) and an AP75 improvement of 19.87*pp* at 0.09 bpp (Figure 39). However, when using the base Domain Translator architecture (which do not incorporate the LRB block) at the highest compression rate (2.52 bpp), the benefits diminished, suggesting that additional adjustments might be necessary to optimize learning in high-rate scenarios when using the base Domain Translator architecture.

3. **Guided Domain Translator Training (GDTT)**: The GDTT process led to substantial improvements in model performance by reducing the performance gap between compressed domain detection and reference models. For AP50, while using the base Domain Translator architecture, the process yielded a maximum gain of 15.97*pp* at 0.35 bpp (Figure 42), while for AP75, the highest improvement was 32.70*pp* at 0.09 bpp (Figure 43). This technique significantly reduced the performance loss observed in baseline models trained with conventional full-training approaches.

4. **Combined Impact and ProMIC Performance**: The integration of all components—LRB, PD-FT, and GDTT—resulted in the ProMIC model achieving the best overall performance. Compared to the baseline model, ProMIC demonstrated substantial improvements across all object detection metrics, reducing the average performance loss significantly. At the highest tested bitrate (2.52 bpp), ProMIC even surpassed the reference model in both AP50 and AP75 metrics (Figures 48 and 49), confirming its capability to effectively extract object detection features directly from the compressed domain.

Overall, the results validate the effectiveness of the proposed approach, demonstrating that object detection in the JPEG-AI compressed domain can achieve performance levels comparable to pixel-domain detection while maintaining efficiency across various compression rates.

4.4 COMPLEXITY ASSESSMENT

In order to evaluate the computational efficiency of the proposed method, a methodology similar to that in [78] was adopted, measuring the number of operations required for execution and comparing it with the number of operations needed for the reference model (Model A), which performs detection on the reconstructed images. The number of operations is expressed in terms of Giga Multiplication and Accumulation (GMAC) operations.

Table 7 presents the complexity of both the reference model (Model A), which includes both image reconstruction and detection processes, and the proposed method (ProMIC), which considers only the processing of the Domain Translator and the pruned object detector. Notably, while the complexity of image reconstruction varies with the selected quality, it has a low standard deviation of just 1.35 GMAC. Therefore, these variations are not considered in the analysis. The Delta column indicates the complexity change relative to the reference model, where a negative value represents a reduction in complexity. The results demonstrate that the proposed method significantly reduces system complexity, decreasing from 315.52 GMAC to 181.93 GMAC. This corresponds to a 42.34% reduction compared to the reference model, underscoring the efficiency of the proposed method in achieving highly competitive, and even superior, results while drastically reducing computational complexity.

Table 7: Computational complexity comparison between the reference model (detection on reconstructed images) and the proposed ProMIC method (detection in compressed domain). Values are presented in Giga Multiplication and Accumulation (GMAC) operations. Delta value indicates computational complexity reduction achieved by the ProMIC method relative to the reference model.

Model	Complexity (GMAC)	Delta (%)
A (Ref.)	315.52	-
ProMIC	181.93	-42.34

Table 8: Computational complexity comparison between the base Domain Translator architecture and the final Domain Translator architecture that integrates the Lightweight Residual Block (LRB). The values are expressed in Giga Multiplication and Accumulation (GMAC) operations. The Delta value represents the reduction in computational complexity achieved by the final Domain Translator architecture compared to the base architecture.

Domain Translator	Complexity (GMAC)	Delta (%)
Base	33.52	-
Base + LRB	33.60	2.39×10^{-3}

To analyze the impact of the proposed Domain Translator module on the system, it was measured the computational complexity of the two proposed Domain Translator architectures: the base architecture and the version that integrates the LRB block for preprocessing compressed representations.

Table 8 presents the complexity of both Domain Translator architectures considered in this study. The complexity is expressed in GMAC, and the Delta column indicates the complexity change relative to the base architecture. The results show that the complexity difference between the two architectures is minimal, not even reaching 1 GMAC: 0.08 GMAC, corresponding to a negligible 2.39×10^{-3}% increase.

However, as discussed in Section 4.3.1, it should be emphasized that the proposed method significantly enhances performance and stabilizes detection accuracy across different rates by minimizing standard deviation. Therefore, given the substantial benefits compared to the minimal complexity overhead, integrating the LRB block into the Domain Translator architecture is justified as an essential component of the system.

5

CONCLUSIONS AND FUTURE WORK

This work has explored the processing of microscopy images in the compressed domain. A novel approach is proposed (ProMIC), for object detection in images' latent representations generated by a learning-based encoder. By eliminating the need for full decompression, the proposed method enhances computational efficiency while maintaining or even improving detection accuracy compared to pixel-domain approaches.

The study also included a preliminary analysis of JPEG-AI compression efficiency against classical codecs (HEVC/H.265 and VVC/H.266) across multiple datasets. The results demonstrated that the performance of the JPEG-AI codec remains similar across different configurations (different operation points and tools employed) and becomes closer to the classical codecs at higher bitrates, highlighting the advantages of learning-based compression in preserving image quality for downstream tasks.

The ProMIC architecture incorporates a Domain Translator to enable object detection networks to effectively process compressed-domain features. It integrates key functionalities such as the Lightweight Residual Block, Pixel Domain Fine-Tuning, and Guided Domain Translator Training, which collectively enhance object detection performance. These components improve detection accuracy, particularly at lower bitrates, by stabilizing performance and reducing the gap between compressed-domain and pixel-domain models. The integration of these techniques led to the highest observed detection accuracy, with ProMIC surpassing both the reference pixel-domain model and the compressed-domain baseline in various scenarios, particularly at high bitrates. Furthermore, ProMIC significantly reduces computational complexity, achieving a 42.34% reduction in operations compared to the reference model that operates on reconstructed images.

The results of this research highlight the potential of compressed-domain image processing for computationally constrained environments, such as embedded systems, real-time processing, and large-scale image analysis. By eliminating full decompression, this approach not only reduces computational costs but also minimizes energy consumption, leading to more efficient data storage, transmission, and overall system performance.

5.1 FUTURE WORK

This work has demonstrated the effectiveness of the proposed ProMIC method, paving avenues for further research. In terms of the proposed model itself, one of the most promising directions is the development of an “online” configuration for the Guided Domain Translator Training (GDTT) process. Indeed, a ”real-time”-like update would allow the domain translator to adapt more flexibly, dynamically adjusting its parameters as new data is processed. This approach would ensure that equivalent information is extracted from latent representations, similar to those obtained from images, while simultaneously optimising the entire model to enhance detection performance. This multi-task approach could enhance the model's robustness to variations in dataset distributions.

Future research could also explore:

- **Extension to Other Vision Tasks**: Investigating the applicability of ProMIC for other tasks such as segmentation, tracking, or recognition in the compressed domain.
- **Application to more recent Object Detection and Segmentation Models**: Exploring the integration of the proposed methodology with state-of-the-art object detection and segmentation models, such as YOLOv12 and Segment Anything Model (SAM) 2, could enhance its practical applicability. These models represent the latest advancements in deep learning for vision tasks, and adapting ProMIC to work with them could further improve detection accuracy and efficiency in the compressed domain.
- **Dynamically Selecting the Most Important Channels from the Compressed Representation**: By identifying and utilizing only the most relevant channels in the latent representation, the complexity of the Domain Translator—and consequently, the overall Compressed Domain Object Detector—could be significantly reduced. This would lead to more efficient computation, making the method more suitable for real-time applications and deployment on resource-limited hardware.

By addressing these challenges, the proposed framework could evolve into a more versatile and widely applicable solution for compressed-domain image processing, paving the way for future advancements in AI-driven image analysis.

BIBLIOGRAPHY

[1] B. Bross, Y.-K. Wang, Y. Ye, *et al.*, "Overview of the versatile video coding (vvc) standard and its applications", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021. DOI: 10.1109/TCSVT.2021.3101953.

[2] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc)", *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1463–1493, 2021. DOI: 10.1109/JPROC.2020.3043399.

[3] D. Wu, D. Tan, M. Baird, J. DeCampo, C. White, and H. R. Wu, "Perceptually lossless medical image coding", *IEEE Transactions on Medical Imaging*, vol. 25, no. 3, pp. 335–344, 2006. DOI: 10.1109/TMI.2006.870483.

[4] 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*, 2015, pp. 406–410. DOI: 10.1109/BigMM.2015.16.

[5] J. Ascenso, E. Alshina, and T. Ebrahimi, "The jpeg ai standard: Providing efficient human and machine visual data consumption", *IEEE MultiMedia*, vol. 30, no. 1, pp. 100–111, 2023. DOI: 10.1109/MMUL.2023.3245919.

[6] Z. Cheng, P. Akyazi, H. Sun, J. Katto, and T. Ebrahimi, "Perceptual quality study on deep learning based image compression", in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 719–723. DOI: 10.1109/ICIP.2019.8803824.

[7] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior", *arxiv*, 2018.

[8] A. Bhardwaj, "Promise and provisos of artificial intelligence and machine learning in healthcare", en, *J. Healthc. Leadersh.*, vol. 14, pp. 113–118, Jul. 2022.

[9] L. Kaestner, "Artificial intelligence meets hematology", en, *Transfus. Apher. Sci.*, vol. 59, no. 6, p. 102 986, Dec. 2020.

[10] Y. Arai, T. Kondo, K. Fuse, *et al.*, "Using a machine learning algorithm to predict acute graft-versus-host disease following allogeneic transplantation", en, *Blood Adv.*, vol. 3, no. 22, pp. 3626–3634, Nov. 2019.

[11] A. Esteva, B. Kuprel, R. A. Novoa, *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks", en, *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017.

[12] S. Chan, V. Reddy, B. Myers, Q. Thibodeaux, N. Brownstone, and W. Liao, "Machine learning in dermatology: Current applications, opportunities, and limitations", en, *Dermatol. Ther. (Heidelb.)*, vol. 10, no. 3, pp. 365–386, Jun. 2020.

[13] B. Ehteshami Bejnordi, M. Veta, P. Johannes van Diest, *et al.*, "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer", *JAMA*, vol. 318, no. 22, p. 2199, Dec. 2017.

[14] A. Sharma and R. Rani, "A systematic review of applications of machine learning in cancer prediction and diagnosis", en, *Arch. Comput. Methods Eng.*, vol. 28, no. 7, pp. 4875–4896, Dec. 2021.

[15] K. A. Tran, O. Kondrashova, A. Bradley, E. D. Williams, J. V. Pearson, and N. Waddell, "Deep learning in cancer diagnosis, prognosis and treatment selection", en, *Genome Med.*, vol. 13, no. 1, p. 152, Sep. 2021.

[16] V. Gulshan, L. Peng, M. Coram, *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs", *JAMA*, vol. 316, no. 22, p. 2402, Dec. 2016.

[17] P. Rajpurkar, J. Irvin, K. Zhu, *et al.*, "CheXNet: Radiologist-level pneumonia detection on chest x-rays with deep learning", *arXiv*, Nov. 2017. arXiv: 1711.05225 [cs.CV].

[18] E. Sorantin, M. G. Grasser, A. Hemmelmayr, *et al.*, "The augmented radiologist: Artificial intelligence in the practice of radiology", en, *Pediatr. Radiol.*, vol. 52, no. 11, pp. 2074–2086, Oct. 2022.

[19] K. Ganapathy, "Artificial intelligence in neurosciences—are we really there?", in *Augmenting Neurological Disorder Prediction and Rehabilitation Using Artificial Intelligence*, Elsevier, 2022, pp. 177–191.

[20] A. A.-A. Valliani, D. Ranti, and E. K. Oermann, "Deep learning and neurology: A systematic review", en, *Neurol. Ther.*, vol. 8, no. 2, pp. 351–365, Dec. 2019.

[21] F. Wang, M. Jiang, C. Qian, *et al.*, "Residual attention network for image classification", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6450–6458. DOI: 10.1109/CVPR.2017.683.

[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[23] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[24] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer", in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1833–1844. DOI: 10.1109/ICCVW54120.2021.00210.

[25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models", in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10 674–10 685. DOI: 10.1109/CVPR52688.2022.01042.

[26] C. Li, C. Guo, L. Han, *et al.*, "Low-light image and video enhancement using deep learning: A survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9396–9416, 2022. DOI: 10.1109/TPAMI.2021.3126387.

[27] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "Repaint: Inpainting using denoising diffusion probabilistic models", in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 451–11 461. DOI: 10.1109/CVPR52688.2022.01117.

[28] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression", *arXiv*, 2016.

[29] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression", *arxiv*, 2018. eprint: 1809.02736 (cs.CV).

[30] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality", in *2016 Picture Coding Symposium (PCS)*, 2016, pp. 1–5. DOI: 10.1109/PCS.2016.7906310.

[31] N. Luka, R. Negrel, and D. Picard, "Image compression using only attention based neural networks", *arxiv*, 2023. eprint: 2310.11265 (eess.IV).

[32] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules", in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7936–7945. DOI: 10.1109/CVPR42600.2020.00796.

[33] E. Alshina, J. Ascenso, and T. Ebrahimi, "Jpeg ai: The first international standard for image coding based on an end-to-end learning-based approach", *IEEE MultiMedia*, vol. 31, no. 4, pp. 60–69, 2024. DOI: 10.1109/MMUL.2024.3485255.

[34] Y. Zhang and Q. Yang, "A survey on multi-task learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022. DOI: 10.1109/TKDE.2021.3070203.

[35] Y. Zhao, X. Wang, T. Che, G. Bao, and S. Li, "Multi-task deep learning for medical image computing and analysis: A review", *Computers in Biology and Medicine*, vol. 153, p. 106 496, 2023, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.106496.

[36] S. Wang, Q. Wang, and M. Gong, "Multi-Task learning based network embedding", en, *Frontiers in Neuroscience*, vol. 13, p. 1387, 2019.

[37] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey", en, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3614–3633, Jul. 2022.

[38] H. Shi, S. Ren, T. Zhang, and S. J. Pan, "Deep multitask learning with progressive parameter sharing", in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 19 867–19 878. DOI: 10.1109/ICCV51070.2023.01824.

[39] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5454–5463. DOI: 10.1109/CVPR.2017.579.

[40] M. Long, Z. Cao, J. Wang, and P. S. Yu, "Learning multiple tasks with multilinear relationship networks", in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 1593–1602, ISBN: 9781510860964.

[41] S. Ruder, "An overview of multi-task learning in deep neural networks", *arXiv*, 2017. DOI: 10.48550/arXiv.1706.05098.

[42] J. Li, X. Liu, W. Yin, M. Yang, L. Ma, and Y. Jin, "Empirical evaluation of multi-task learning in deep neural networks for natural language processing", en, *Neural Comput. Appl.*, Aug. 2020.

[43] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3994–4003. DOI: 10.1109/CVPR.2016.433.

[44] L. Xiao, H. Zhang, and W. Chen, "Gated multi-task network for text classification", in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 726–731. DOI: 10.18653/v1/N18-2114.

[45] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Latent Multi-Task architecture learning", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4822–4829, Jul. 2019.

[46] L. Liu, T. Chen, H. Liu, S. Pu, L. Wang, and Q. Shen, "2C-Net: Integrate image compression and classification via deep neural network", en, *Multimed. Syst.*, vol. 29, no. 3, pp. 945–959, Jun. 2023.

[47] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing", in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 675–684. DOI: 10.1109/CVPR.2018.00077.

[48] R. Zhang, S. Tan, R. Wang, *et al.*, "Biomarker localization by combining CNN classifier and generative adversarial network", in *Lecture Notes in Computer Science*, ser. Lecture notes in computer science, Cham: Springer International Publishing, 2019, pp. 209–217.

[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?", in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014.

[50] X. Xu, C. Lian, S. Wang, *et al.*, "Asymmetric multi-task attention network for prostate bed segmentation in computed tomography images", *Medical Image Analysis*, vol. 72, p. 102 116, 2021, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2021.102116.

[51] X. Wang, L. Jiang, L. Li, *et al.*, "Joint learning of 3d lesion segmentation and classification for explainable covid-19 diagnosis", *IEEE Transactions on Medical Imaging*, vol. 40, no. 9, pp. 2463–2476, 2021. DOI: 10.1109/TMI.2021.3079709.

