UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN "MATEMATICA"

CICLO XXXVI

Transformative Approaches for Deep Learning in Resource-Constrained Scenarios

Coordinatore: Prof. Leonardo Biliotti

Tutore: Prof. Marko Bertogna

Correlatore: Prof. Giorgia Franchini

Dottorando: Carmelo Scribano

Anni Accademici: 2020/2021 – 2022/2023

Abstract

Latest advancements in Artificial Intelligence, and in particular in Deep Learning, have catalyzed groundbreaking progress across diverse applications such as Computer Vision, Natural Language Processing, and content generation. However, the significant computational demands inherent in training and executing powerful Deep Learning models have hindered widespread adoption of these techniques in certain application contests. One area poised to benefit greatly from Deep Learning, especially applied to computer vision, is the automotive sector, particularly in the development of driver assistance systems. In this context, minimizing inference costs is a priority in order to enable deployment on the low-power embedded devices found in vehicles. Conversely, the costs and complexities associated with training phase are sometimes substantial, exemplified by recent transformer-based models for natural language processing and image synthesis models utilizing the Denoising Diffusion Probabilistic paradigm. This thesis addresses two primary objectives: (i) proposing low-computational-cost solutions for computer vision applications in automotive settings and (ii) presenting innovative approaches to formulating efficient Deep Learning models using lossy compression techniques. To achieve the former goal, this study develops two models for Driver Monitoring Systems and Advanced Driving Assistance Systems, employing a Multi-Task Learning approach. This choice enable significant computational savings by sharing a substantial portion of the neural architecture across different tasks. For the latter objective, this thesis introduces an approximation of the transformer attention layer leveraging the Discrete Cosine Transform. Additionally, it proposes a strategy for incorporating Vector Quantization-based compression techniques into the image generation process using Diffusion Models. Through experimental analyses and quantitative evaluations, this thesis demonstrates the effectiveness of the proposed methods in reducing the complexity and computational costs in the concerned contexts.

Contents

List of Figures

List of Tables

List of Abbreviations

ADAS Advanced Driving Assistance System. 3, 42–45

BN Batch Normalization. 29

BSD Blind Spot Detection. 42

CNN Convolutional Neural Network. 8, 11, 12, 32, 48, 50, 81

DCT Discrete Cosine Transform. 4, 71, 72, 77, 81, 89

DDPM Denoising Diffusion Probabilistic Model. 3, 93, 94, 105, 110

DL Deep Learning. 11, 75

DMS Driver Monitoring System. 3, 6, 7, 9, 18

FCW Forward Collision Warning. 42

GAN Generative Adversarial Networks. 93

GPU Graphics Processing Units. 1, 39

GRU Gated Recurrent Unit. 73

IoU Intersection over Union. 64, 65

LDM Latent Diffusion Model. 94

LSTM Long Short-Term Memory. 73

MAC Multiply-Accumulate. 40, 67

MTL Multi-Task Learning. 3, 8, 10, 15

NLP Natural Language Processing. 4, 71, 87

OPQ Optimized Product Quantization. 102, 107

PQ Product Quantization. 101, 102, 107, 108

RNN Recursive neural Network. 71, 73

RoI Region of Interest. 20

RQ Residual Quantization. 103, 107, 108

SGD Stochastic Gradient Descent. 28

SoC System on Chip. 40

SWA Stochastic Weight Averaging. 29

TOPs Tera Operation per Seconds. 40, 46, 68

VAE Variational Autoencoders. 98, 108

VQ Vector Quantization. 94, 98, 102–108, 111

Chapter 1

Introduction

1.1 Motivations

The widespread use of so-called artificial intelligence technologies, a term that currently identifies deep learning methods with abuse of notation, is spreading rapidly to the point of becoming ubiquitous. The field of study of these techniques is growing and expanding at breakneck speed, as are the technological and human resources devoted to the ever more frenetic pursuit of performance improvement. Performance in this case is almost universally identified as the ability of developed algorithms to solve tasks, with the aim of matching or increasingly exceeding human capabilities. One universally recognized issue concerns the computational costs and consequently the hardware resources required for training and inference of modern, high-performance deep learning paradigms. For several years now, these issues have been addressed from several fronts:

1. Development of specialized techniques in reducing computational cost and memory consumption to be applied to existing models. These include quantization techniques, which is the process of reducing the precision of numerical values in a neural network, typically from 32-bit floating-point numbers to lower precision formats such as 8-bit integers. Pruning on the other side is the process of removing redundant or less important connections (weights or neurons) from a neural network, resulting in a sparser network architecture.

2. Another approach is to leverage hardware accelerators tailored for deep learning workloads. These accelerators, typically include parallel accelerators like Graphics Processing Units (GPU) or specialized neural processing units (NPUs), designed to perform matrix operations that underlie deep learning loads. By offloading computation to these accelerators, the overall system throughput can be significantly improved.

3. In addition, software-level optimizations can also play a crucial role in improving the efficiency of deep learning inference. Techniques such as kernel fusion, loop unrolling, and parallelization can help maximize the utilization of the available computational resources.

Orthogonally to these development directions, which are aimed at the wide-ranging optimization of the entire deep learning landscape, one possible direction of work is to develop techniques and architectures tailored to the desired

outcome. Driven by an in-depth knowledge of the application domain and the intrinsic characteristics of the methodologies exploited, a search for new paradigms for the simplification and optimization of different deep model families is therefore proposed. The work of this thesis can be divided into two main parts: the first is the development of computer vision models with a reduced computational footprint, aimed at use in an automotive context. The second part, on the other hand, focuses on the use of techniques borrowed from the domain of lossy data compression in order to overcome the inherent inefficiencies in defining specific deep learning methods. In brief, this thesis introduces cross-cutting approaches at various levels that can reduce the computational cost and complexity of different families of deep learning methods.

1.2 Scope

The first target domain is the automotive industry, specifically driver and driving assistance systems that use cameras and computer vision algorithms. This application area is emblematic of the application of deep learning models in which substantial limitations in the computational platform for inference coexist with stringent latency constraints and the demand for excellent performance on the implemented computer vision tasks. In this application field, delegating the computation of the inference part to a remote server equipped with GPUs is an inadvisable solution for multiple reasons:

1. The risks related to the protection of privacy are increasingly serious; by transmitting images taken inside or outside the vehicle, the risk of incurring privacy violations is obvious.

2. The latency introduced by the transmission of data and the reception of results does not allow the use of these methodologies in safety-critical contexts where it is necessary to comply with hard-real-time criteria.

3. Data coverage may also not be guaranteed throughout the territory.

4. In addition, the costs of operating the infrastructure have to be covered, which typically involves the payment of a subscription by the vehicle owner in order to be able to use the safety devices. This leads to an unacceptable situation.

Consequently, it turns out to be essential that inference is carried by the computing units in the vehicle, even in the absence of a network connection. In particular, the embedded platforms that can be used in-vehicle for inference generally belong to the System on Chip family (SoC), with limited computational resources compared to high-end GPUs or CPUs commonly used in data centers or desktop applications. In addition, these platforms must meet stringent constraints in terms of energy consumption and efficiency, and of course they must be available at a relatively low price.

The second domain of interest, on the other hand, that does not look at inference on embedded devices, concerns two fundamental architectural paradigms in the deep learning domain: Transformers and Denoising Diffusion Probabilistic Model (DDPM). The former are a family of models for processing sequences, which are, among other things, behind recent successes in the field of natural language modeling, with LLMs being the prominent embodiment. DDPM, on the other hand, represent the state of the art in the field of generative models, a term used to identify techniques for generating synthetic images by means of neural networks. These methodologies, while unrelated, harbor inefficiencies that make them particularly difficult and time-consuming to train. Regarding transformers, the best known limitation is identified in the growth of memory consumption proportional to the square of the input length. This is due to the formulation of the Attention layer, which is the basis of this type of models for sequence processing. For Diffusion Models, on the other hand, the difficulty in training the generative model lies in learning the high-level semantic structure of natural images. This is to be found in the fact that the information content of an image is scattered and not uniform along the pixels of the image.
In this thesis, two original methodologies are proposed to mitigate the mentioned problems, notably in both cases the developed methodologies share the commonality of being derived from the domain of data compression.