[52] H. Xie, H. Lei, X. Zeng, *et al.*, "Amd-gan: Attention encoder and multi-branch structure based generative adversarial networks for fundus disease detection from scanning laser ophthalmoscopy images", *Neural Networks*, vol. 132, pp. 477–490, 2020, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2020.09.005.

[53] J. Zhao, J. Huang, D. Zhi, *et al.*, "Functional network connectivity (FNC)-based generative adversarial network (GAN) and its applications in classification of mental disorders", en, *Journal of Neuroscience Methods*, vol. 341, no. 108756, p. 108 756, Jul. 2020.

[54] S. Marques, F. Schiavo, C. A. Ferreira, J. Pedrosa, A. Cunha, and A. Campilho, "A multi-task CNN approach for lung nodule malignancy classification and characterization", en, *Expert Systems with Applications*, vol. 184, no. 115469, p. 115 469, Dec. 2021.

[55] J. Chen, G. Yang, H. Khan, *et al.*, "Jas-gan: Generative adversarial network based joint atrium and scar segmentations on unbalanced atrial targets", *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 1, pp. 103–114, 2022, Cited by: 49; All Open Access, Green Open Access. DOI: 10.1109/JBHI.2021.3077469.

[56] X. Xiong, Y. Ding, C. Sun, *et al.*, "A cascaded multi-task generative framework for detecting aortic dissection on 3-d non-contrast-enhanced computed tomography", *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 10, pp. 5177–5188, 2022. DOI: 10.1109/JBHI.2022.3190293.

[57] Q. Wang, W. Xue, X. Zhang, F. Jin, and J. Hahn, "Pixel-wise body composition prediction with a multi-task conditional generative adversarial network", en, *Journal of Biomedical Informatics*, vol. 120, no. 103866, p. 103 866, Aug. 2021.

[58] J. Xue, K. He, D. Nie, *et al.*, "Cascaded multitask 3-d fully convolutional networks for pancreas segmentation", *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 2153–2165, 2021. DOI: 10.1109/TCYB.2019.2955178.

[59] L. Song, J. Lin, Z. J. Wang, and H. Wang, "An end-to-end multi-task deep learning framework for skin lesion analysis", *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2912–2921, 2020. DOI: 10.1109/JBHI.2020.2973614.

[60] A. Abbasi, E. Miahi, and S. A. Mirroshandel, "Effect of deep transfer and multi-task learning on sperm abnormality detection", en, *Computers in Biology and Medicine*, vol. 128, no. 104121, p. 104 121, Jan. 2021.

[61] H. Shi, S. Ren, T. Zhang, and S. J. Pan, "Deep multitask learning with progressive parameter sharing", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 19 924–19 935.

[62] X. Fu, G. Yang, K. Zhang, N. Xu, and J. Wu, "An automated estimator for cobb angle measurement using multi-task networks", en, *Neural Computing and Applications*, Nov. 2020.

[63] J. Zhang, W. Hu, S. Li, *et al.*, "Chromosome classification and straightening based on an interleaved and multi-task network", *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 8, pp. 3240–3251, 2021. DOI: 10.1109/JBHI.2021.3062234.

[64] C. Xu, J. Howey, P. Ohorodnyk, M. Roth, H. Zhang, and S. Li, "Segmentation and quantification of infarction without contrast agents via spatiotemporal generative adversarial learning", en, *Medical Image Analysis*, vol. 59, no. 101568, p. 101 568, Jan. 2020.

[65] Y.-H. Wu, S.-H. Gao, J. Mei, *et al.*, "Jcs: An explainable covid-19 diagnosis system by joint classification and segmentation", *IEEE Transactions on Image Processing*, vol. 30, pp. 3113–3126, 2021. DOI: 10.1109/TIP.2021.3058783.

[66] S. Tsutsui, T. Hirakawa, T. Yamashita, and H. Fujiyoshi, "Semantic segmentation and change detection by multi-task u-net", in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 619–623. DOI: 10.1109/ICIP42928.2021.9506560.

[67] M. Xu, K. Huang, and X. Qi, "Multi-task learning with context-oriented self-attention for breast ultrasound image classification and segmentation", in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, 2022, pp. 1–5. DOI: 10.1109/ISBI52829.2022.9761685.

[68] X. Wang, M. Xu, J. Zhang, L. Jiang, and L. Li, "Deep multi-task learning for diabetic retinopathy grading in fundus images", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, pp. 2826–2834, May 2021.

[69] S. Vesal, M. Gu, A. Maier, and N. Ravikumar, "Spatio-temporal multi-task learning for cardiac mri left ventricle quantification", *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 7, pp. 2698–2709, 2021. DOI: 10.1109/JBHI.2020.3046449.

[70] R. Zhang, X. Xiao, Z. Liu, Y. Li, and S. Li, "Mrln: Multi-task relational learning network for mri vertebral localization, identification, and segmentation", *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2902–2911, 2020. DOI: 10.1109/JBHI.2020.2969084.

[71] X. Wang, Y. Fang, S. Yang, *et al.*, "A hybrid network for automatic hepatocellular carcinoma segmentation in H&E-stained whole slide images", en, *Medical Image Analysis*, vol. 68, no. 101914, p. 101 914, Feb. 2021.

[72] R. Torfason, F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Towards image understanding from deep compression without decoding", 2018. eprint: 1803.06131 (cs.CV).

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *axiv*, 2015. eprint: 1512.03385 (cs.CV).

[74] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs", *axiv*, 2016. eprint: 1606.00915 (cs.CV).

[75] Q. Shen, J. Cai, L. Liu, *et al.*, "Codedvision: Towards joint image understanding and compression via end-to-end learning", in *Advances in Multimedia Information Processing – PCM 2018*, R. Hong, W.-H. Cheng, T. Yamasaki, M. Wang, and C.-W. Ngo, Eds., Cham: Springer International Publishing, 2018, pp. 3–14, ISBN: 978-3-030-00776-8.

[76] Y. Deng and L. Karam, "Learning-based compression for material and texture recognition", *ArXiv*, vol. abs/2104.10065, 2021.

[77] Tensorflow, *GitHub - tensorflow/compression: Data compression in TensorFlow — github.com*, https://github.com/tensorflow/compression, [Accessed 27-02-2025].

[78] A. Alkhateeb, A. Gnutti, F. Guerrini, R. Leonardi, J. Ascenso, and F. Pereira, "Jpeg ai compressed domain face detection", in *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, 2024, pp. 1–6. DOI: 10.1109/MMSP61759.2024.10743984.

[79] M. Trigka and E. Dritsas, "A comprehensive survey of machine learning techniques and models for object detection", en, *Sensors (Basel)*, vol. 25, no. 1, Jan. 2025.

[80] R.-A. Bratulescu, R.-I. Vatasoiu, G. Sucic, S.-A. Mitroi, M.-C. Vochin, and M.-A. Sachian, "Object detection in autonomous vehicles", in *2022 25th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2022, pp. 375–380. DOI: 10.1109/WPMC55625.2022.10014804.

[81] A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya, "Object detection algorithms for video surveillance applications", in *2018 International Conference on Communication and Signal Processing (ICCSP)*, 2018, pp. 0563–0568. DOI: 10.1109/ICCSP.2018.8524461.

[82] M. Ilyas, H. Y. Khaw, N. M. Selvaraj, Y. Jin, X. Zhao, and C. C. Cheah, "Robot-assisted object detection for construction automation: Data and information-driven approach", *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 6, pp. 2845–2856, 2021. DOI: 10.1109/TMECH.2021.3100306.

[83] C. Albuquerque, R. Henriques, and M. Castelli, "Deep learning-based object detection algorithms in medical imaging: Systematic review", *Heliyon*, vol. 11, no. 1, e41137, 2025, ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2024.e41137. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240584402417168X.

[84] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey", *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023. DOI: 10.1109/JPROC.2023.3238524.

[85] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[86] Z. Chen, K. Chen, and J. Chen, "Vehicle and pedestrian detection using support vector machine and histogram of oriented gradients features", in *2013 International Conference on Computer Sciences and Applications*, 2013, pp. 365–368. DOI: 10.1109/CSA.2013.92.

[87] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[88] Z. Wang, H. Xiao, W. He, F. Wen, and K. Yuan, "Real-time sift-based object recognition system", in *2013 IEEE International Conference on Mechatronics and Automation*, 2013, pp. 1361–1366. DOI: 10.1109/ICMA.2013.6618111.

[89] H. R. Kher and V. K. Thakar, "Scale invariant feature transform based image matching and registration", in *2014 Fifth International Conference on Signal and Image Processing*, 2014, pp. 50–55. DOI: 10.1109/ICSIP.2014.12.

[90] S. A. Nawaz, J. Li, U. A. Bhatti, M. U. Shoukat, and R. M. Ahmad, "AI-based object detection latest trends in remote sensing, multimedia and agriculture applications", en, *Front. Plant Sci.*, vol. 13, p. 1 041 514, Nov. 2022.

[91] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017. DOI: 10.1109/TPAMI.2016.2577031.

[92] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn", in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[93] M. Karthi, V. Muthulakshmi, R. Priscilla, P. Praveen, and K. Vanisri, "Evolution of yolo-v5 algorithm for object detection: Automated detection of library books and performace validation of dataset", in *2021 International Conference on Innovative Computing, Intelligent Communication and Smart*

Electrical Systems (ICSES), 2021, pp. 1–6. DOI: 10.1109/ICSES52305.2021.9633834.

[94] R. Varghese and S. M., "Yolov8: A novel object detection algorithm with enhanced performance and robustness", in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024, pp. 1–6. DOI: 10.1109/ADICS58448.2024.10533619.

[95] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single shot MultiBox detector", in *Computer Vision – ECCV 2016*, ser. Lecture notes in computer science, Cham: Springer International Publishing, 2016, pp. 21–37.

[96] C. Albuquerque, R. Henriques, and M. Castelli, "Deep learning-based object detection algorithms in medical imaging: Systematic review", en, *Heliyon*, vol. 11, no. 1, e41137, Jan. 2025.

[97] W. Fazli, M. Yingliang, K. Dawar, A. Muhammad, and B. Syed U K, "Biomedical image segmentation: A systematic literature review of deep learning based object detection methods", *arxiv*, 2024. eprint: 2408.03393 (eess.IV).

[98] M. G. Ragab, S. J. Abdulkadir, A. Muneer, *et al.*, "A comprehensive systematic review of yolo for medical object detection (2018 to 2023)", *IEEE Access*, vol. 12, pp. 57 815–57 836, 2024. DOI: 10.1109/ACCESS.2024.3386826.

[99] A. Baccouche, B. Garcia-Zapirain, C. Castillo Olea, and A. S. Elmaghraby, "Breast lesions detection and classification via YOLO-based fusion models", en, *Comput. Mater. Contin.*, vol. 69, no. 1, pp. 1407–1425, 2021.

[100] M. Z. Khaliki and M. S. Başarslan, "Brain tumor detection from images and comparison with transfer learning methods and 3-layer CNN", en, *Sci. Rep.*, vol. 14, no. 1, p. 2664, Feb. 2024.

[101] P. Zeng, S. Liu, S. He, *et al.*, "Tuspm-net: A multi-task model for thyroid ultrasound standard plane recognition and detection of key anatomical structures of the thyroid", *Computers in Biology and Medicine*, vol. 163, p. 107 069, 2023, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2023.107069. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482523005346.

[102] C. A. Fischer, L. Besora-Casals, S. G. Rolland, *et al.*, "MitoSegNet: Easy-to-use deep learning segmentation for analyzing mitochondrial morphology", en, *iScience*, vol. 23, no. 10, p. 101 601, Oct. 2020.

[103] J. Liu, W. Li, C. Xiao, B. Hong, Q. Xie, and H. Han, "Automatic detection and segmentation of mitochondria from sem images using deep neural network", in *2018 40th Annual International Conference of the IEEE Engi-*

neering in Medicine and Biology Society (EMBC), 2018, pp. 628–631. DOI: 10.1109/EMBC.2018.8512393.

[104] Z. S. Hossein, R. P. Mohammadiani, and S. Izadi, "Mitochondrial segmentation in microscopy images using unet-vgg19", in *2024 14th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2024, pp. 125–130. DOI: 10.1109/ICCKE65377.2024.10874572.

[105] Y. Ding, J. Li, J. Zhang, *et al.*, "Mitochondrial segmentation and function prediction in live-cell images with deep learning", en, *Nat. Commun.*, vol. 16, no. 1, p. 743, Jan. 2025.

[106] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms", in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[107] V. Casser, K. Kang, H. Pfister, and D. Haehn, "Fast mitochondria detection for connectomics", in *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, T. Arbel, I. Ben Ayed, M. de Bruijne, M. Descoteaux, H. Lombaert, and C. Pal, Eds., ser. Proceedings of Machine Learning Research, vol. 121, PMLR, Jun. 2020, pp. 111–120. [Online]. Available: https://proceedings.mlr.press/v121/casser20a.html.

[108] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context", in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1. [Online]. Available: https://cocodataset.org/#detection-eval.

[109] Y. Blau and T. Michaeli, "Rethinking lossy compression: The rate-distortion-perception tradeoff", in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 675–685. [Online]. Available: https://proceedings.mlr.press/v97/blau19a.html.

[110] G. Bjøntegaard, "Calculation of average psnr differences between rd-curves", 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:61598325.

[111] Z. Luo, W. Jia, and S. Perry, "Compressed point cloud classification with point-based edge sampling", en, *EURASIP J. Image Video Process.*, vol. 2024, no. 1, Aug. 2024.

[112] A. V. Katsenou, F. Zhang, M. Afonso, and D. R. Bull, "A subjective comparison of av1 and hevc for adaptive video streaming", in *2019 IEEE Inter-*

national Conference on Image Processing (ICIP), 2019, pp. 4145–4149. DOI: 10.1109/ICIP.2019.8803523.

[113] "Jpeg ai common training and test conditions", ISO/IEC JTC 1/SC 29/WG 1, Covilhã, Portugal, Standard, Jul. 2023, 100th JPEG Meeting, 17th-21st July 2023.

[114] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment", in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, 2003, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.

[115] Z. Wang and Q. Li, "Information content weighting for perceptual image quality assessment", *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1185–1198, 2011. DOI: 10.1109/TIP.2010.2092435.

[116] Z. Li and M. Manohara, *Toward A Practical Perceptual Video Quality Metric — netflixtechblog.com*, https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652, [Accessed 11-02-2025], 2016-6-6.

[117] H. Sheikh and A. Bovik, "Image information and visual quality", *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006. DOI: 10.1109/TIP.2005.859378.

[118] N. Ponomarenko, F. Silvestri, K. Egiazarian, M. Carli, J. Astola, and V. Lukin, "On between-coefficient contrast masking of dct basis functions", English, in *Proceedings of the Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics, VPQM 2007, Scottsdale, Arizona, USA, 25-26 January 2007*, Contribution: organisation=sgn,FACT1=1, 2007, 4 p.

[119] A. B. Watson and J. A. Solomon, "Model of visual contrast gain control and pattern masking", en, *J. Opt. Soc. Am. A Opt. Image Sci. Vis.*, vol. 14, no. 9, pp. 2379–2391, Sep. 1997.

[120] V. Laparra, J. Ballé, A. Berardino, and E. P. Simoncelli, "Perceptual image quality assessment using a normalized laplacian pyramid", in *Human Vision and Electronic Imaging*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:1036792.

[121] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment", *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011. DOI: 10.1109/TIP.2011.2109730.

[122] A. Lucchi, Y. Li, K. Smith, and P. Fua, "Structured image segmentation using kernelized features", in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 400–413, ISBN: 978-3-642-33709-3.

[123] E. S. Paulo, "Lossy compression of biomedical images for computer vision analysis", Available at http://hdl.handle.net/10400.8/10029, Master's thesis, School of Technology and Management, Leiria, Portugal, Jun. 2024.

[124] N. Kasthuri, K. J. Hayworth, D. R. Berger, *et al.*, "Saturated reconstruction of a volume of neocortex", en, *Cell*, vol. 162, no. 3, pp. 648–661, Jul. 2015.

[125] Delmic, *Large-scale em imaging. faster and automated.* [Online]. Available: https://www.delmic.com/en/products/fast-imaging/fast-em.