1.3 Methodologies

The first part of this thesis, devoted to the development of computationally efficient deep models for automotive applications, is focused on the potential of Multi-Task Learning (MTL)-based architectures. This term refers to the training of models capable of solving multiple tasks in parallel for the same input, sharing a significant portion of the computation required for feature extraction. In an automotive context, it is normal for perception systems to be implemented as multiple subtasks, which would normally require the execution of a trained neural network for each. Consequently, the ability to combine them under a single architecture is an essential capability to be able to perform inference with the available resources. Specifically this first part of the thesis is divided into two chapters, the first (Chapter 2) discusses the development of a model for the Driver Monitoring System (DMS). The second chapter of the first part (Chapter 3), on the other hand, details the development of the counterpart, a perception model for Advanced Driving Assistance System (ADAS). In both cases, the Mullti-task approach proves crucial, for the DMS it is necessary, from the image of the driver's face, to simultaneously infer several parameters indicative of the driver's alertness state and in addition detect deleterious actions (e.g., cell phone use). For ADAS, on the other hand, it is required to decipher the environment surrounding the vehicle by analyzing the image of a frontal camera. This includes detection of other road users, lane edges, and classification of driving conditions (route type, weather, and illumination conditions). In both cases, the process of developing and training the proposed models is detailed, with an important focus on the availability of the data used. Accompanying this, a thorough performance evaluation of the solved tasks on

standard benchmarks is proposed. Also included is a quantitative analysis of execution times found on embedded deployment platforms.

The second part of the thesis is also divided into two chapters. The first (Chapter 4) is devoted to presenting and discussing the proposed methodology to mitigate the quadratic complexity problem of the self-attention layer of transformers. The proposed method consists of an approximation of the attention matrix by exploiting the properties of the Discrete Cosine Transform (DCT). The experimental analysis is conducted on a standard Natural Language Processing (NLP) benchmark, although the introduced approximation is application agnostic. The evaluation includes comparison of performance on the NLP task with different attention formulations used in the literature. In addition, memory consumption and inference latency as input length varies is analyzed. In the second chapter of the second part (Chapter 5) instead, it is proposed to employ a traditional lossy compression scheme, based on established vector quantization techniques, to define a latent space that abstracts the imperceptible details of the images on which the DDPM is trained. This type of generative model is trained by learning to reverse a process of image degradation, defined by corrupting the training images with noise. A limitation of this choice is that one risks wasting expressive capacity of the generative model to model irrelevant image details (such as the background). In the presented work, is addressed the strategy to adapt the DDPM for generation of compressed images with the proposed encoding, which can be decompressed at the end of the generation process to obtain the final images. The results shown are helpful in demonstrating the viability of this choice.

The results of some contributions presented in this thesis have already been, or are in the process of being, authored in high-profile publications. These works include DCT-Former [1], CERBERUS [2], and the Model-Based latent Diffusion Models [3]. Links with related publications will be explored in more detail, when necessary, in the following chapters.

Part 1: Multi-Task Learning for Automotive Perception

Chapter 2

Driver Monitoring System

2.1 Motivations

Road traffic accidents continue to pose a significant threat to public safety, claiming the lives of over 1.7 Millions each year globally. The vast majority of these accidents are attributable to the human factor, the main causal factors include conditions such as driver distraction and fatigue, emphasizing the critical need for advanced in-vehicle monitoring systems. Although vehicle manufacturers have been implementing driver assistance systems of various types in their lineup for a few years now, Driver Monitoring System (DMS) are still not widely available, and even fewer after-market devices are commercially available that can be retrofitted to existing vehicles to equip them with this technology. The reasons for the low uptake of this type of system, as well as for other driver assistance systems, are varied, ranging from the impact on the cost of the vehicle to the user perception of the technology behind these systems. A key difference, however, between DMS and other assistance systems, such as lane keeping or automatic emergency braking, is that as complex as the latter may be to design and develop, they are always clearly definable by simple physical rules and concepts. For example, it is unambiguous to describe a system that keeps the vehicle in the center of the lane of travel, whereas a system that makes sure the driver is awake and alert is certainly more ambiguous, since wakefulness and alertness are concepts that are not immediately traceable to physical measures. Thus, a DMS implementation would likely start from the analysis of certain observable parameters to attempt to infer phenomena that are attributable to losses of attention or fatigue. A theoretically viable solution is the analysis of certain biological parameters: medical literature might suggest monitoring of physiological parameters such as heart rate, respiration, body temperature, and many others to infer with reasonable accuracy the psychophysical state of a subject. Clearly, acquiring this kind of information in a realistic driving scenario is quite prohibitive, as it would require the use of specialized and potentially invasive equipment. A technically opposite solution might be to use sensor readings easily obtainable from the vehicle, such as acceleration data or more simply interactions with controls (typically steering) in order to assume the occurrence of critical conditions if certain heuristics are violated. The problem is that this type of approach is unlikely to be sufficiently accurate because of the low correlation between the observable values and the causes sought, especially if, for example, one is trying to detect the early symptoms of a phenomenon like fatigue.

2.1.1 Camera Based DMS

One viable option is to use advanced computer vision techniques to infer useful information from the images provided by a driver-facing camera installed inside the vehicle, through techniques such as posture analysis or face analysis it is possible to derive useful information to estimate the indicators of the subject's psychophysical condition. As will be explained in Section 2.2.2 there are several ways of conceiving camera-based DMS, the two main ones being based either on observation of the driver's entire body figure, mainly focusing on body and hand analysis, and approaches instead that are focused on face observation. The former are more suitable for detecting distractions due to engagement in non-driving-related activities (cell phone or infotainment use or interactions with passengers), while the latter are more appropriate for monitoring wakefulness but also lend themselves to detecting some signs of distraction. Mainly, this chapter discusses systems of the second type, which process images from a driver-facing camera, typically mounted in the steering wheel area. From an algorithmic point of view, it is unrealistic to think of developing an end-to-end classification algorithm to directly estimate, for example, the probability that a driver is about to fall asleep. This is because, training the classifier with a supervised approach would require a curated and annotated dataset of video sequences in which the event in question occurs, which are clearly not available. It is straightforward to guess how symptoms of fatigue can be inferred from a careful analysis of the face by observing the opening of the eyelids or detecting the occurrence of yawning by measuring the opening of the mouth, or how an analysis of gaze direction can indicate whether one is attentive. In the computer vision literature, especially considering the modern and powerful deep learning based methods, there is a long list of techniques and algorithms that may be useful in facial analysis, which are already used in other applications such as human-computer interaction, augmented reality (e.g, for camera effects), animation (e.g, for avatar creation) and many other areas. In practice, however, the deep learning models used in these application examples cannot be taken for granted to be directly usable in an automotive safety context due to high computational costs. As explained in the introduction, in the automotive context, it is not acceptable to delegate part of the computational load to cloud-based inference because of the stringent latency requirements and the privacy issues involved. This chapter, therefore, is devoted to detailing the work and results in the development of a computer vision model conceived expressly for the implementation of driver monitoring algorithms and intended for in-vehicle inference with reduced computational capabilities.

Multi-Task Learning for DMS

A preliminary stage of this work started with the development of an algorithm based on the detection of facial landmarks: this term refers to a set of coordinates corresponding to specific facial features such as eyes, nose, mouth, and corners of the face. By regressing the coordinates of this keypoints from a face

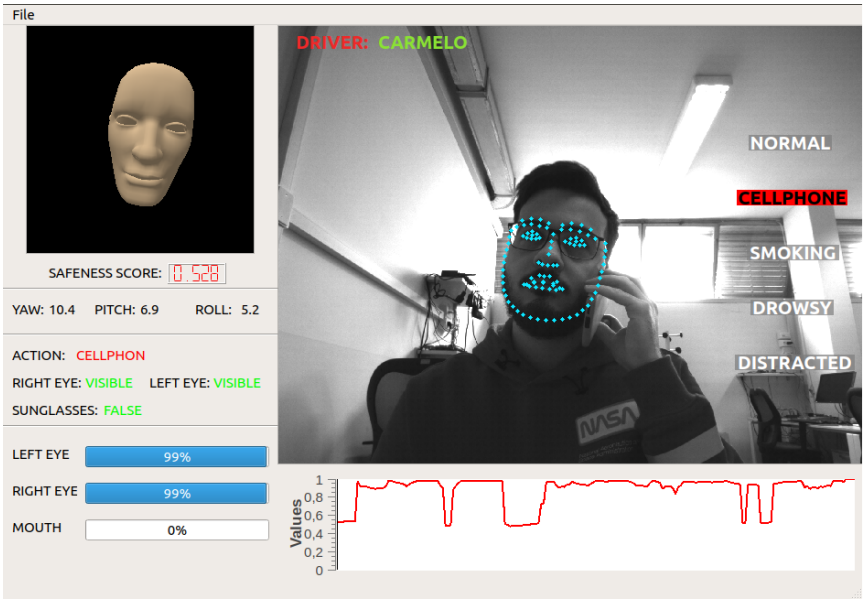


FIGURE 2.1: Interface of the developed DMS based on the proposed Multi-task network.

image provided as input, it is possible to compute useful biometrics information, such as the mouth or eyes opening level, or the head pose. In reality, a purely landmark-based approach soon proved suboptimal for several reasons, the first being that many of the deep learning based detectors tested were either too computationally onerous or not sufficiently performant and robust to the various extreme lighting conditions that can occur during driving. The second issue is that estimating the values of interest from landmarks requires additional computational cost (particularly for laying the head, which requires a PnP algorithm), and highly biased results can be obtained if one starts from inaccurate landmarks. In the end, some interesting information is simply not obtainable from landmark estimation alone, in particular, it soon became apparent that it would be useful to resolver additional classification tasks of both low-level (eye visibility classification, in case sunglasses are worn) and high-level to detect, for example, the presence of a cell phone at the ear. These reasons prompted the development of a Convolutional Neural Network (CNN) expressly designed for the purpose. In deep learning, Multi-Task Learning (MTL) involves training models to simultaneously tackle multiple tasks using the same input. Unlike traditional neural networks, which are typically trained for single-task solutions, MTL leverages shared feature extraction layers across tasks. MTL leads to significant savings in memory and computational resources by sharing feature extraction layers among tasks. Using a Multi-Task Learning (MTL) approach, the proposed network provides all the necessary outputs in a single inference step, without resorting to post-processing and heuristics and without having to use additional classifiers. The multi-task approach has led to a clear advantage from a computational point of view, but also helped to improve generalization, enhancing efficiency in training, and the ability to handle situations where there is limited data for each individual task.