[126] "Information technology - JPEG AI learning-based image coding system - Part 1: Core coding system, ISO/IEC PRF 6048-1", International Organization for Standardization, Geneva, CH, Standard, 2025.

[127] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2, 2019.

[128] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, *Detectron*, https://github.com/facebookresearch/detectron, 2018.

[129] M. Butt, N. Glas, J. Monsuur, R. Stoop, and A. de Keijzer, "Application of YOLOv8 and detectron2 for bullet hole detection and score calculation from shooting cards", en, *AI (Basel)*, vol. 5, no. 1, pp. 72–90, Dec. 2023.

[130] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[131] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation", in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 1180–1189.

[132] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework", in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

APPENDIX

A

APPENDIX A

Table 9: Number IDs used for the training and validation sets of the *Lucchi++* dataset.

Dataset	**Instance number ID**	
	Train set	**Validation set**
Lucchi++	0,2,3,5,6,8,9,11,12,14,15,16, 17,20,21,22,25,26,28,29,30,32, 33,34,35,36,37,38,39,41,42,43, 45,46,48,49,50,51,53,54,55,56, 57,59,60,61,63,65,66,67,69,70, 72,73,74,75,77,79,80,81,82,84, 85,86,87,88,89,91,92,93,94,96, 97,99,100,102,103,104,105,107, 109,110,111,112,114,115,116,117, 119,120,121,122,124,125,126,127, 129,130,131,133,134,135,136,137, 138,140,142,143,144,146,147,148, 149,151,152,153,154,156,157,158, 159,161,162,163,164	1,4,7,10,13,18, 19,23,24,27,31, 40, 44,47,52,58, 62,64,68,71,76, 78,83, 90,95,98, 101,106,108,113, 118,123,128,132, 139,141,145,150, 155,160

B

APPENDIX B

Table 10: Optimised parameter values for the pixel-domain detection model when applied to uncompressed images.

Parameter	Value
SOLVER.IMS_PER_BATCH	1
SOLVER.BASE_LR	1.88×10^{-3}
Delta Learning Rate	1.01×10^{-3}
Initial Iteration	1328
Number of Steps	38
Iteration Step	915

Table 11: Optimised parameter values for the Model A at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	1	4	2	4	2
SOLVER.BASE_LR	8.30×10^{-4}	3.63×10^{-3}	1.12×10^{-3}	2.82×10^{-3}	6.00×10^{-3}
Delta Learning Rate	1.00×10^{-4}	1.00×10^{-4}	1.00×10^{-4}	1.00×10^{-4}	1.00×10^{-4}
Initial Iteration	825	688	998	577	5000
Number of Steps	65	36	37	63	0
Iteration Step	399	823	480	507	1000

Table 12: Optimised parameter values for the Model B at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	6	5	8	3	9
SOLVER.BASE_LR	7.46×10^{-2}	6.47×10^{-2}	3.78×10^{-2}	2.88×10^{-3}	6.26×10^{-3}
Delta Learning Rate	7.46×10^{-2}	6.47×10^{-2}	3.78×10^{-2}	1.62×10^{-4}	1.12×10^{-4}
Initial Iteration	379	1911	1352	585	4000
Number of Steps	2	56	71	59	3
Iteration Step	184	904	887	557	2000

Table 13: Optimised parameter values for the Model C at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	4	7	5	11	7
SOLVER.BASE_LR	3.38×10^{-2}	4.53×10^{-2}	4.44×10^{-2}	6.46×10^{-2}	4.66×10^{-2}
Delta Learning Rate	3.38×10^{-2}	4.53×10^{-2}	4.44×10^{-2}	6.46×10^{-2}	4.66×10^{-2}
Initial Iteration	667	289	1881	1687	104
Number of Steps	20	13	2	65	82
Iteration Step	345	116	358	491	809

Table 14: Optimised parameter values for the Model D at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	1	1	3	3	1
SOLVER.BASE_LR	3.99×10^{-1}	1.30×10^{-1}	9.69×10^{-1}	7.20×10^{-1}	1.01×10^{-1}
Delta Learning Rate	1.450×10^{-4}	1.70×10^{-4}	1.87×10^{-4}	1.56×10^{-4}	1.02×10^{-4}
Initial Iteration	5000	5000	5000	5000	5000
Number of Steps	0	0	0	0	0
Iteration Step	1000	1000	1000	1000	1000

Table 15: Optimised parameter values for the Model E at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	12	1	1	1	5
SOLVER.BASE_LR	2.14×10^{-6}	9.08×10^{-5}	4.81×10^{-3}	1.08×10^{-2}	6.83×10^{-4}
Delta Learning Rate	2.12×10^{-6}	9.07×10^{-5}	4.81×10^{-3}	1.08×10^{-2}	6.83×10^{-4}
Initial Iteration	1891	380	1306	497	140
Number of Steps	17	84	41	95	37
Iteration Step	598	181	945	781	271

Table 16: Optimised parameter values for the Model F at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	5	12	4	9	5
SOLVER.BASE_LR	2.47×10^{-2}	4.13×10^{-2}	6.17×10^{-2}	4.66×10^{-2}	6.28×10^{-2}
Delta Learning Rate	2.47×10^{-2}	4.13×10^{-2}	6.17×10^{-2}	4.65×10^{-2}	6.19×10^{-2}
Initial Iteration	1529	1440	1110	343	791
Number of Steps	81	36	6	14	81
Iteration Step	529	608	197	544	109

Table 17: Optimised parameter values for the Model G at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	7	7	4	11	4
SOLVER.BASE_LR	6.14×10^{-2}	2.14×10^{-2}	1.70×10^{-2}	5.96×10^{-2}	9.67×10^{-3}
Delta Learning Rate	6.13×10^{-2}	2.14×10^{-2}	1.67×10^{-2}	5.96×10^{-2}	9.66×10^{-3}
Initial Iteration	448	1681	1458	488	1140
Number of Steps	92	18	4	77	33
Iteration Step	925	724	268	567	665

Table 18: Optimised parameter values for the Model H at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	1	5	1	1	1
SOLVER.BASE_LR	4.23×10^{-1}	2.37×10^{-6}	6.79×10^{-1}	1.32×10^{-1}	2.83×10^{-1}
Delta Learning Rate	1.00×10^{-4}	1.04×10^{-6}	1.00×10^{-4}	1.00×10^{-4}	1.00×10^{-4}
Initial Iteration	5000	760	5000	5000	5000
Number of Steps	0	70	0	0	0
Iteration Step	1000	776	1000	1000	1000

Table 19: Optimized parameter values for the ProMIC model at each target rate.

Parameter	Target Rate (bpp)				
	0.12	0.25	0.50	0.75	1.00
SOLVER.IMS_PER_BATCH	7	5	4	2	4
SOLVER.BASE_LR	7.87×10^{-5}	2.37×10^{-6}	1.72×10^{-2}	1.85×10^{-3}	2.39×10^{-2}
Delta Learning Rate	7.43×10^{-5}	1.04×10^{-6}	1.65×10^{-2}	1.84×10^{-3}	2.36×10^{-2}
Initial Iteration	1617	760	812	1762	1956
Number of Steps	48	70	95	82	35
Iteration Step	204	776	615	378	134

C

APPENDIX C

This appendix presents the compression efficiency results for the *Kasthuri++* and *Delmic* datasets. The results are evaluated using objective quality metrics specified in the "JPEG AI Common Training and Test Conditions" document [113].

C.1 KASTHURI++ DATASET

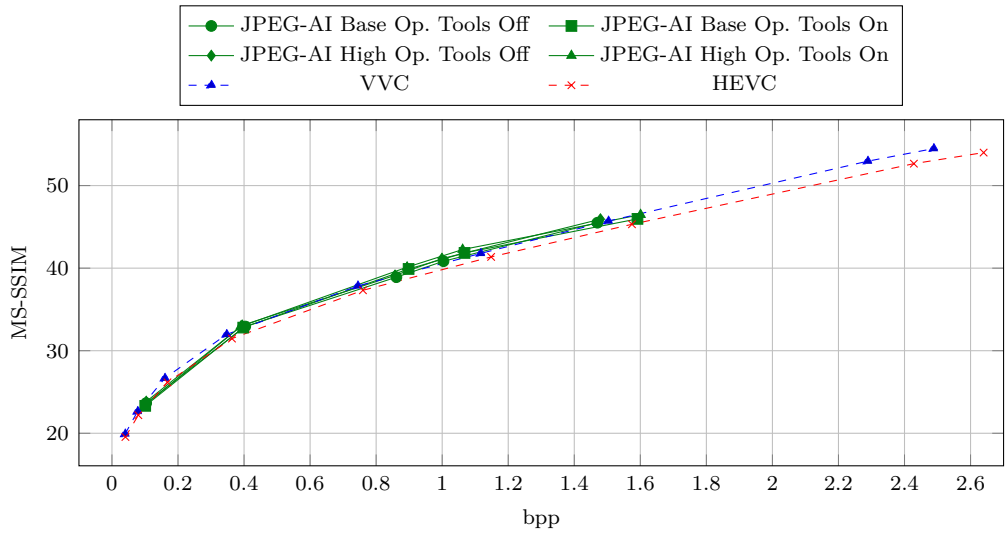


Figure 50: PSNR-HVS-M vs. bits per pixel (bpp) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

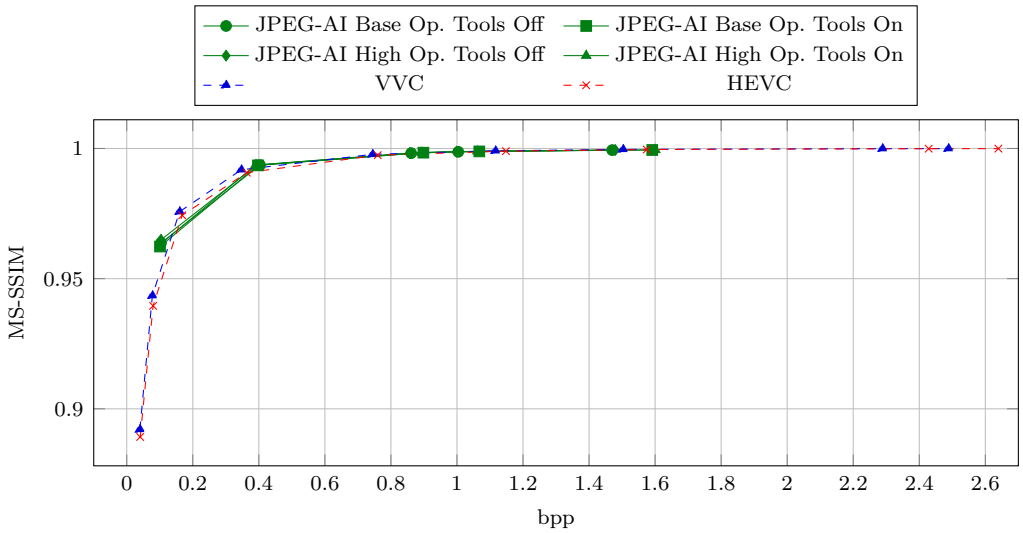


Figure 51: Multi-Scale Structural Similarity (MS-SIM) vs. bits per pixel (bpp) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

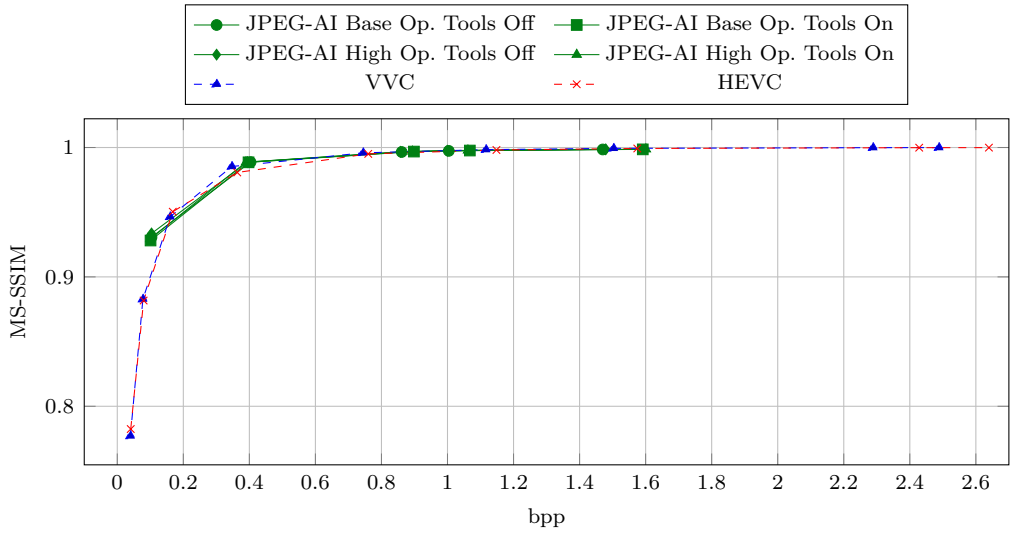


Figure 52: Information Content Weighted Structural Similarity Measure (IW-SSIM) vs. bits per pixel (bpp) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

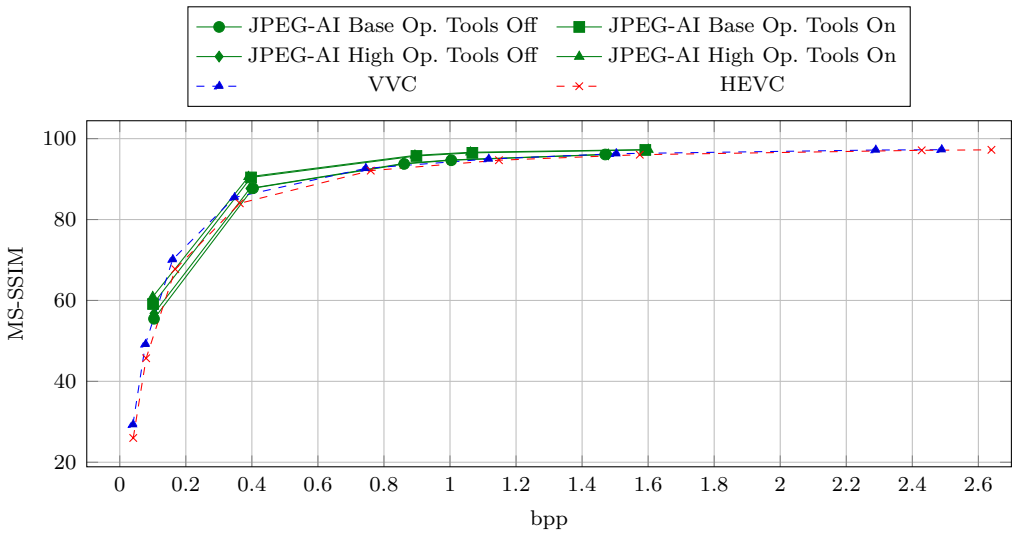


Figure 53: Video Multimethod Assessment Fusion (VMAF) vs. bits per pixel (bpp) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

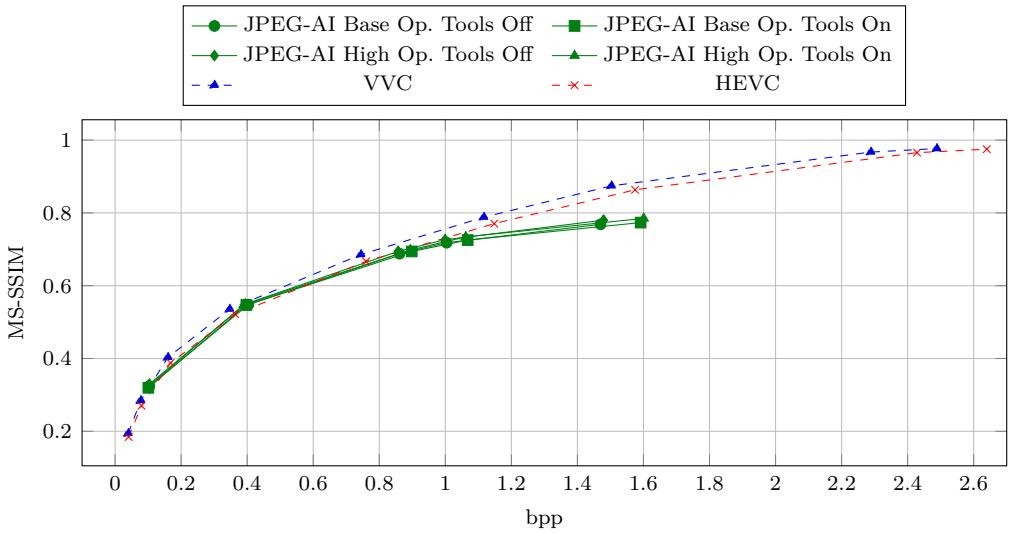


Figure 54: Visual Information Fidelity (VIF) vs. bits per pixel (bpp) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

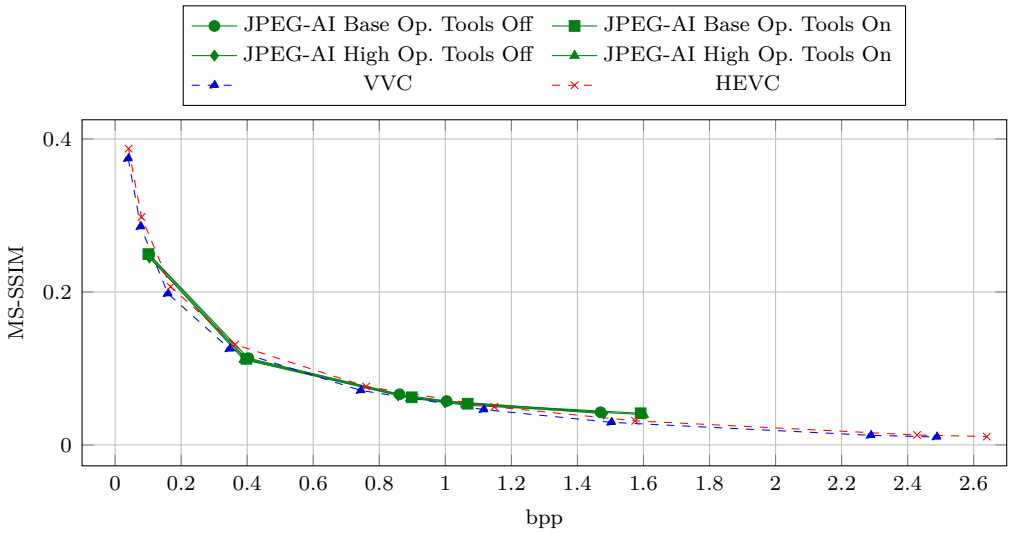


Figure 55: Normalized Laplacian Pyramid (NLPD) vs. bits per pixel (bpp) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

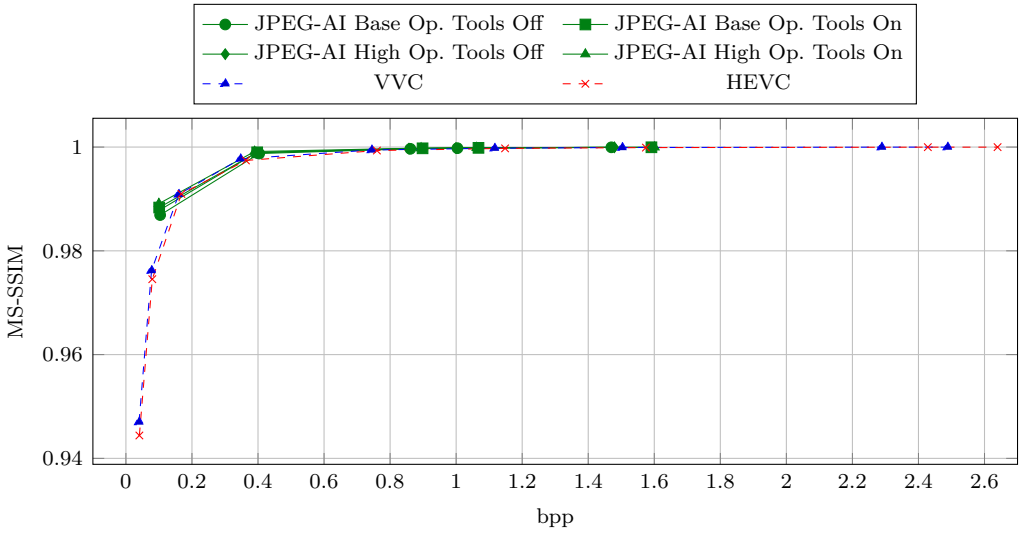


Figure 56: Feature Similarity (FSIM) for the *Kasthuri++* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

C.2 DELMIC DATASET

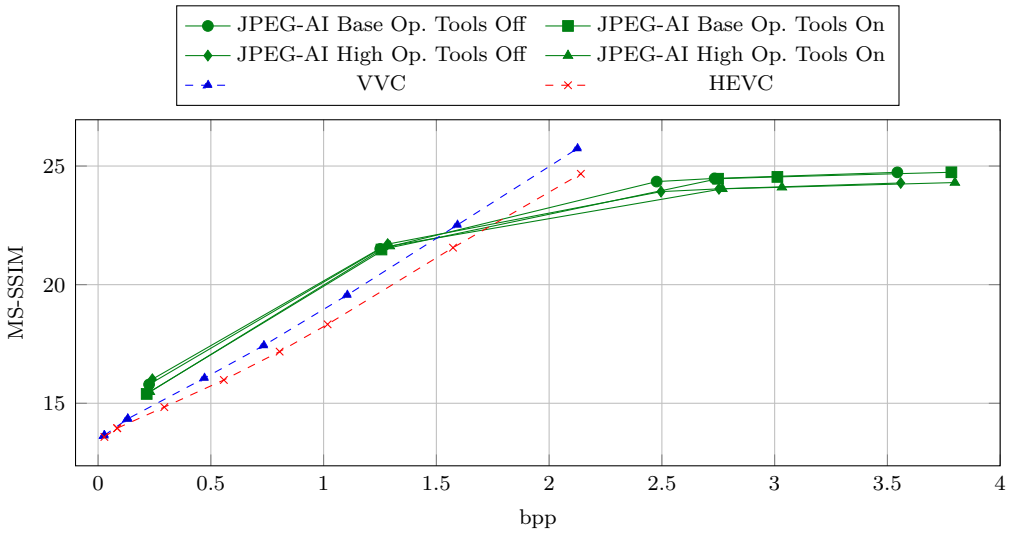


Figure 57: PSNR-HVS-M vs. bits per pixel (bpp) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

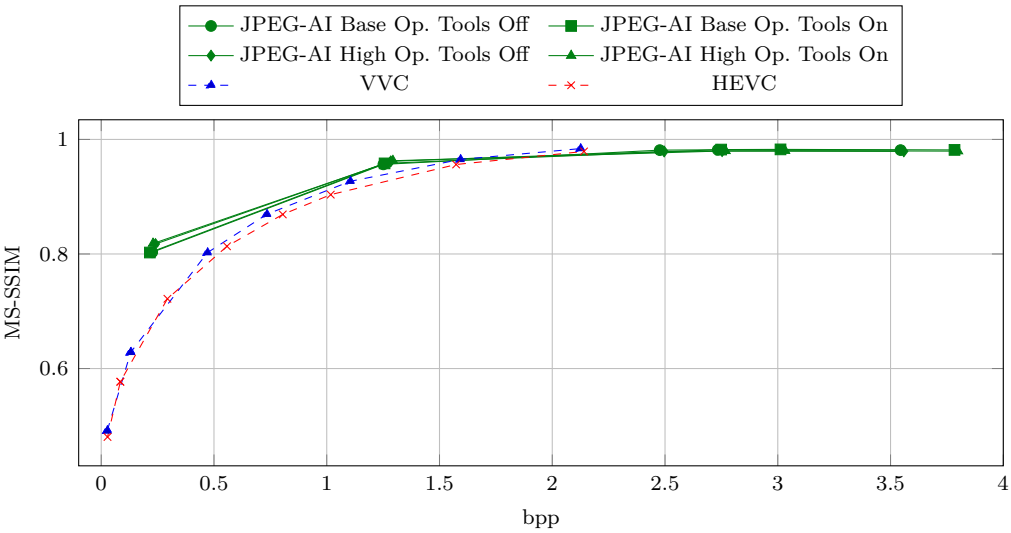


Figure 58: Multi-Scale Structural Similarity (MS-SIM) vs. bits per pixel (bpp) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

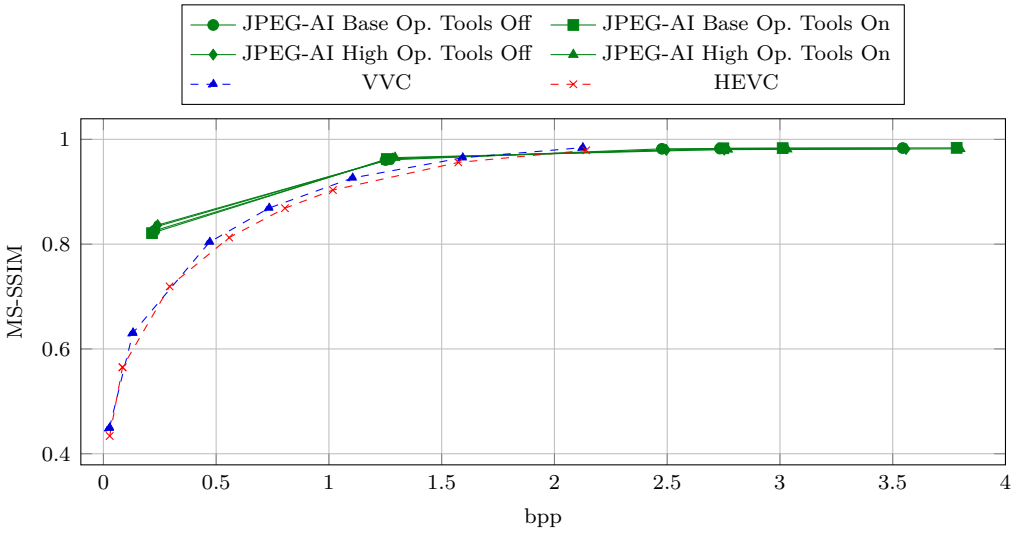


Figure 59: Information Content Weighted Structural Similarity Measure (IW-SSIM) vs. bits per pixel (bpp) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

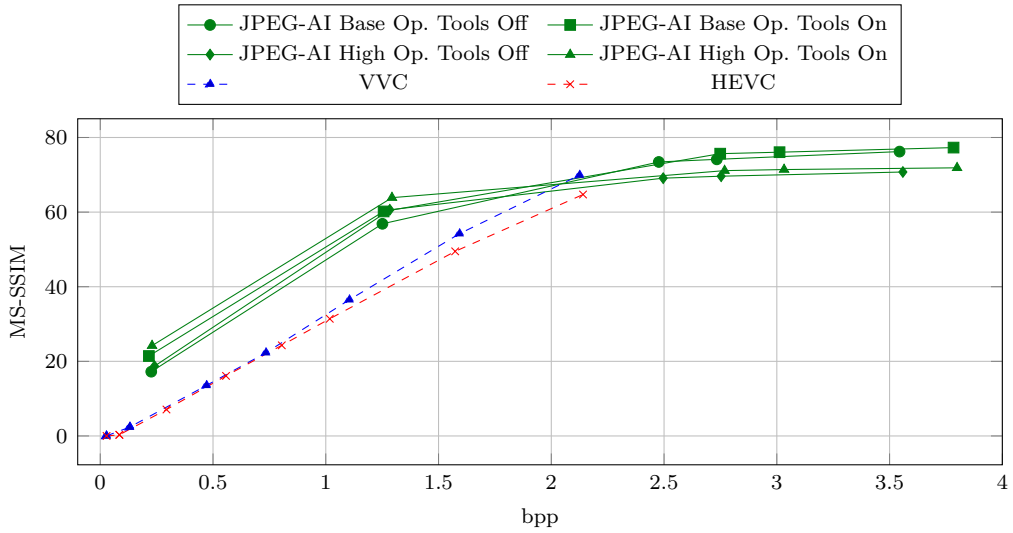


Figure 60: Video Multimethod Assessment Fusion (VMAF) vs. bits per pixel (bpp) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

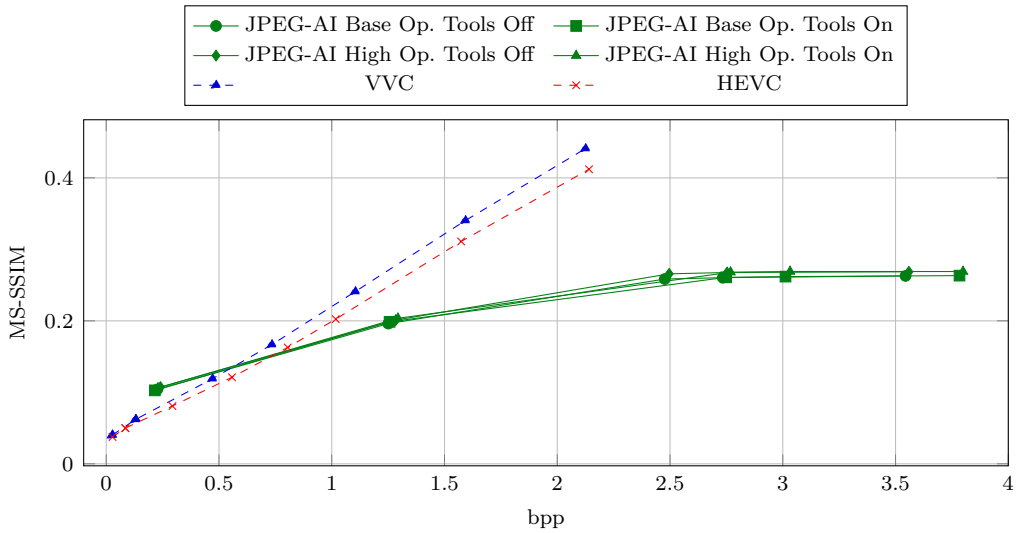


Figure 61: Visual Information Fidelity (VIF) vs. bits per pixel (bpp) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

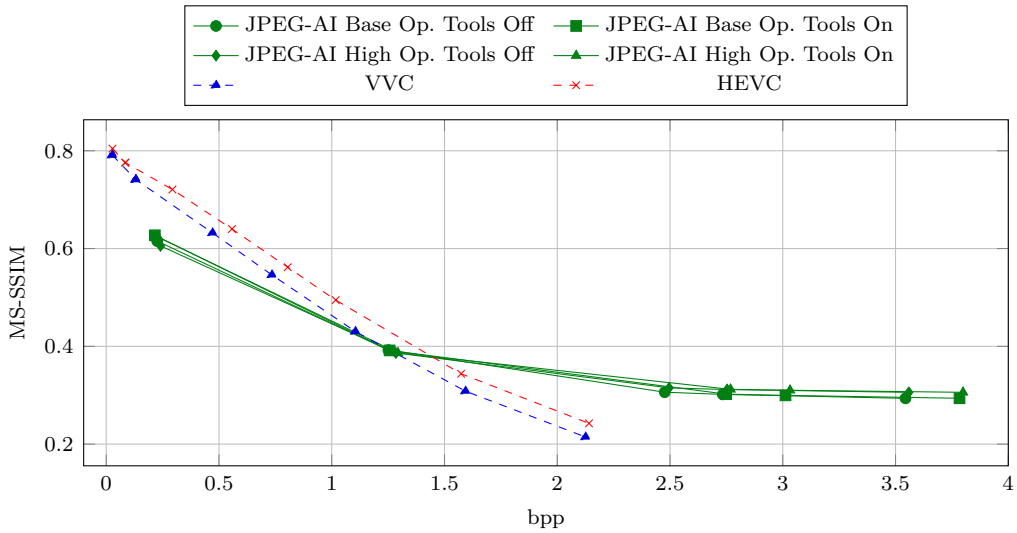


Figure 62: Normalized Laplacian Pyramid (NLPD) vs. bits per pixel (bpp) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

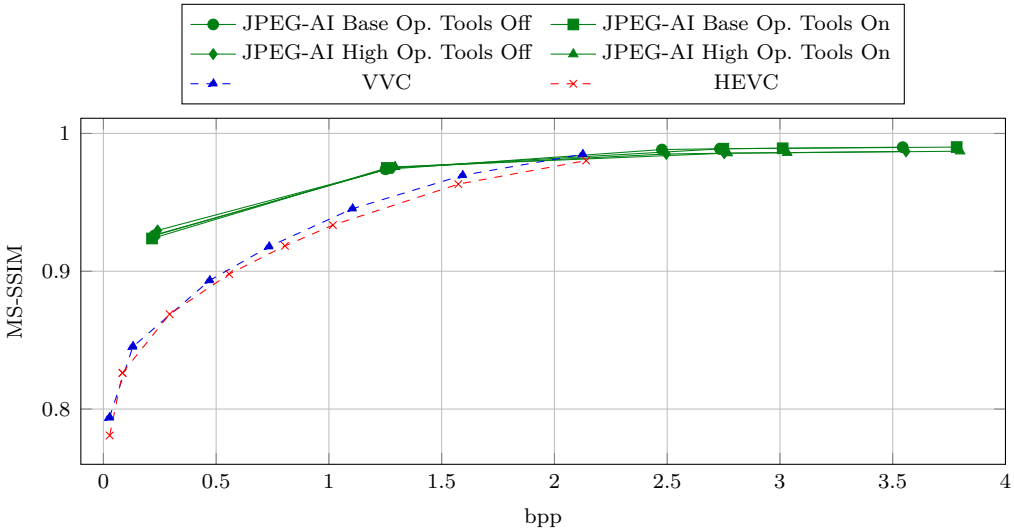


Figure 63: Feature Similarity (FSIM) for the *Delmic* dataset using different codecs. Different codecs are indicated by unique marker symbols and colours: VVC: Blue Triangle; HEVC: Red Cross; JPEG-AI Base Operation Point Tools Off: Green Circle; JPEG-AI Base Operation Point Tools On: Green Square; JPEG-AI High Operation Point Tools Off: Green Diamond; JPEG-AI High Operation Point Tools On: Green Triangle.