To summarize, the developed model provides the following data:

- Regression of 98 facial landmarks.
- Regression of the eyelid opening value for each eye.
- Classification of the visibility of each eye.
- Classification of the level of mouth opening.
- Regression of 3 values for head orientation (yaw, pitch, and roll).
- Identification of 2 distracting actions (cell phone use or cigarette smoking).

It should be pointed out that the proposed model is only useful for inferring the information listed above from a single frame, the full DMS (Shown in Figure 2.1) involves the analysis of the temporal sequence defined by the series of predictions along a time frame. This time analysis, in the prototype system, was implemented using a set of heuristics, and therefore is not addressed in this thesis.

One of the limiting factors in setting up such multi-task models is that a dataset with each item labeled for all the tasks to solve is required. Section 2.3.2 details how a dataset existing in the literature (annotated by landmark regression and face element segmentation) was used to derive annotations for all tasks, with the notable exception of the action classification task. In the original dataset, there are no images in relevant quantities where the subject uses a cell phone or smokes a cigarette, which could be labeled as distraction tasks, at the same time there are no specific datasets for these tasks in the literature. Therefore, one of the main scientific contributions of this chapter is to describe the original methodology proposed to circumvent this obstacle.

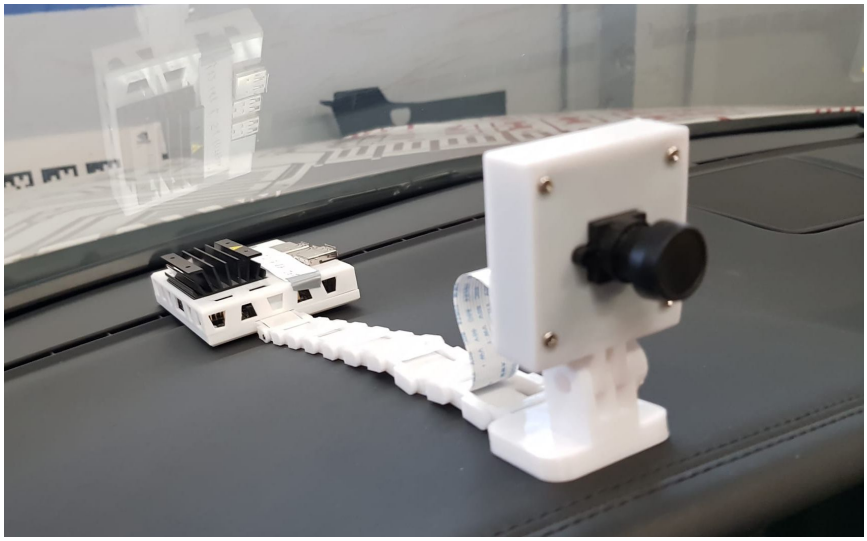


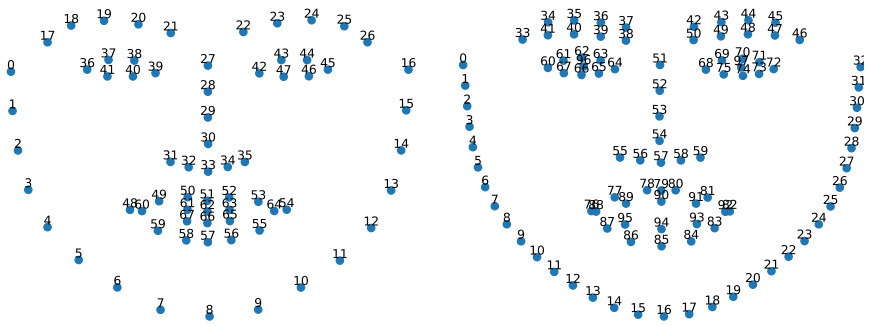
FIGURE 2.2: Prototype hardware for the DMS, based on Nvidia Jetson Nano

2.2 Background

Drawing up a comprehensive and exhaustive review of the literature related to the work in this chapter requires considerable effort. In fact, in addition to discussing the possible existing approaches to the development of camera-based DMS, it is necessary to delve into the specific literature for each sub-task included in the proposed method. Moreover, multi-task learning in itself is a broad field of study with known and several still open issues. Therefore, it was chosen to structure this background section as follows: First, the main aspects of atomic tasks that one needs to know in order to understand the work done are introduced, then an overview of the known approaches to the driver-monitoring problem is given. Finally, the vast world of deep Multi-Task Learning (MTL) is introduced, with special emphasis on the analysis of some contributions in the literature that share some aspects with the work presented.

2.2.1 Related Tasks

Facial Landmarks Detection

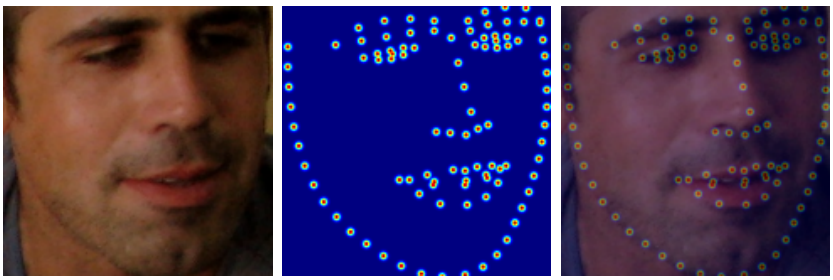


(A) 68-points (300w) (B) 98-points (wflw)

FIGURE 2.3: Labeling schemes for facial landmarks

Facial landmarks detection, which is also sometimes referred to with abuse of notation as face alignment, is a computer vision problem that involves finding a predetermined set of keypoints from an image representing a face. Formally, let N be the predefined number of desired keypoints and (x_i, y_i) with $i = (1, .., N)$ be the pixel-space coordinates of the i-th keypoint, the detection problem can be defined as the search for a mapping $\Phi : I \rightarrow [(x_0, y_0), ..., (x_N, y_N)]$ from the input image I to the set of landmarks coordinates. The key aspect is that each landmark implicitly has a semantic meaning, which is defined in relation to its specific location in the face, so the landmark set is not permutation invariant. The predetermined pattern that defines the placement of landmarks is not universal, the most common patterns in the literature today are 68- or 98-point patterns, defined in Figure 2.3; Other patterns used range from 5-point up to even 468 [4], the choice of the appropriate pattern obviously depends on the indented application.

Traditional computer-vision approaches to this problem involve the use of statistical models that encode both shape and texture variations of face images [5], [6], that can be fitted to an unseen image with an iterative process in order to recover the landmarks locations. Early approaches in the machine-learning era were based on a similar idea. Noticeably, in [7], a facial template is defined as a mean configuration of key points, which is used as an initial estimate of the position of landmarks. A cascade of regression trees is trained to iteratively refine the estimated locations of landmarks starting from this template. Each regressor progressively refines the estimate by predicting a set of offset vectors to be added to the current estimate. This method, although largely obsolete today, is still used because of its extreme computational efficiency, with the authors claiming 1ms execution time on 2014 hardware.
Deep Learning (DL) based methods define the current state of the art for the facial landmarks detection, as is now the case for almost every computer vision task. To summarize as possible the vast landscape of deep facial landmarks detections, the preeminent distinction is between coordinate regression (or direct) methods and heatmap-based methods. The former operates by directly predicting the vector of the regressed landmarks coordinates, while the latter instead produce as an intermediate representation a heatmap that encodes the spatial location of landmarks. The next chapter will discuss extensively this type of encoding, which is widely used in other fields that require keypoint regression. The usage of heatmaps for coordinates regression with Convolutional Neural Network (CNN) is mutated from the domain of Human pose estimation [8], where is successfully exploited for the regression of body keypoints. Heatmap-based landmarks regressors are closely related to the latter [9]–[11]; in short a set of N landmarks can be encoded as a heatmap $H \in \mathbb{R}^{(N \times w \times h)}$, with each location of $H_i(x, y)$ encoding the probability of the i-th landmarks being localized at that location.