D

APPENDIX D

This appendix presents the ablation study results for the *Lucchi++* dataset using all object detection metrics described in Section 2.5.1.3. These metrics include AP50, AP75, APs, APm, APl, and AP.

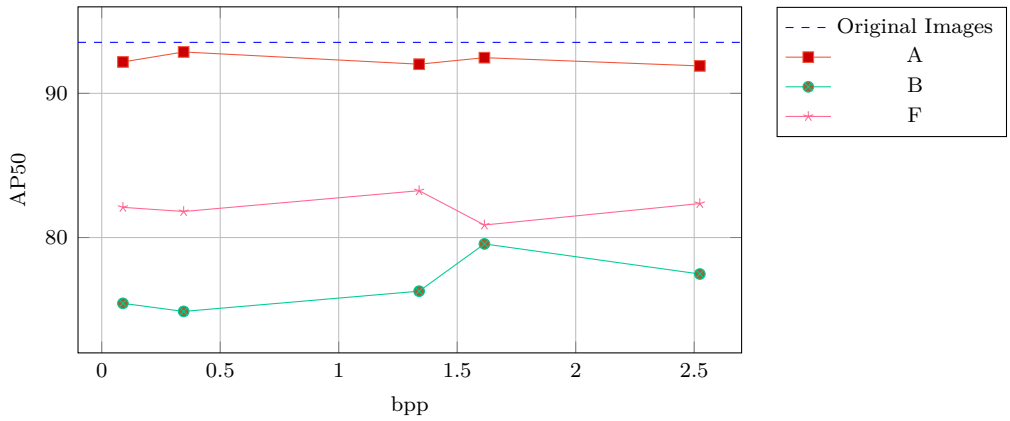


Figure 64: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

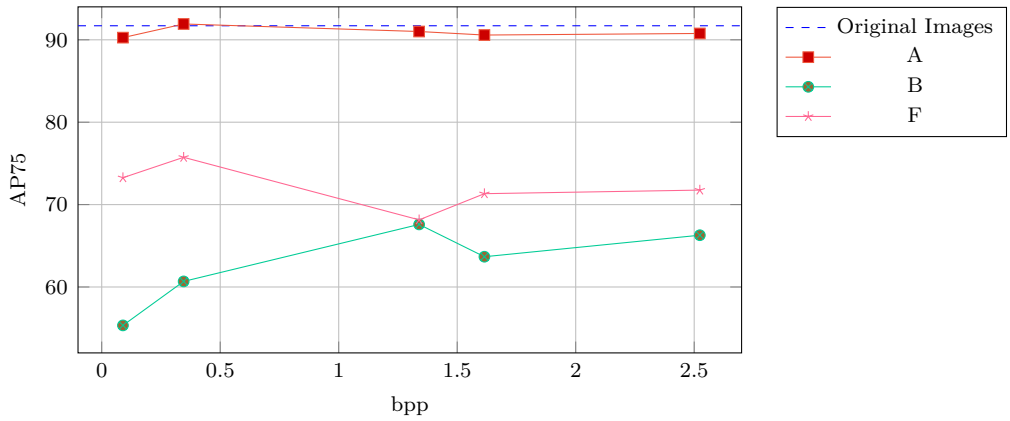


Figure 65: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

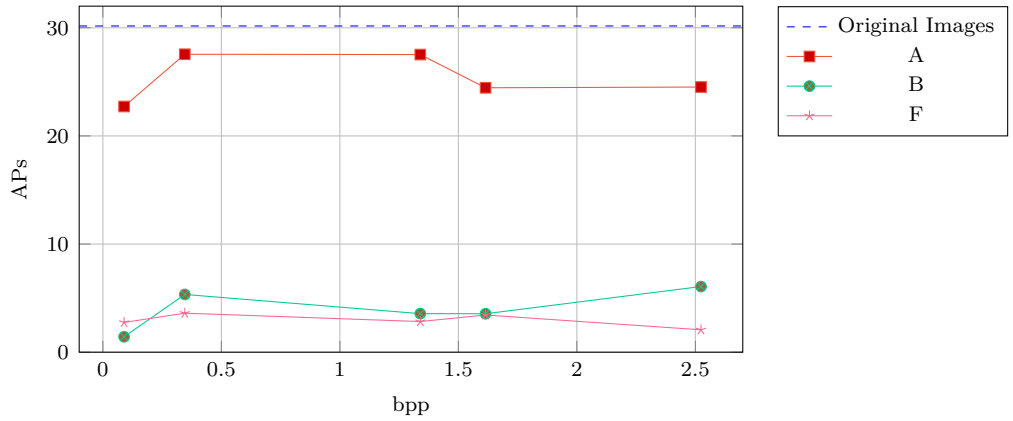


Figure 66: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

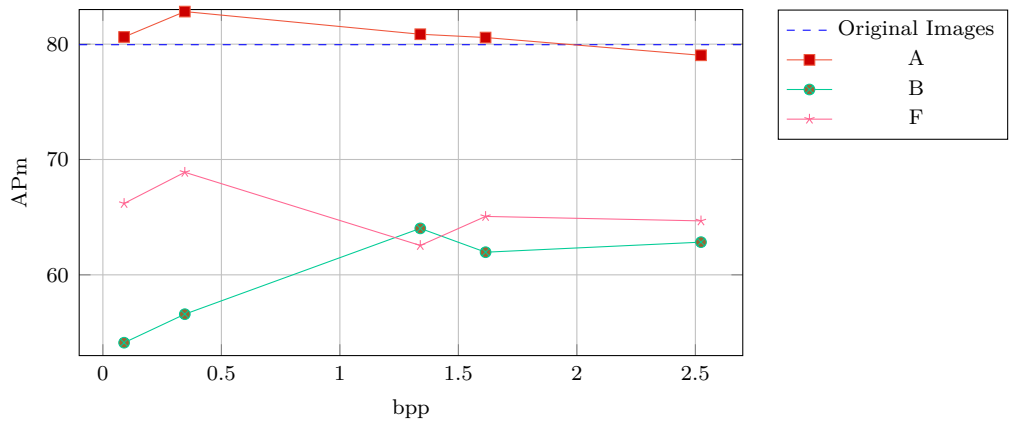


Figure 67: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

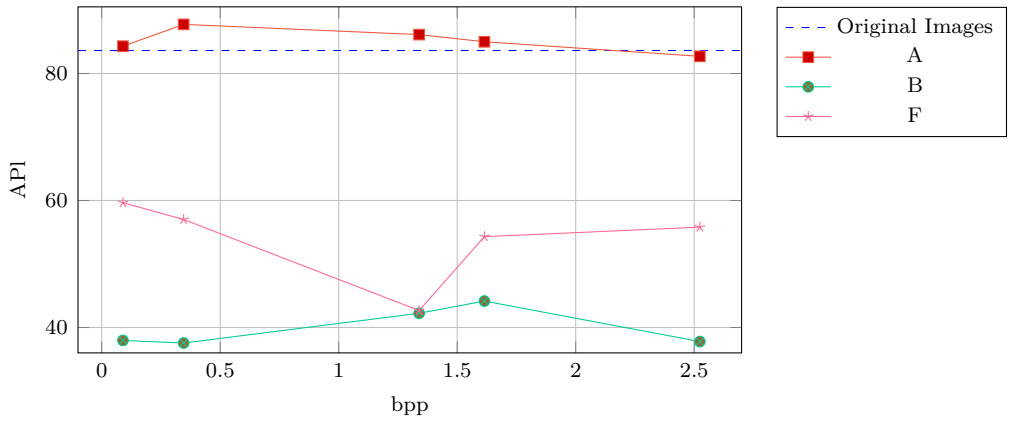


Figure 68: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

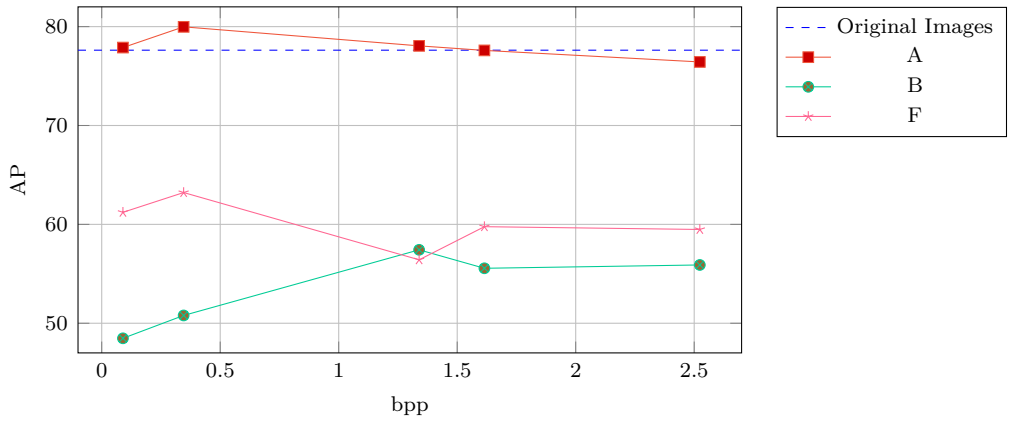


Figure 69: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture without the Lightweight Residual Block (LRB) preprocessing stage (Model B), and (4) detection using the final Domain Translator architecture with integrated LRB stage (Model F).

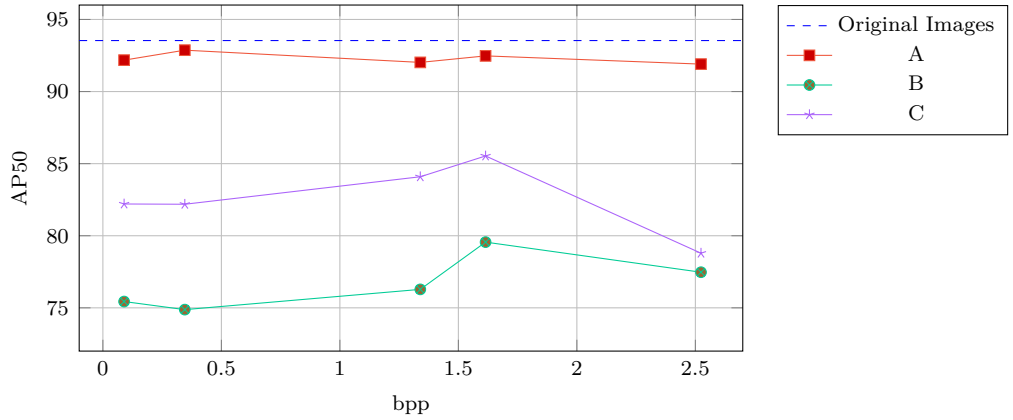


Figure 70: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

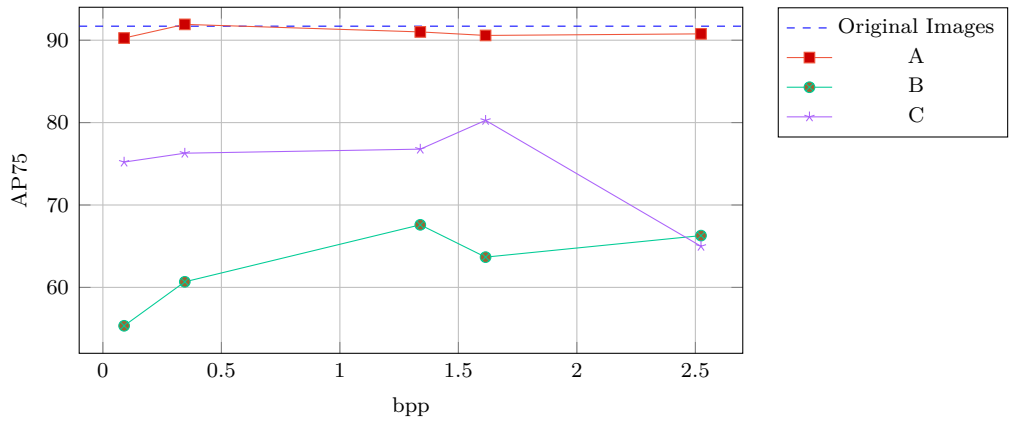


Figure 71: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

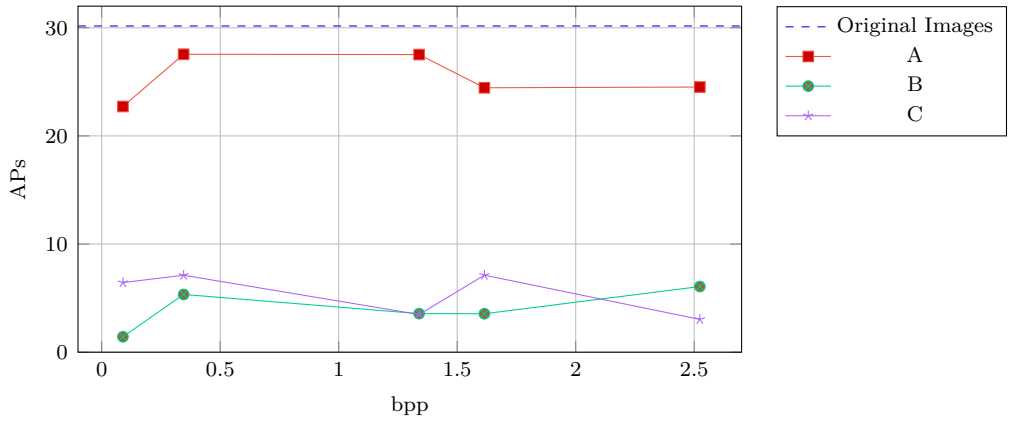


Figure 72: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

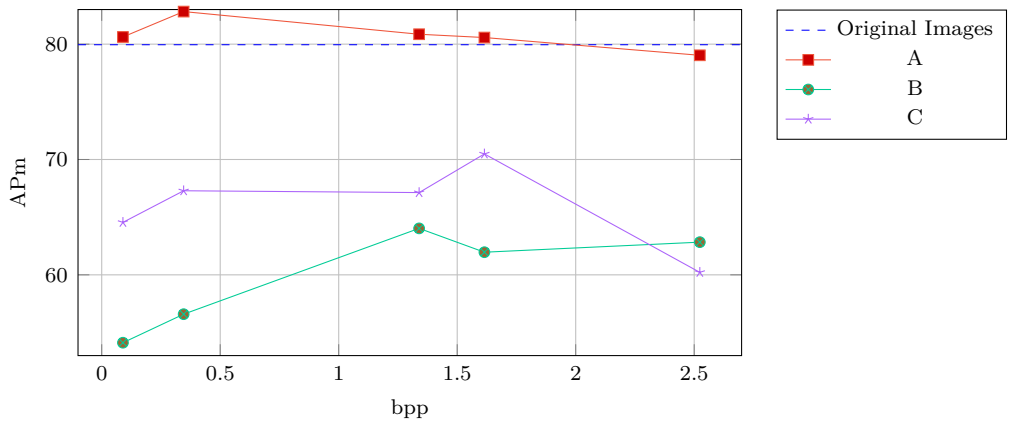


Figure 73: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

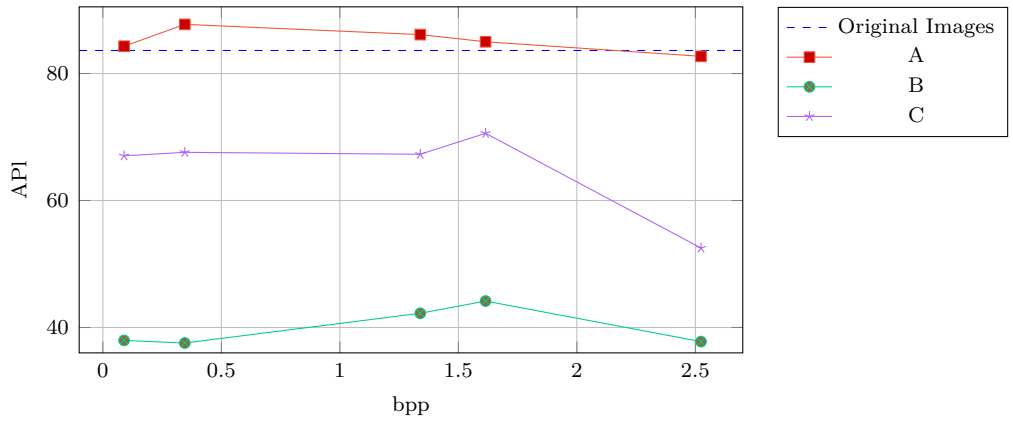


Figure 74: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APl versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