(A) Source (B) Target (C) Superimposed

FIGURE 2.4: Heatmaps for facial landmarks detection.

Therefore, it is possible to train a fully-convolutional neural network to predict H from I Figure 2.4, then the landmarks coordinates for each landmark can be retrieved from the corresponding H_i selecting the location with the largest value ($(x_i, y_i) = \text{argmax}_{x,y}(H_i)$). Regression methods trained with this coding is generally considered to be more robust to variations in shape and appearance than direct regression, which is a key aspect in a complex domain such as human pose estimation. In reality, the face is a semi-rigid structure, and the variety of configurations it can assume is relatively narrow are not seemingly so evident.

The main disadvantages are that the H tensor can have a high spatial dimension and still a post-processing step is needed to obtain the coordinates. For the purpose of this work, landmarks regression is only one of the solved tasks, and still the remaining tasks need to produce 1D output vectors (Figure 2.10). Therefore, in order to simplify the architecture of the developed neural network and minimize post-processing, a direct coordinate regression approach was chosen. Direct regression models [12]–[18] are the oldest and are straightforward to define, since the neural network produces a vector of the landmarks coordinates, typically by applying one or more fully-connected layers to feature maps extracted from a convolutional backbone. One of the earliest deep detectors [12] propose to train a shallow Convolutional Neural Network (CNN) to regress the initial landmarks coordinate (only 5 landmarks were considered), then a cascade of several shallower networks are used to refine the initial estimate. This type of approach is obsolete today, and with rare exceptions single-stage models are used. More modern lines of work include the search for objective functions particularly suitable for training this task [16], [17], [19] and in general the search for strategies that can facilitate the process of learning a sufficiently rich and robust feature set [13], [18], [20].

Regarding datasets that can be used for training now there are several; One of the first datasets large enough to successfully train a deep regressor is 300 Faces in-the-Wild (300W) [21], this was obtained by combining images from 6 previously existing datasets, re annotating them using the 68 landmarks scheme. It consists of about 4000 images divided between train and test sets, making it a fairly small dataset by today's standards. The term "In The Wild" in this case indicates that the dataset contains images acquired in realistic scenarios, with variations in appearance, background, and environmental conditions, which distinguishes it from, for example, datasets acquired in a studio setup. 300W Large-Pose (LP) [22] is an extension of 300W where the images have been augmented by synthesizing new views with particularly pronounced head poses (e.g., in profile), which are notoriously the weak point of many detectors. Wider Facial Landmarks in-the-Wild (WFLW) [23] instead consist of 10,000 images in total with 98-points landmarks annotations. In addition, attribute annotations are present for 6 classes of interest (large pose, accentuated expression, abnormal illumination, use of make-up, occlusion, blur), which can be useful in the training and validation phases. For this work the newer dataset Landmark guided face Parsing dataset (LaPa) [24] is used, as detailed in Section 2.3.2; This is an extremely large dataset of more than 22,000 annotated images with 106-points landmarks and 11-class face parts segmentation masks.

Head Pose Estimation

In computer vision, the term poses estimation identifies a process by which the N-Degrees-of-Freedom transformation of an object with respect to a predetermined reference system is estimated. A typical case is 6-D pose estimation, where the goal is to derive a transformation in 6 degrees of freedom (3 of rotation and 3 of translation). This is particularly used in robotics applications to realize pick-and-place applications. Instead, in this area, the transformation to

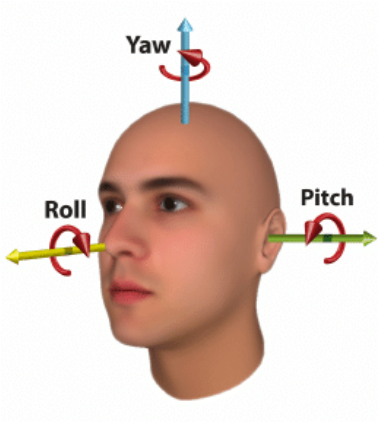


FIGURE 2.5: Euler Angles for head Orientation

be derived is that of the head, with respect to an orthogonal reference system typically defined as in Figure 2.5. Mainly for the application of DMS, one is interested in rotation, which can be expressed with the convention of Euler angles (yaw, pitch and roll). For the application of DMS, the estimation of rotation is of main interest, which can be expressed with the convention of Euler angles (yaw, pitch, and roll). Translation/scale estimation is also widely employed for applications of augmented reality, human-machine interaction, and several others. A classic method for implementing head pose estimation is to use facial landmarks estimation and a Perspective-n-Point (PnP) algorithm, which allows estimating the pose based on a set of 2D keypoints coordinates and known 3D coordinates. Typically, a subset of the detected facial landmarks is used, usually including the corners of the eyes, tip of the nose, and corners of the mouth. A fixed 3D landmarks set is defined for these points, expressed by fixing the origin $(0, 0, 0)$ of the coordinate system at the center of the head. Using this 3D set and the corresponding 2D detections, the PnP algorithm then estimates the transformation that aligns the 3D coordinates of the landmarks with their corresponding 2D image coordinates. Some main PnP algorithms are iterative algorithms, like EPnP (Efficient Perspective-n-Point) and P3P (Perspective-Three-Point).

Another category of approaches is direct pose estimation from the image using deep learning methods [25]–[27]. A limitation in the development of these methods is the difficulty in annotating real image datasets with head pose, some options include generating partially synthetic images [22], acquiring dedicated datasets using systems of or wearable devices [28]. In a multi-task learning context, it may make sense to derive ground-truth by applying the PnP algorithm to annotations of facial landmarks. Joint learning of landmark and pose can incentivize learning of both, and at inference time the pose estimation is robust to imperfect detection of landmarks and saves the execution of an iterative PnP algorithm.

2.2.2 Driver Monitoring

The scientific literature on camera-based driver monitoring system is extremely scattered and fragmentary, therefore producing a compressive survey of this subject is beyond the scope of this chapter. In this section, therefore, an attempt is made to outline a general taxonomy of the most valued approaches, trying to highlight the different ways in which driver monitoring can be conceived. A first categorization relates to the positioning of the camera in the vehicle, which naturally has a decisive impact on the type of analysis possible. In one case, the camera faces the driver's face and mainly the face region is analyzed to infer signs of drowsiness or distraction. In the other case, the camera observes the entire upper part of the driver's body, typically in profile, with a camera typically placed on the door pillar opposite the driver's side.

Face based

By analyzing the frontal image of the driver's face, this type of DMS lends itself particularly well to detecting signs of fatigue through an analysis of the biometric parameters that can be inferred. Face-based drowsiness detection is by no means a recent topic, in a seminal study on driver drowsiness [29] from 1994 the PERCLOS is introduced, which to this day is still used. PERCLOS is originally defined as the percentage of the time over one minute that the eyelids are at least 80% closed. This definition can be generalized as the PERCLOS(%) as a function of the percentage of the time that the eye is closed [30].

$$\text{PERCLOS} = \frac{\text{Total Time with Eyes Closed}}{\text{Total Monitoring Time}} \times 100\% \tag{2.1}$$

From an implementation standpoint, a PERCLOS-based drowsiness detection system can be designed by relying on the detection of facial landmarks [31]. The eye-opening state can be derived from the measurement of the Eye Aspect Ratio (EAR), defined as the ratio between height and width of the eye. This measure is also used in the data preparation phase of this work, and is detailed in Equation (2.3). Relying only on landmarks to measure eye openness, however, is not particularly robust with respect to variations in head pose (e.g., bowing the head slightly may incorrectly indicate that the eyes are closed). For this reasons, over the years, some datasets have been introduced to perform direct classification of the eyelid opening. The ZJU Eyeblink database [32] is one of the oldest, it consists of 80 clips of 20 subjects portrayed in 4 different conditions. The labels for eye blinks are provided, which can be used to isolate the frames with closed eyes. The Closed eyes in the Wild (CEW) dataset [33] provide 1,193 patches of 24×24 pixels obtained from face images collected from publicly available sources. MRL Eyes [34] is another similar dataset, also provided eye patches with eye-opening labels, while also providing labels for gender, lighting conditions, presence of reflection or of glasses. Relying solely on observation of the eyes for the development of DMS is limiting; first, they may be occluded by the presence of sunglasses or even prescription glasses; second, there are other signs of fatigue that deserve to be considered.