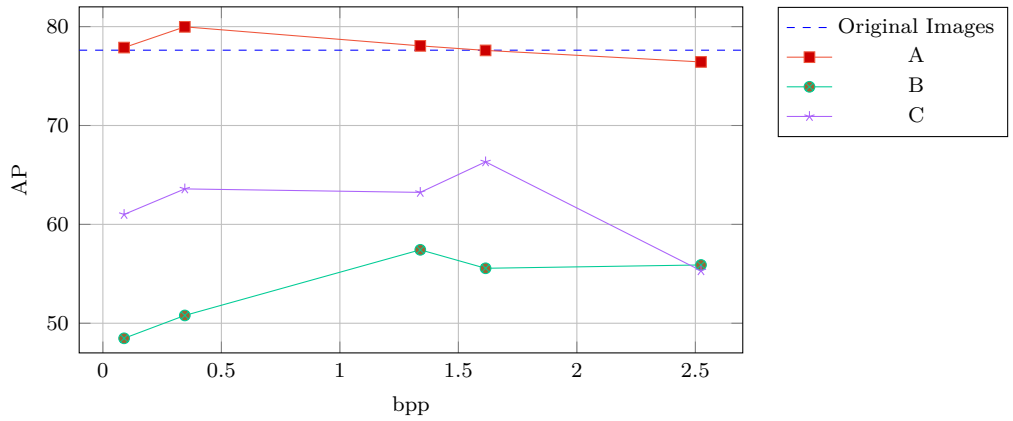


Figure 75: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model C).

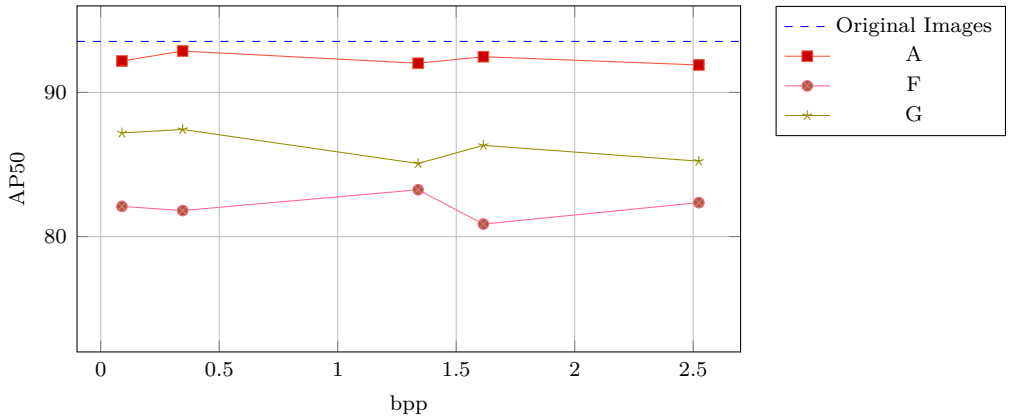


Figure 76: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final Domain Translator architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

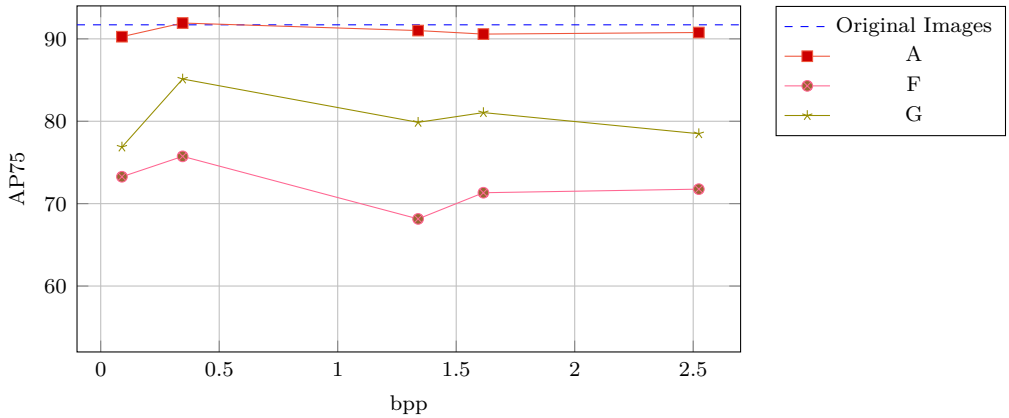


Figure 77: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final compressed domain processing architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

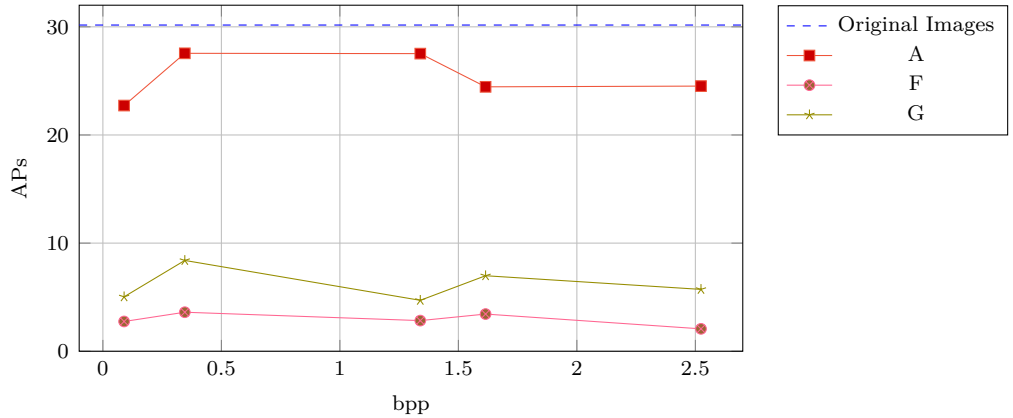


Figure 78: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final Domain Translator architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

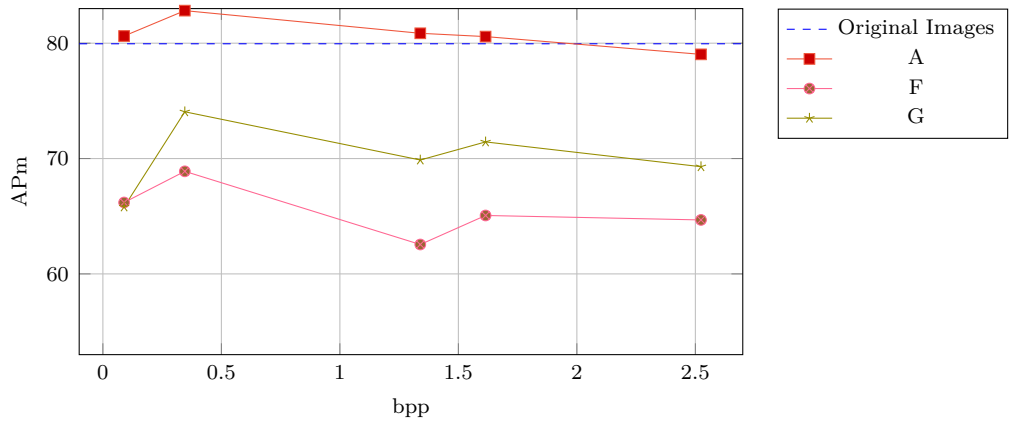


Figure 79: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final Domain Translator architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

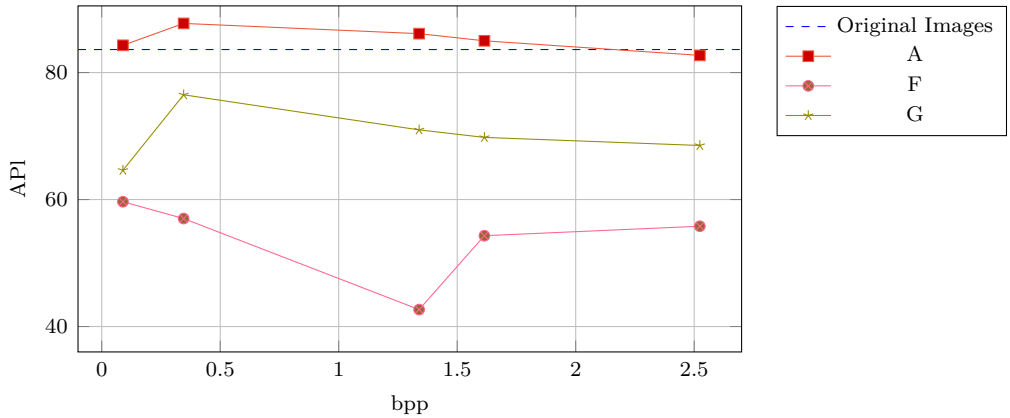


Figure 80: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APl versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final Domain Translator architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

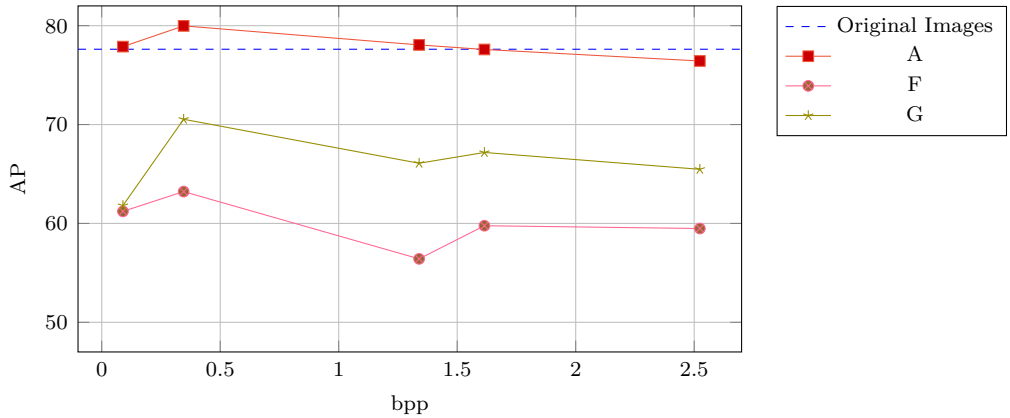


Figure 81: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using the final Domain Translator architecture with integrated LRB block and trained following the Pixel Domain Fine Tuning (PD-FT) process (Model G).