One of the earliest signals of drowsiness is yawning, this can be detected by measuring the degree of mouth opening over time. Similarly to the EAR, the Mouth Aspect Ratio (MAR) can be defined using facial landmarks detected for the mouth region. YawDD [35] is a dataset produced explicitly for the task of yawn detection, the effort that has been put into this work by the authors is considerable, since it consists of 107 subjects (57 male and 50 female) and is acquired in realistic driving scenario under various lighting conditions. Two sub-datasets are provided, the first consists of a total of 322 videos of 5-40 seconds recorded with a camera installed under the front mirror. Each video has a class label (1- normal driving, 2- talking, and 3- yawning). For the second sub-dataset the camera is installed over, and a single video is acquired for each subject where all the 3 actions are performed consecutively. A drastic approach to drowsiness detection is the end-to-end approach by direct classification of the temporal sequence of images. The work of [36] is an important building block in this direction, thanks to the introduction of the Real-Life Drowsiness Dataset (RLDD) dataset. It consists of 30 hours of video of 60 subjects (51 men and 9 women) each labeled with an alertness level class (Alert, Low Vigilant or Drowsy). Participants in the dataset personally acquired video sequences at different times of the day, self-reporting their physical state. The problem with end-to-end classification is that the supervisory signal is the single label for each video, which is extremely sparse information that is unlikely to benefit the extraction of a sufficiently rich spatio-temporal feature set.

2.2.3 Multi-Task Learning

Multi-Task Learning (MTL) is regarded in a seminal work [37] as "an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias". In the context of deep learning MTL involves the development and training of models that can solve multiple tasks simultaneously with the same input, in contrast, typically a neural network, even if it produces multiple outputs, is trained to solve only one task. As pointed out in [38], this approach is typically regarded of two advantages: first, individual tasks can benefit each other during training and lead to an overall improvement in performance. Intuitively, this may be justified by the fact that multi objective optimization may lead to the learning of a richer feature set, on the other hand, however, a negative-transfer phenomenon could occur where the different objectives counteract each other degrading performance. The second main aspect of multi-task learning is the substantial savings in memory and computational resources due to the sharing of multiple feature extraction layers among multiple tasks; this aspect is of particular interest for the purpose of this thesis. A macroscopic division among multi-task learning approaches is between hard parameter sharing and soft parameter sharing methodologies.

In hard parameters sharing typically there is a shared backbone between all tasks that eventually branches into dedicated heads to produce the output for individual tasks; the shared backbone (typically a convolutional network)

extracts the common feature set, while the dedicated heads learn the map between the feature set and the output domain. During training, the shared weights are updated based on the combined loss from all tasks, thus enforcing strong constraints towards learning representations that are beneficial for all tasks simultaneously. This approach is particularly beneficial when tasks are closely related and share similar basic characteristics. However, it may not be flexible enough to adapt to variations in task complexity or requirements. In contrast, in Soft parameter sharing each task have it's own set of parameters and a cross-talk mechanism is employed in order to encourage parameter similarity across tasks, typically by penalizing differences between them. Soft parameter sharing provides a balance between task-specific learning and leveraging shared information across tasks, allowing the model to adapt to the inherent differences of each task while still benefiting from shared representations. However, soft-parameter sharing causes the fundamental element of computation and memory savings to be lost, so the work presented in this chapter falls strictly within the domain of models with hard parameter sharing.

Multi-Task Driver Monitoring

There are a few contributions in the literature with a more or less wide intersection with the work presented, so the following are some of the most similar works.

In [39] they introduce the Multi-Task Driver Monitoring Framework (MT-DMF), it is a pipeline that combines classical computer vision algorithms for face analysis, implemented in the DLib library [40], with an original CNN that simultaneously classifies the state of eye-opening, mouth, head pose, and most interestingly, the estimate of driver fatigue. First, DLib's face detector is used to identify and map the RoI of the face, also facial landmarks detection is performed, again with the implementation of the [7] algorithm present in Dlib. The landmarks are used to derive a binary mask identifying the eye region, this mask together with the cropped image of the face is given as input to the multi-task CNN. They propose an architecture based on the venerable VGG19 convolutional model [41], which they pre-train on a facial expression recognition task and fine-tuned on the driver monitoring tasks using the NTHU-DDD dataset [42]. Although they provide some accuracy measures taken on the same dataset the results of this work are quite elusive; The dataset used is very limited, consisting of only 36 subjects, acquired in a sudio setup with little variety of scenery, appearance, or illumination. In addition, the use of Dlib for detection of landmarks, which are then used marginally, appears to be an outdated choice.

The work of [43] is two-pronged, the first dealing with the development of a multi-task network for driver monitoring, while the rest of the paper describes the hardware/software architecture designed for the deployment of the in-vehicle system. Regarding the model, one of the main contributions is the acquisition of an original dataset for training the multi-task network. This dataset is not publicly available, it counts of $18,954$ annotated images with the state of the eyes (open, closed) head (front, up, down, right, left) and mouth (open, closed)

and an overall label of the driver's state (normal, distraction, fatigue, drowsiness). The driver state ground-truth has been labeled using a combination of PERCLOS, duration of eye closure, mouth opening, and head direction. The proposed multi-task classification model is based on a Mobilenet architecture and is trained for the the 3 facial behavior classification tasks (eyes, head, and mouth status). At inference time the driver state is estimated based on a combination of the aforementioned classification values over a period of time. The main contribution of this work is certainly in the definition of the dataset, but it appears to be acquired in a controlled scenario, again lacking in diversity of scenario and appearance. The fact in fine that the dataset or model is not available limits its scientific interest.

[44] is the closest work ever to the one presented and has some fascinating aspects. First, they introduce a novel dataset for driver monitoring (FDUDrivers), consisting of 20000 images of 100 different driver, supposedly acquired in real driving conditions. Unfortunately, to date, this dataset is publicly unavailable, so an in-depth analysis is not possible. They develop a lightweight multi-task CNN (DANet) to solve the tasks of head pose estimation, landmarks detection and of special interest gaze estimation. The proposed model adopts an interesting design: the first convolutional layers, which process the input image, are totally shared among all 3 tasks, a first branch is dedicated to the landmark detection and pose estimation tasks, which share the remaining convolutional layers. Another branch is instead specialized for gaze estimation, the intriguing aspect is that the output of landmark prediction is employed to isolate the feature map region as input to the gaze estimation branch, using the ROI align layer [45]. The pose estimation output also contributes to the prediction of gaze, via the "Object centric Aware Block" module presented in their work. It is not entirely clear how this one is trained, since the output of gaze estimation depends on the other two. It would be important to understand whether predictions from the other branch or ground-truth values are actually used in training. No specialized classification labels are provided (e.g., for the eyes or mouth), although they could likely be easily added.

The differences that distinguish the work presented in this chapter from the work seen above are many; for one thing, the number of tasks solved simultaneously (6) is the highest ever, and the training data have a wide variety of appearance, allowing for exceptional levels of robustness under deployment conditions. The addition of the distraction classification tasks is one of the most relevant aspects, since no other work appears in the literature that does anything similar. In the end, a great effort has been made to present an extensive and comprehensive experimental treatment.

2.3 Proposed Approach

2.3.1 Problem formulation

As presented in the introduction, the developed model makes it possible to obtain all the necessary information for the implementation of the DMS in a single inference step. More precisely, the model consists of a convolutional neural network that takes as input an RGB image $I \in \mathbb{R}^{(3\times w\times h)}$ depicting the region of the image that tightly encloses the subject's face. In the reference implementation of the DMS, an initial face detection step is therefore necessary to extract the region of interest; for this purpose, an object detection model such as those introduced in the next chapter, trained on a face detection dataset, can be used. The output of the model encodes the predictions of the individual regression and classification tasks into a single one-dimensional vector $O \in (1 \times d_o)$, the composition of individual outputs is summarized as Table 2.1.

Task	**Type**	**Shape**	**Act**	**Range**
Landmarks	Reg	196	None	[0,1]
Eyes	BC	4	sigmoid	[0,1]
Mouth	MC	3	softmax	0,1,2
Head	Reg	3	None	$[-\frac{\pi}{2}, \frac{\pi}{2}]$
Distr.	MC	3	softmax	0,1,2

TABLE 2.1: Definition of outputs. Regression (**Reg**), Binary Classification (**BC**) and Multi-class Classification (**MC**).

Specifically, the coordinates (x, y) of the 98 facial landmarks are concatenated in 196-dimensional vector and expected to be expressed in range $(0, 1)$, normalized with respect to image height and width. The 3 Euler angles for head orientation (yaw, pitch, and roll), expressed in radians, are instead not normalized. For the classification tasks, standard output activation functions are applied in order to obtain the values sought, applying the element-wise sigmoid to the 4 predicted values for the eyes constrains each value in the range (0,1). For multi-class predictions, on the other hand, softmax is used, which constrains the vector of predictions to a probability distribution over the possible classes. Thus, an occlusion value and an eyelid opening value are predicted for each eye, with a value close to 1 indicating that the eye is visible (e.g., not occluded by sunglasses) and respectively that the eyelid is open. For the mouth, on the other hand, the opening is classified into 3 predetermined levels: closed, semi-open or open. Finally, the distraction classification task predicts three possible classes: normal, cell phone use worn to the ear, or cigarette smoking.

Addressing the lack of training data

As already mentioned, one of the major difficulties in developing this work is the limited availability of training data, particularly for distraction classification tasks, for which there are no annotated data in the literature. The proposed solution implies the acquisition of a small set of images for each of the 2 classes

of interest (cell phone use and smoking), to be combined with the large literature dataset used for all other tasks. A training strategy is then developed to overcome the lack of labels for the remaining tasks in the face of minimal manual annotation work, and finally then train the model simultaneously for all tasks. Specifically, the acquired images are manually labeled by annotating only 5 particularly representative facial landmarks, then a version of the model previously trained on the literature dataset is used to derive pseudo-labels for the new images. The degree of alignment between the 5 hand-annotated landmarks and the corresponding predicted landmarks (isolated from the 98 total output) provides a proxy for the goodness of fit of the pseudo-labels obtained through the pretrained model. A new dataset is defined by combining the literature dataset (where the class for distraction is assumed to be "normal") with the subsets of the pseudo-annotated images: the resulting dataset is obviously extremely unbalanced on the 'normal" class, so an appropriate training strategy is required. It is then possible to train all 5 tasks simultaneously from scratch, using the pseudo-labels for the acquired subsets; a particular loss formulation that uses the computed alignment value to weight the individual loss contributions is crucial at this stage to mitigate performance degradation due to inaccurate pseudo-label. The acquisition and pre-processing of the distraction subsets is detailed in Section 2.3.2, while in Section 2.3.4 the formulation of the weighted objective functions is provided.

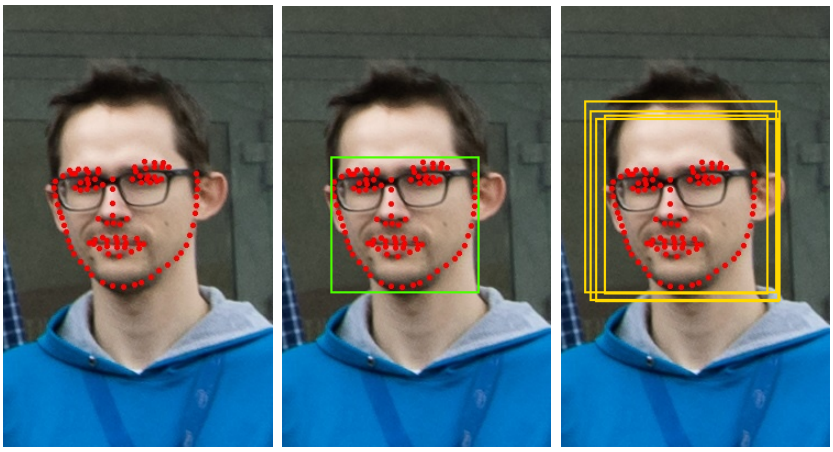
2.3.2 Data Preparation

FIGURE 2.6: Samples from the LaPa Dataset

As the base of this work the Landmark guided face Parsing dataset (LaPa) [24] Dataset is chosen, this is a relatively recent dataset and less established than others such as 300w [21] and WFLW [23] but has several advantages. This dataset consists of 22,176 annotated images, making it one of the largest existing in the literature in its category, among which the authors isolated 2,000 images for validation and 2,000 for testing, with thus 18,176 reserved for training. Each image represents a face in the foreground, for which annotations are provided for 106 facial landmarks and a 11-class pixel-level semantic segmentation map. Segmentation maps in particular were found to be useful for deriving

auxiliary labels for multi-task training; the complete pre-processing pipeline is presented below.

Labels and Preprocesing



(A) Landmarks (B) Minimal RoI (C) Augmented RoIs

FIGURE 2.7: Example of RoI Augmentation

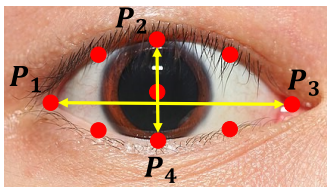
At inference time, the developed model will take as input the face region cropped from the original image; an off-the-shelf object detector trained for a face detection task will be used to detect the face Region of Interest (RoI). The RoI identified by the detector will never be perfect and will always involve slight variations in alignment and scale from the ideal face region, so it is critical that the developed model be robust to these variations. A careful pre-processing of the training set is essential to achieve the robustness. To begin, the 106 annotated landmarks are subsampled by selecting only the 98 in the WFLW format, then from the landmarks it is possible to define the minimum RoI which tightly encloses the labeled landmarks. The augmented RoIs are obtained by applying shift and scale factors to the minimal RoI, specifically first the aspect ratio is adjusted to a square format, then a scaling factor is applied to include a larger region, and finally a shift is applied in the x and y directions, always taking care to include all annotated landmarks. The augmented RoI is used to crop the image, which can be subject to further spatial augmentation by applying rotations ad horizontal mirroring; it is essential to apply the same transformations to the image and landmarks annotation in order for these to be preserved for the augmented image. In Table 2.2 values used for the preparation of train and test/validation splits are given. For the training set, the shift, scale and rotation values are uniformly sampled over the range of values chosen. For efficiency reasons, spatial augmentations are precomputed and are not applied dynamically during the train, 10 augmented versions with different combinations are processed for each train image. In addition, appearance augmentation are randomly applied during the training, including variations in brightness and contrast, color jittering, blurring and addition of Gaussian noise.

In the pre-processing stage, subsequently to the spatial augmentation, multi-task training labels are also derived. For the eye tasks (occlusion and eyelid

Operation	**Train**	**Test/Val**
Shift	[0.0, 1.0]	0.0
Scale	[1.0, 1.1]	1.0
Rotate	[-35°, +35°]	0.0°
Flip (*prob.*)	0.5	0.0

TABLE 2.2: Spatial augmentation configurations

opening labels), landmarks and segmentation mask are leveraged; For each eye, first the Eye Aspect Ratio (EAR) defined in Equation (2.2) is computed using the landmarks, a value of EAR close to 1 should indicate 'opening of the eyelid (resp. a value close to zero closing), but at a preliminary stage it was noted that this metric alone is not sufficiently reliable. Therefore, $\mathrm{P}_{l/r}$ defined as the pixel count for the left/right eye label in the segmentation mask used in addition for the definition of the visibility ($\mathrm{viz}_{l/r}$) and eyelid opening ($\mathrm{open}_{l/r}$) labels. The definition of the labels is reported in Equation (2.3)



$$\mathrm{EAR} = \frac{\|P1 - P3\|}{\|P2 - P4\| + 1^{-3}} \tag{2.2}$$

$$\begin{aligned} \mathrm{viz}_{l/r} &\stackrel{\text{def}}{=} \mathrm{P}_{l/r} > 10 \text{ and } \mathrm{EAR}_{l/r} > 0.1 \\ \mathrm{open}_{l/r} &\stackrel{\text{def}}{=} \mathrm{viz}_{l/r} \text{ and } (\mathrm{P}_{l/r}\ \mathrm{EAR}_{l/r}) > 13.5 \end{aligned} \tag{2.3}$$

The label for the level of mouth opening is derived similarly, using the pixel count derived from segmentation and an aspect ratio calculated using landmarks. Finally, in the end, the Euler angles for head rotation are estimated with the Perspective-n-Point (PnP) algorithm, which allow estimating the pose of a 3D object by associating its 2D image projections with known 3D coordinates. Given a standard set of 10 3D (x, y, z) landmark coordinates used in the literature and the corresponding 2D labeled keypoints, the 6-DoF pose is estimated by minimizing a 3D-2D projection problem. The clear advantage of training the network to predict rotation instead of using PnP on predicted landmarks is the savings of running the algorithm in inference (PnP algorithms can be expensive) and being robust to inaccurate landmark predictions.

Distracting Actions Dataset

For the classification tasks, two subsets of images were acquired for the classes cell phone usage and smoking detection. The images used were collected from various publicly accessible sources by searching for specific keywords and were manually curated to verify that they met the criteria sought, (i.e., the cell phone is worn to the ear or the cigarette in the mouth is clearly visible). Thus, 858 images were obtained for the 'cellphone" class and 715 for the 'smoking" class; considering that the more than 20 thousand LaPa images are considered for the

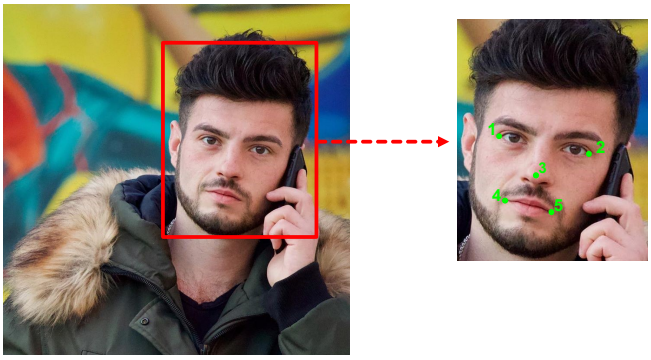
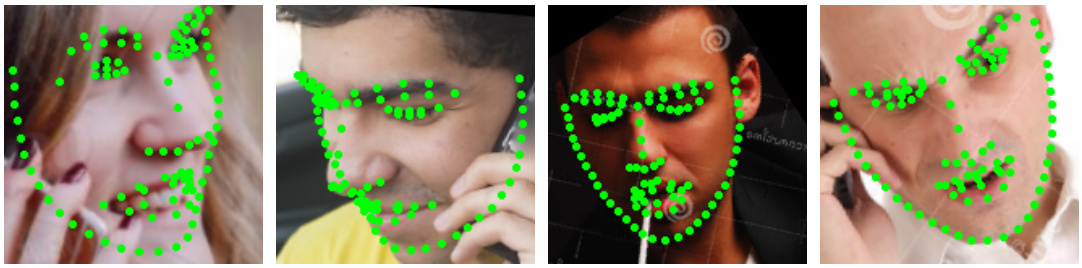