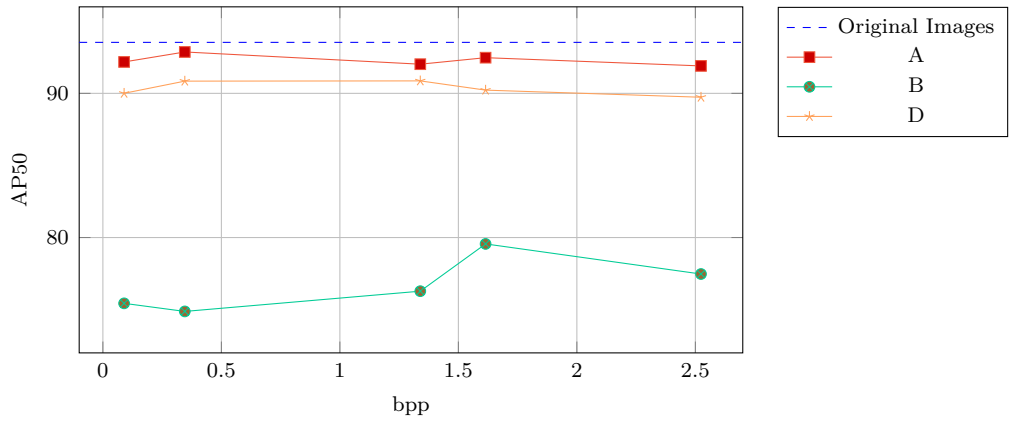


Figure 82: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

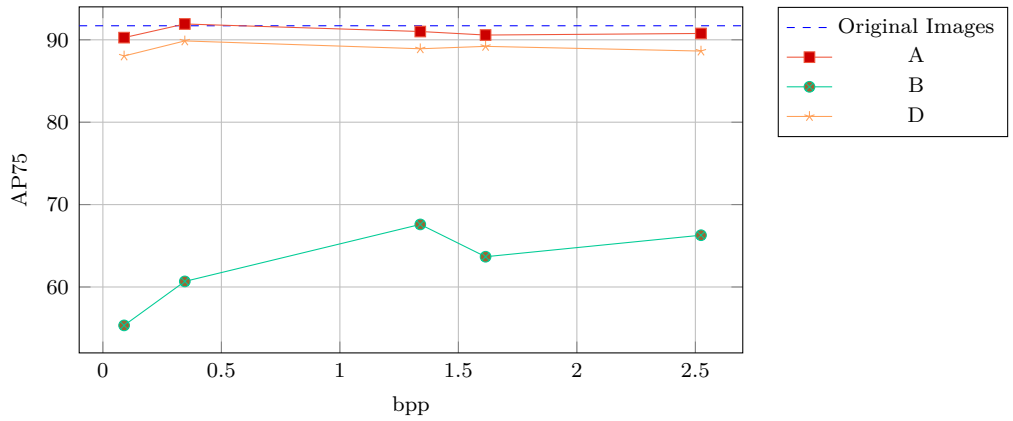


Figure 83: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

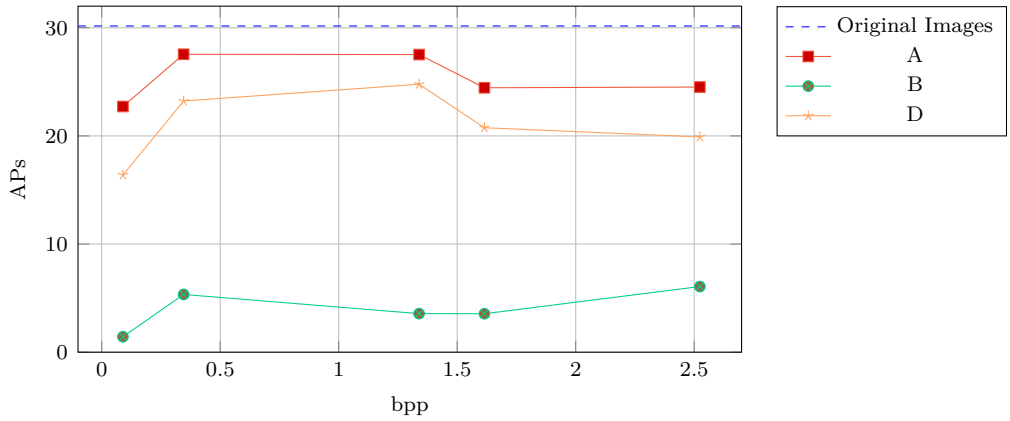


Figure 84: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

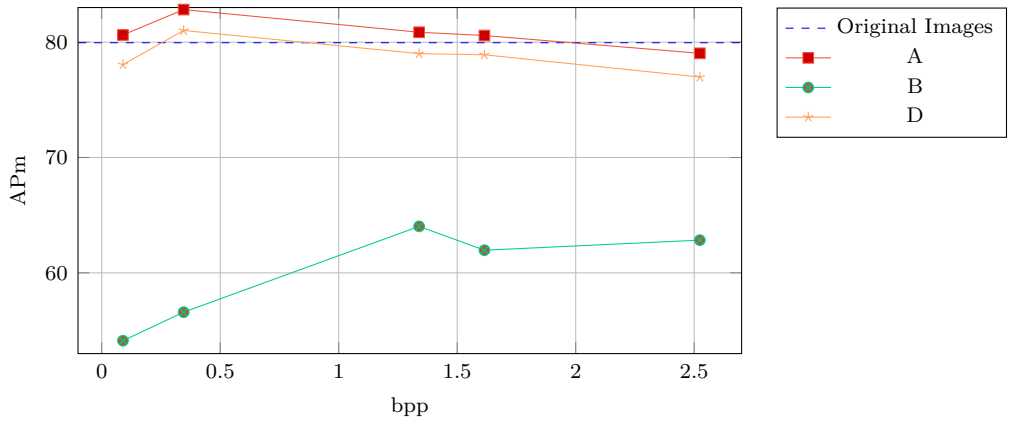


Figure 85: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

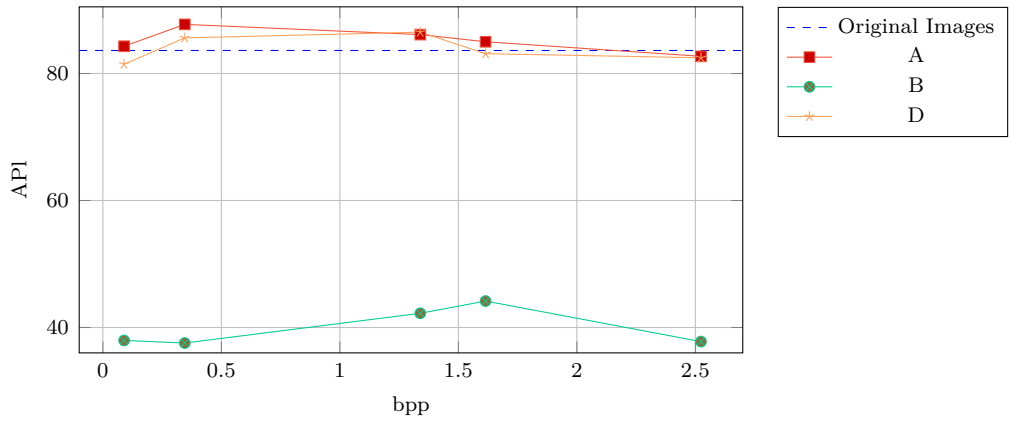


Figure 86: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APl versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

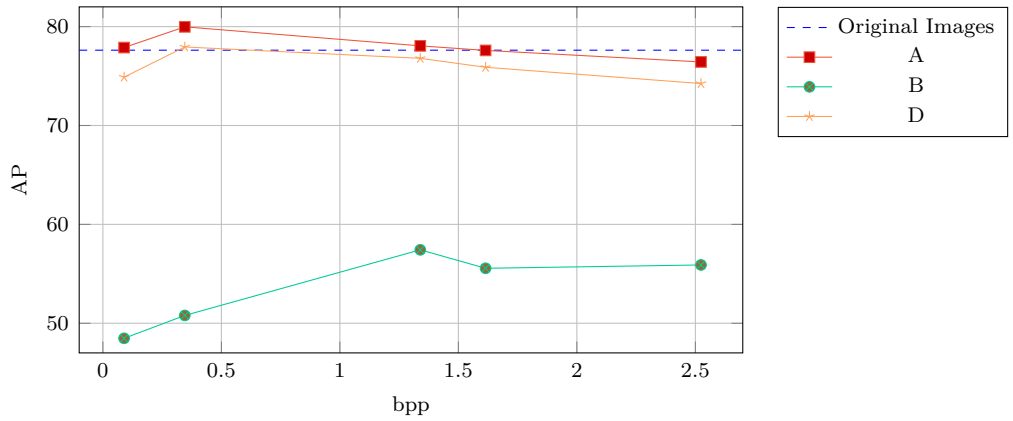


Figure 87: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the base Domain Translator architecture and full training (Model B), and (4) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D).

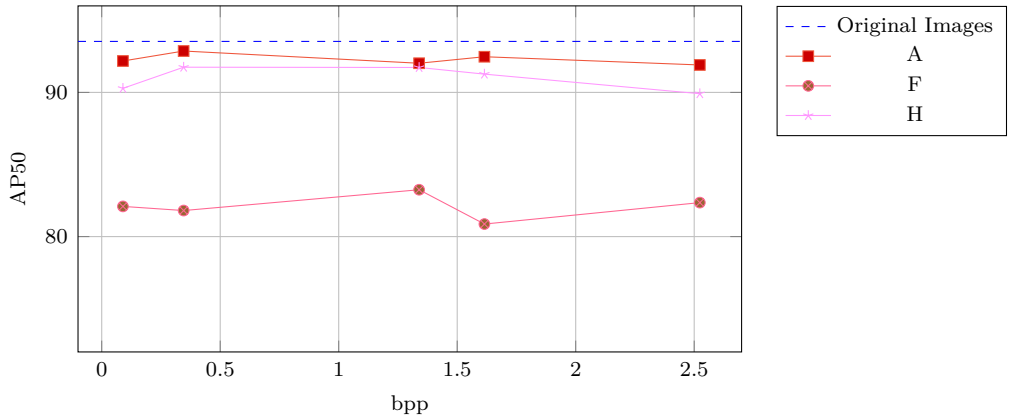


Figure 88: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

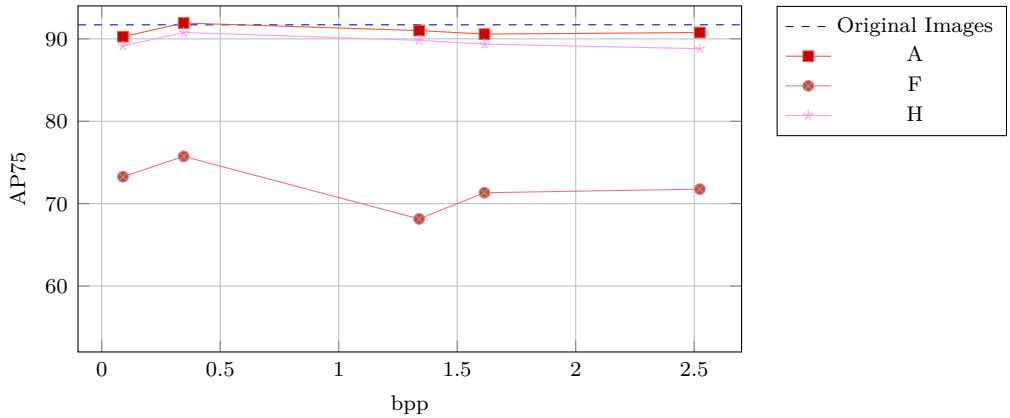


Figure 89: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

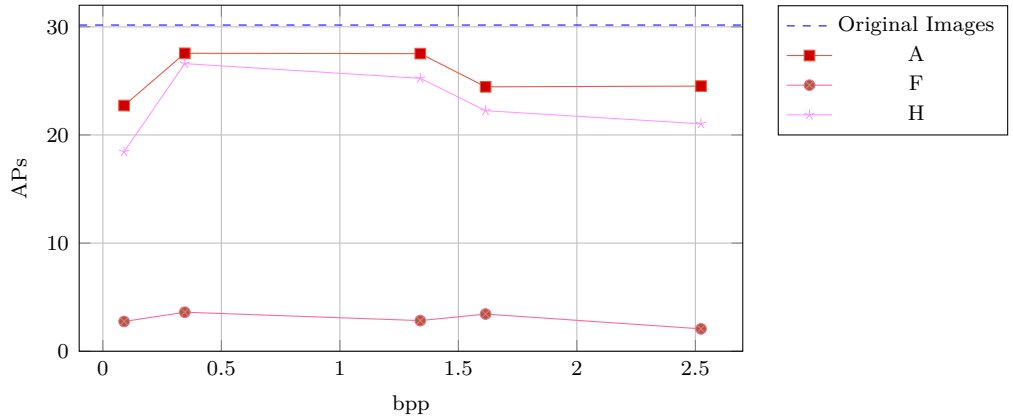


Figure 90: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

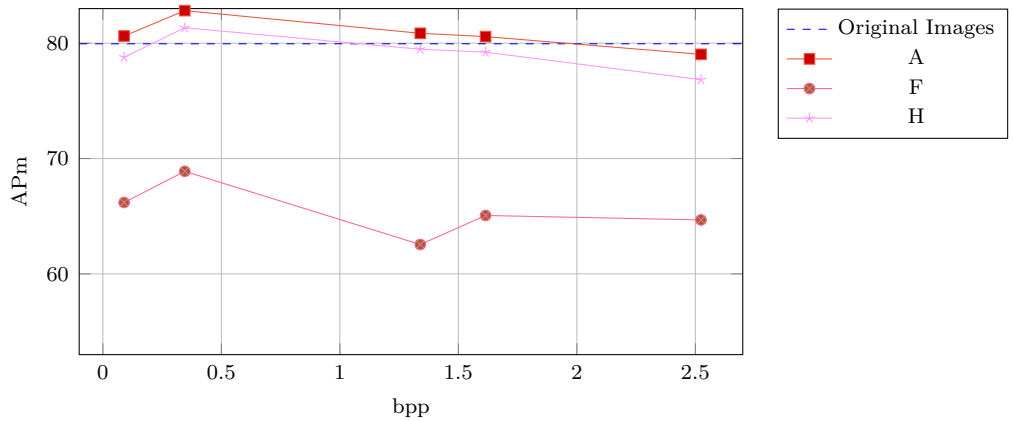


Figure 91: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

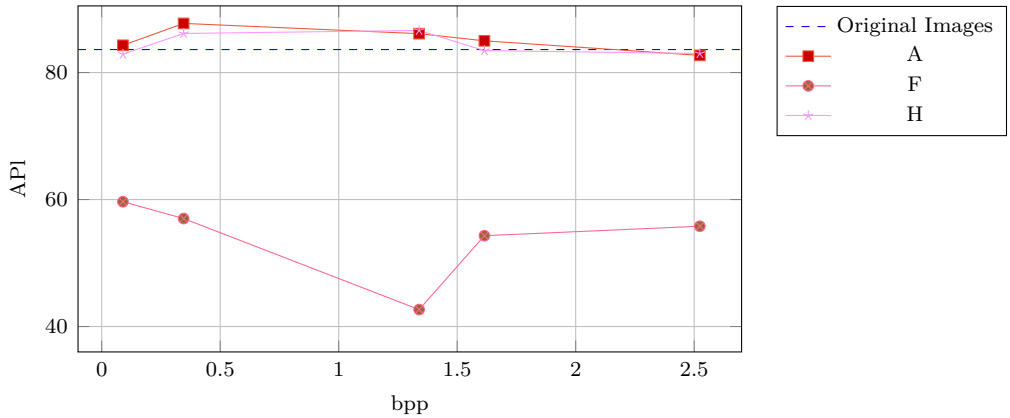


Figure 92: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APl versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

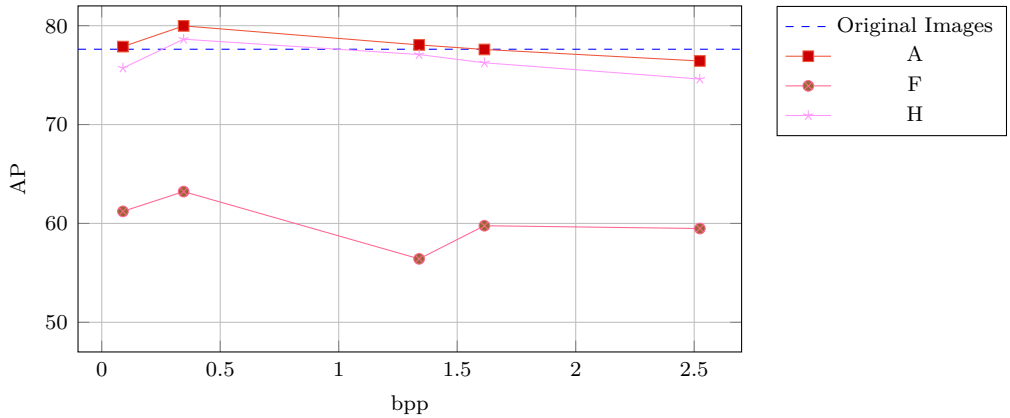


Figure 93: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using the final Domain Translator architecture with integrated LRB block and full training (Model F), and (4) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H).

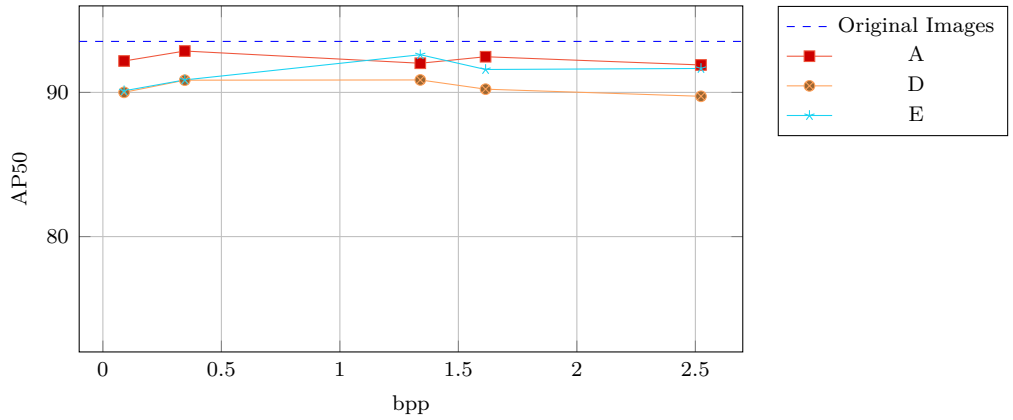


Figure 94: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

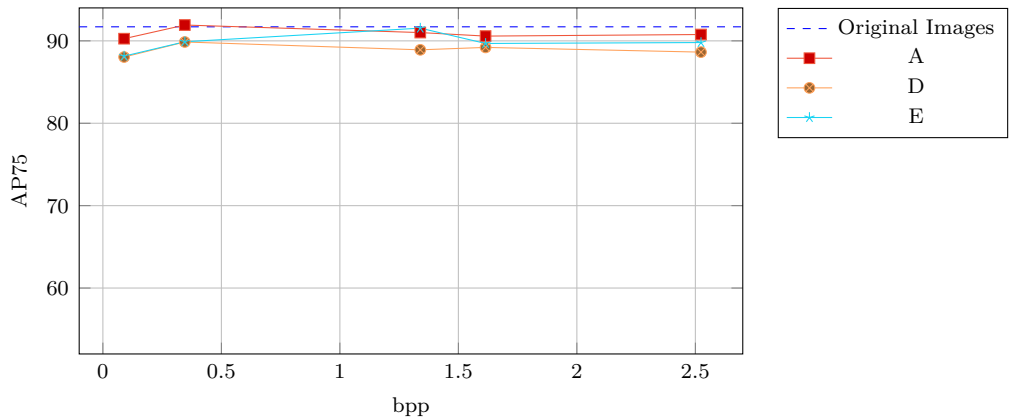


Figure 95: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

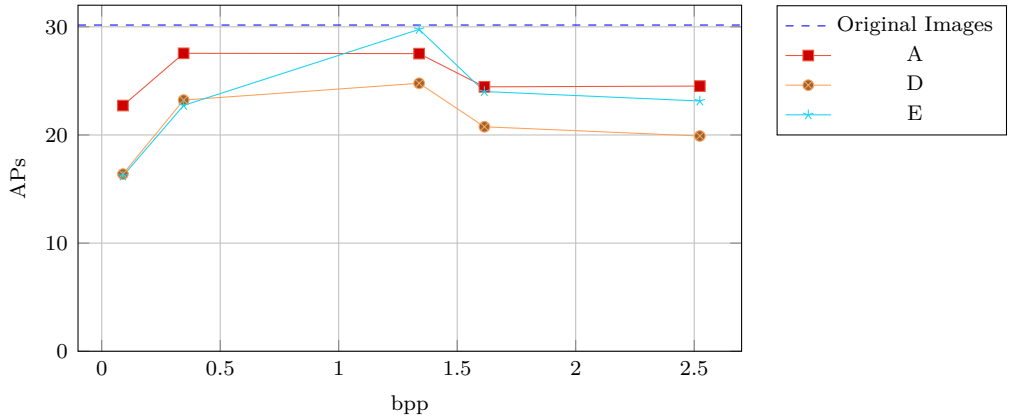


Figure 96: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

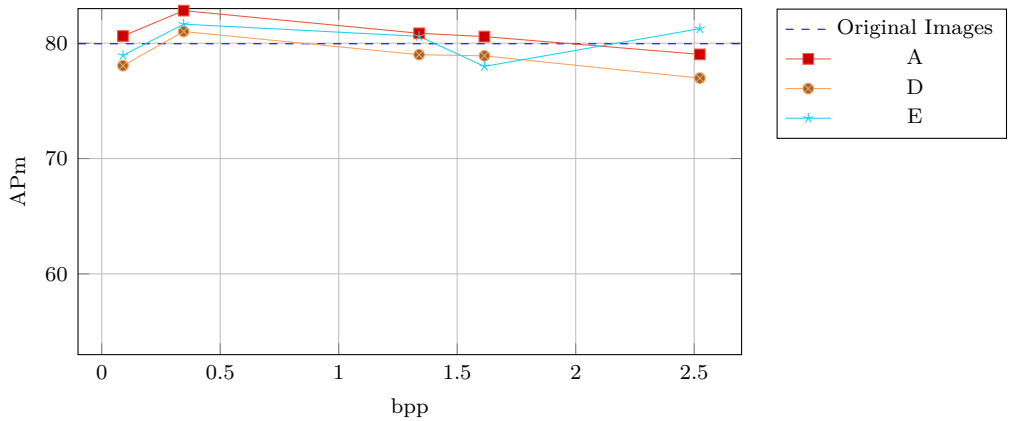


Figure 97: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

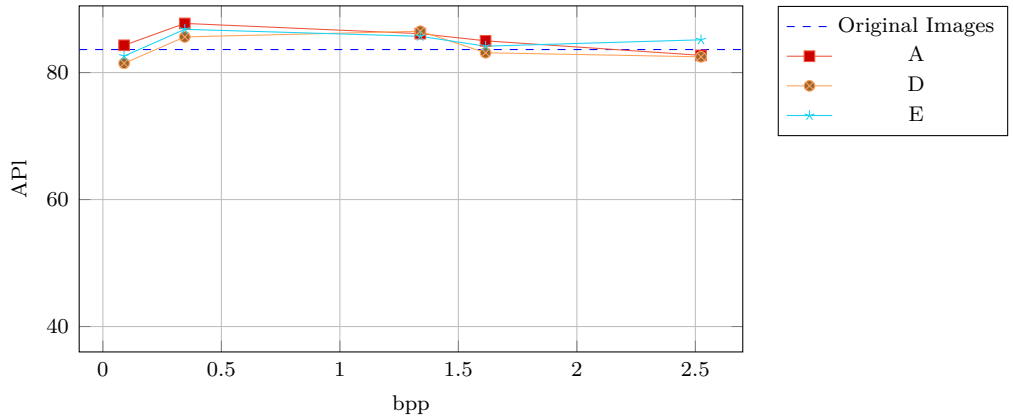


Figure 98: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APl versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

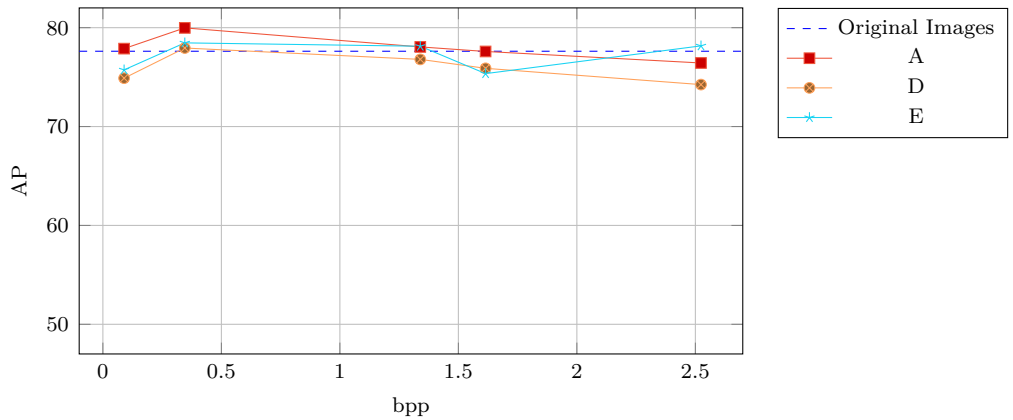


Figure 99: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using base Domain Translator architecture trained through the Guided Domain Translator Training (GDTT) process (Model D), and (4) detection using base Domain Translator architecture trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model E).

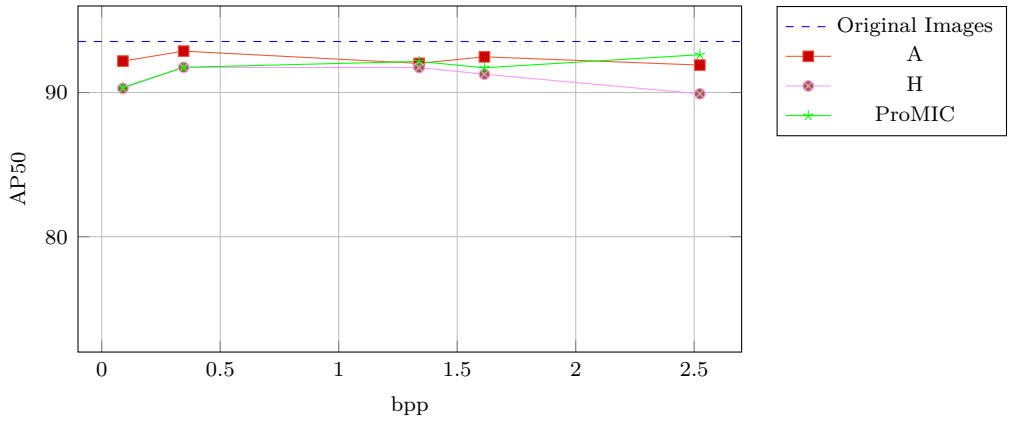


Figure 100: Performance comparison of object detection models across different image compression settings in terms of AP50. The graph shows AP50 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

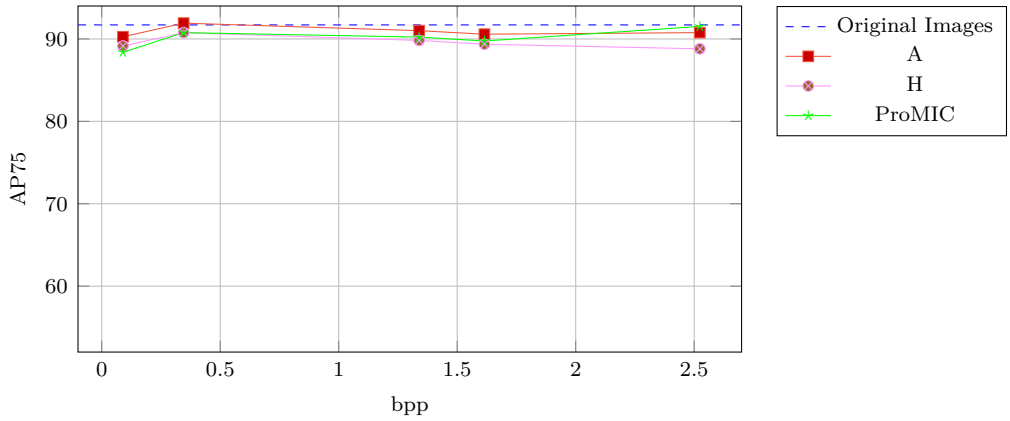


Figure 101: Performance comparison of object detection models across different image compression settings in terms of AP75. The graph shows AP75 versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

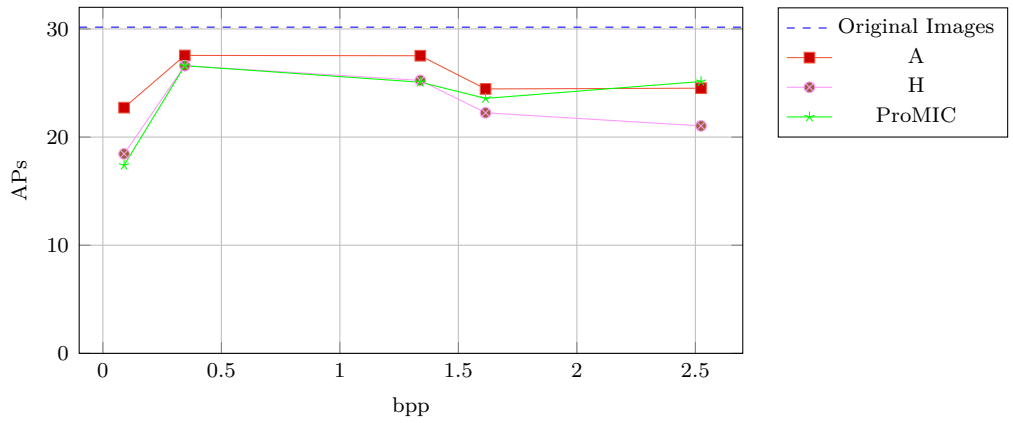


Figure 102: Performance comparison of object detection models across different image compression settings in terms of APs. The graph shows APs versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

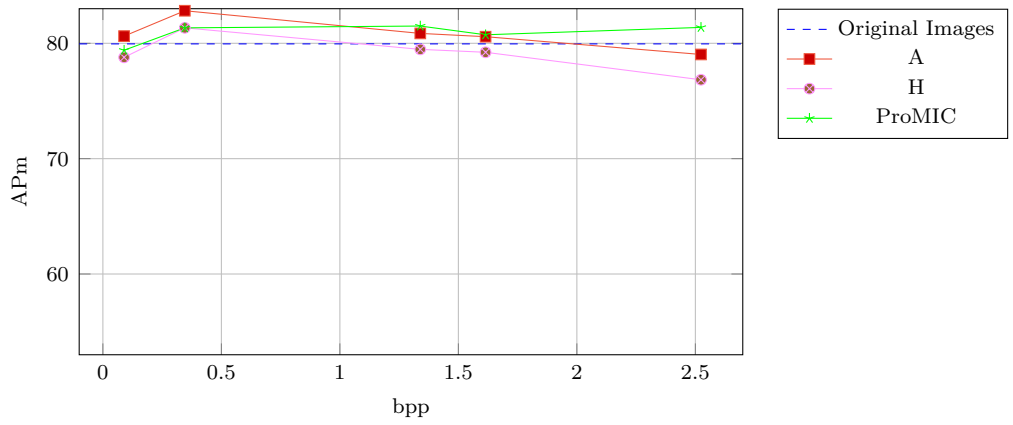


Figure 103: Performance comparison of object detection models across different image compression settings in terms of APm. The graph shows APm versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

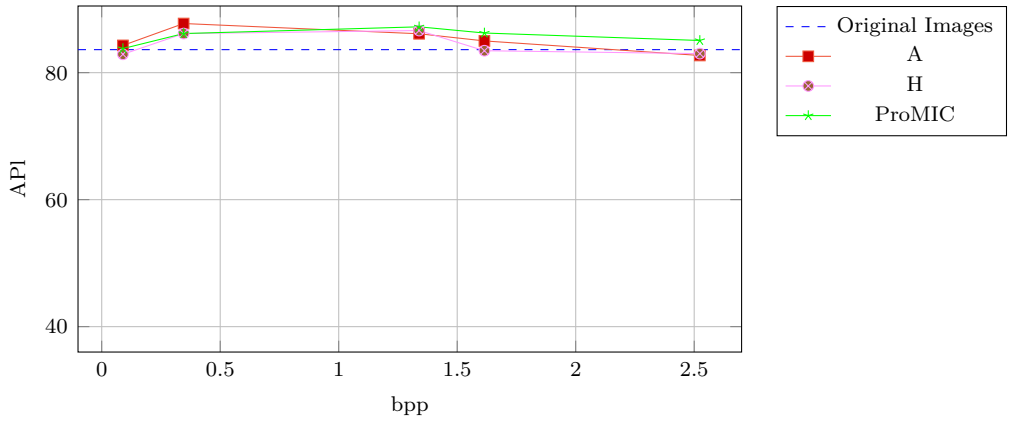


Figure 104: Performance comparison of object detection models across different image compression settings in terms of APl. The graph shows APl versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

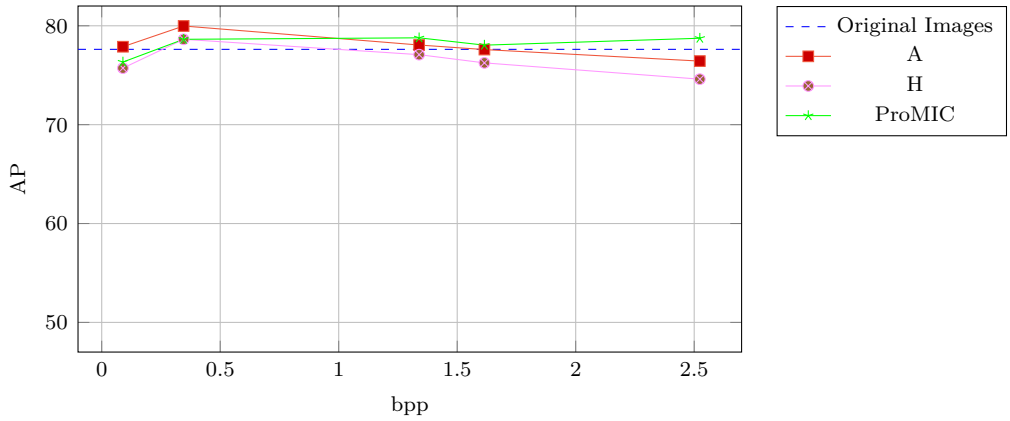


Figure 105: Performance comparison of object detection models across different image compression settings in terms of AP. The graph shows AP versus bits per pixel (bpp) for four scenarios: (1) detection on original uncompressed images (Original Images), (2) detection on reconstructed images (Model A), (3) detection using final Domain Translator architecture with integrated LRB block trained through the Guided Domain Translator Training (GDTT) process (Model H), and (4) detection using the proposed method that uses the final Domain Translator architecture with integrated LRB block trained through the GDTT process and fine tuned through the Pixel Domain Fine Tuning (PD-FT) process (Model ProMIC).

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"Processing of Microscopy Images in the Compressed Domain"*, é original e foi realizado por Nicolas Vasconcellos (2232720) sob orientação de Prof. Luís Távora, Prof. Carlos Grilo, Prof. Rolando Miragaia e Prof. Lucas Thomaz. .

Leiria, March 2025

Nicolas Vasconcellos