FIGURE 2.8: Labeling of the five reference landmarks

'normal" class, the severe imbalance of the dataset on this task is evident. To implement the multi-task training strategy by including the distraction task, it was necessary to manually annotate 5 facial landmarks, as depicted in Figure 2.8, for all the collected images. To obtain the full (pseudo) labels, the most powerful version of the model (Large), described in Section 2.4.4 and trained on the tasks of LaPa, is employed. The subsequent spatial augmentation steps are reproduced as well as for the original dataset with regard to RoI extraction, the labels, on the other hand, are kept those predicted by the pretrained model.

$$c = \alpha \cdot (1 - \log(d)) \tag{2.4}$$

The confidence score is defined according to the alignment between the 5 annotated keypoints and the corresponding predicted, is calculated as defined in Equation (2.4), where d is defined as the 2-norm of the difference between the predicted and labeled keypoint pairs (normalized with respect to the image size) and $\alpha \in (0, 1)$ is a scaling factor.



(A) $c = 0.633$ (B) $c = 0.765$ (C) $c = 0.887$ (D) $c = 0.998$

FIGURE 2.9: Pseudo-labeled images with different confidence values

It was qualitatively observed that the confidence thus calculated is an extremely reliable proxy for the goodness of predictions, as can be guessed from Figure 2.9. In Section 2.3.4 is detailed the strategy to leverage the pseudo-labeled images and the confidence score to simultaneously optimize for all tasks.

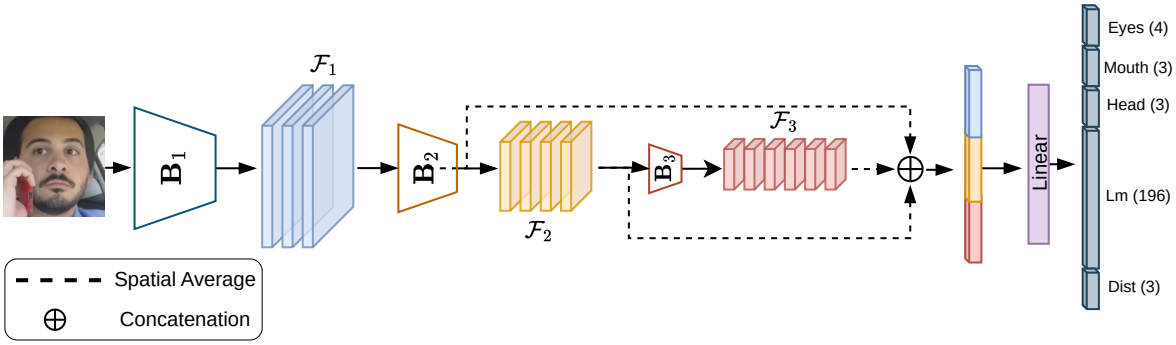


FIGURE 2.10: Architecture of the proposed Model

2.3.3 Model Architecture

The architecture of the proposed model is designed to balance the tradeoff between computational efficiency and performance, while at the same time preferring a simple, well-understood approach that does not make use of custom layers and operations that could make deployment on the inference platform complicated. Thus, the architectural pattern of Mobilenet-v2 [46], although no longer very recent, was chosen as the starting point. This is a family of convolutional models for vision applications, designed to optimize computational cost and memory consumption; the key choices in the mobilenet-v2 family are the replacement of simple convolutional layers with depthwise separable convolution blocks and the use of Inverted residual as the building block base.
Specifically, the separable convolution replaces a convolutional layer $(n \times n)$ with a depth-wise $(n \times n)$ convolution followed by a point-wise (1×1) convolution. For an input tensor $(c \times w \times h)$, a regular convolution applies a set of convolutional filters $(k \times c \times n \times n)$ to obtain an output $(k \times w' \times h')$ (leaving out the factor of stride and padding), therefore each of the k filters is applied across all input channels simultaneously. Instead, in a depth-wise convolution (as depicted in Figure 2.12), a convolutional filter is applied to each input channel, resulting in a set of output channels equal to the number of input channels. The point-wise convolution, instead, is just a regular convolution with a kernel size of (1×1); therefore, it is easy to derive how replacing a single convolutional layer with the separable convolution results in a saving of $c((k-1) * n^2 + k)$ parameters.

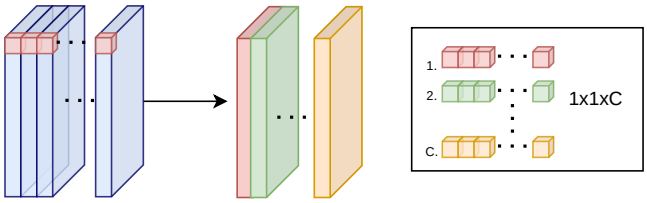


FIGURE 2.11: Point-Wise Convolution

A mobilenet-v2-*type* network is defined as a stack of Inverted Residual blocks, each characterized by the number of output channels (o), the expansion factor (t) and the stride (s); If $t > 1$, an initial point-wise convolution is applied

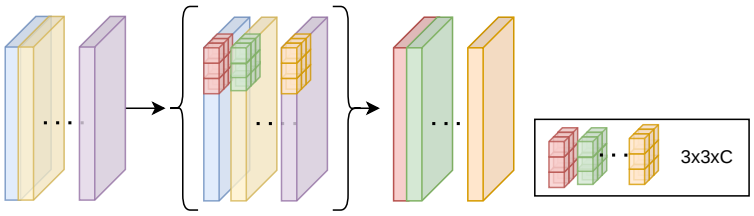


FIGURE 2.12: Depth-wise Convolution

to increase the number of input channels from k to kt, then the depth-wise convolution is applied with stride s, followed by the point-wise convolution with o output channels. If the dimensions are compatible ($s = 1$ and $o = k$), a skip connection is introduced by adding the input of the block to its output, this choice is commonly justified as to improve the flow of information and gradients, mitigating vanishing gradient problems and enabling more effective training.

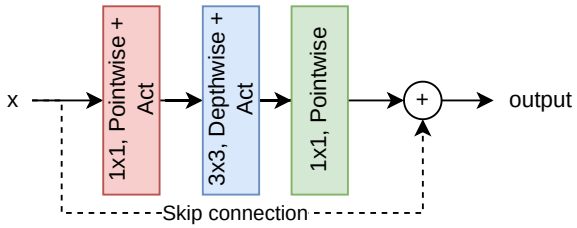


FIGURE 2.13: Inverted Residual Block

For this work, such an architecture was developed in different configurations, the exact number of blocks, as well as the configuration of each, depends on the exact variant; in total 3 different configurations with incremental computational cost (`tiny`, `small` and `large`) are proposed. In Section 2.4.4 a comparison of the performance of the three variants is proposed, when not otherwise indicated the intermediate (`small`) configuration is always used. As depicted in Figure 2.10, the feature maps ($\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$) of two intermediate convolutional blocks ($\mathbf{B}_1$ and $\mathbf{B}_2$) and the last block ($\mathbf{B}_3$) with different stride factors (4, 8, 16) are averaged across the spatial dimension to obtain a single value for each channel, the resulting vectors are then concatenated and fed to the final layer. A single fully-connected layer is used to obtain the output vector, aggregating the intermediate feature maps to produce the output is aimed at improving robustness and generalization by aggregating information at varying scales.

2.3.4 Optimization Strategy

Objective functions for Base Tasks

The choice of the objective function for each task is critical to the overall performance of the model; when training a multi-task model, it is not precisely predictable which effect the variation in the objective function chosen for one task might have on the training of all the others, so the development of an optimal training strategy can be extremely laborious. For this work, it was deemed

appropriate to start with sufficiently understandable literature objective functions. Hereafter, referring to the taxonomy of Table 2.1 the losses tested for each of the baseline tasks are detailed.

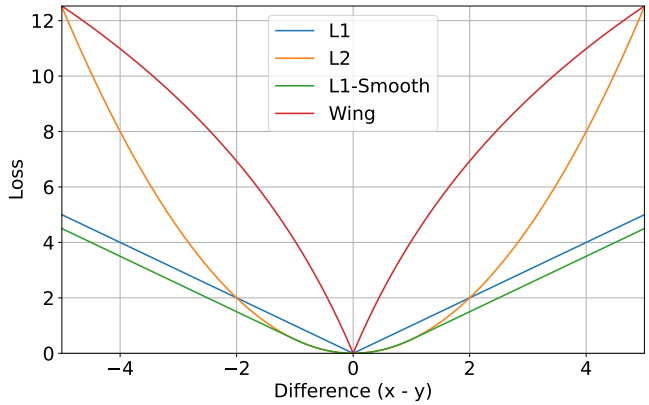


FIGURE 2.14: Comparison of objective functions of landmarks localization

The facial landmarks localization task is certainly the one to be given the most attention, since as mentioned in Section 2.3.2 convolutional networks have considerable difficulty in direct coordinate regression tasks. Given the vector of predicted landmarks t and the corresponding prediction p, the general formulation is defined in Equation (2.5) as a function d of the difference between the predicted and ground-truth values.

$$\mathcal{L}_{lm}(t, p) = \sum_{i=0}^{2N} d(t_i - p_i) f \tag{2.5}$$

The most common d choices for this type of regression are norm-1 (absolute value) and norm-2 (mean squared error), both reported in Equation (2.6). A middle ground between the two is represented by the smooth-$L1$,

$$\begin{aligned} \text{L1}(x) &= |x| \\ \text{L2}(x) &= \frac{1}{2}x^2 \end{aligned} \tag{2.6}$$

This function is quadratic for small values of x and linear for larger values, the rationale behind this strategy is to simultaneously mitigate the magnification of larger errors due to the squaring operation of L2 while avoiding the over penalization of small errors induced by the L1.

$$\text{Smooth-L1}(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| < 1 \\ |x| - \frac{1}{2} & \text{if } |x| \geq 1 \end{cases} \tag{2.7}$$

The authors of [17] observed that the above functions cause updates to be dominated by large location errors, masking the contribution of relatively small shifts. They propose the Wing-loss function, reported in Equation (2.8); for values of $|x| \in (-w, w)$ it behaves logarithmic, controlled by a curvature factor ϵ,

in order to accentuate the contribution of minor localization, outside has a linear trend that is useful in mitigating the excessive contribution of large localization errors in the early stages of the train. The term $C = w - w\ln(1 - |x|/\epsilon)$ ensure a smooth transition between the linear and nonlinear regions.

$$\text{WingLoss}(x) = \begin{cases} w\ln(1 - |x|/\epsilon) & \text{if } |x| < \epsilon \\ |x| - C & \text{if } |x| \geq \epsilon \end{cases} \tag{2.8}$$

In [47] the same authors note that wing-loss could cause doggedness on negligible localization errors, so it is useful to introduce into the formulation a region small $(-r, r)$, with r typically set to 0.5, such that for $|x| < r$ the loss is always 0. With this addition, therefore, the Rectified Wing-Loss could be defined.

For the remaining tasks, the choice of objective function more straightforward, in particular for the multi-class mouth opening classification task the standard Cross-entropy loss with Softmax is used, instead for each of the single-class eye classification tasks the binary cross-entropy is used, and the individual terms are added together. The regression of Euler angles requires some shrewdness, in fact functions such as L1 or L2 would not take into account the circular nature of these values and can lead to errors, so (Equation (2.9)) the operation modulo versus period 2π is introduced to the difference in absolute value of the angles (in radians) to properly calculate the offset between angles. In addition, the cosine function helps to smooth the penalization when approaching zero.

$$\text{EulerLoss}(\gamma, \overline{\gamma}) = 1 - \cos(|\gamma - \overline{\gamma}| \mod 2\pi) \tag{2.9}$$

Distraction Classification Loss

Optimizing the distraction classification task without degrading the performance of the basic tasks is definitely the most challenging aspect of this work. In fact for this task there is a substantial imbalance of more than 10:1 in favor of the base class, so the cellphone and smoking classes are extremely underrepresented, plus for the subset of acquired images only the pseudo-labels obtained as described in Section 2.3.2 are available. In machine-learning problems, imbalance in the dataset is often approached with balancing techniques such as undersampling the dominant class or oversampling the minority class, however in this peculiar multi-task problem the usage of these techniques would inevitably alter the distribution of the remaining tasks, with hard track side effects. Therefore, the best thing is to act at the loss level exclusively in optimizing the affected task.

The most general formulation of loss for the multi-class classification task is given in Equation (2.10), where x and y are respectively the one-hot vector for the ground-truth class and the output of the neural networks, both vectors of n elements. The most popular choice for the error function $f(\cdot)$ is the Cross-Entropy function, which is expressed in Equation (2.11) as the negative logarithm of the softmax.

$$\mathcal{L}_{MC} = \sum_{i=0}^{n} f(x, y)_i \tag{2.10}$$

$$\text{CE}(x, y)_i = -x_i \log \left(\frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \right) w_i \tag{2.11}$$

In addition, a weight w_i can be introduced to balance the contribution of each class to the overall loss. This option can be useful for handling class balancing in a problem such as the one under consideration: intuitively the dominant class should be penalized with a low w value to favor underrepresented classes, counterintuitively however this reasoning can be overturned. The value of Equation (2.11) is zero unless i is the ground truth class ($x_i = 1$), therefore w_i balance the contribution of the elements that *should be* classified as class i, in other words a large w_i would severely penalizing false-positives for the class i. Thus, assigning a high weight to minority classes incentivizes the model to learn the decision boundary from them, resulting in potentially implying overfitting on the training images. As can be appreciated from the samples in Figure 2.9, the subsets for the minority classes (cell phone and cigarette) are little varied and the images are very easy to classify, with the result that effectively this strategy leads to minimizing the classification loss by learning subtle peculiarities of the training dataset, essentially overfitting on the training samples and annihilating the ability to generalize to realistic situations. Instead, assigning a larger weight to the majority class discourages this kind of behavior, leading to decision boundaries that are more conservative around the majority class.

$$\text{FL}(x, y)_i = \alpha \left(1 - \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \right)^{\gamma} \text{CE}(x, y)_i \tag{2.12}$$

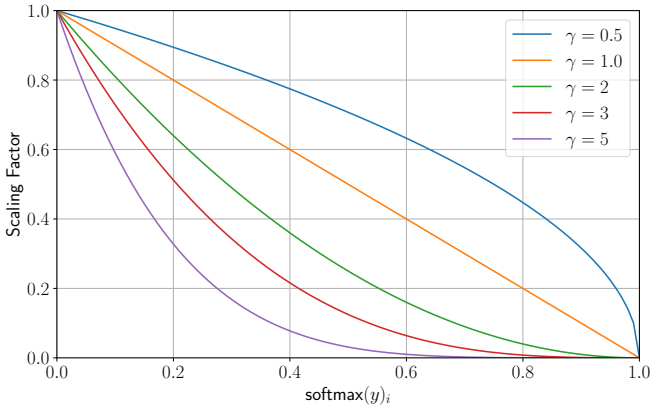


FIGURE 2.15: Plot of the exponential smoothing term used in the Focal Loss

As an alternative to Cross-Entropy, Focal Loss [48] is considered; this objective function is designed for cases where there are numerous easy-to-classify examples whose contribution eventually dominates the gradient magnitude.

As defined in Equation (2.12), focal loss essentially introduces an exponential weight factor modulated by a term γ to cross entropy. The ratio that appears in parentheses is simply $\text{softmax}(y)_i$, when $\gamma > 1$, the larger is the softmax score for class i (and thus the more confident the prediction model is), the smaller the exponential weight will be, as shown in Figure 2.15. The scalar term α is present in the original formulation but does not play a relevant role here. As a result, there is an incentive to learn from samples over which the model has little confidence and which are assumed to be the most difficult to classify. Potentially then it lends itself to the case of interest, in fact as already mentioned most samples are easy to classify and this can lead to an excess of false positives, (e.g., simply by touching the hand to the face can be misclassified as cell phone use).

Combined Optimization

The overall loss for multi-task training is simply a weighted sum of the individual contributions, in Equation (2.13) α_i denote the weight for the i-th task. In addition, the overall loss, for each task, is the sum of the contributions of the individual elements of the minibatch, each with its own weight $\gamma_j \in (0, 1)$; the When training only the basic tasks of LaPa γ_j is always 1 (so this weight is not used). Instead, when training the full model by including pseudo-labeled images for distraction tasks, this mechanism is crucial in mitigating the propagation of imperfect labels in the learning process. The confidence score of Equation (2.4), computed using the difference between 5 hand-labeled reference keypoints and the corresponding predictions from a pseudo-labeler teacher model, is the γ for the pseudo labeled samples, instead, the LaPa samples always have confidence 1. The same γ is used for all tasks of a minibatch element, except for the distraction task (when enabled), for which the weight is not applied since the distraction class is not pseudo-labeled.

$$\mathcal{L}_{TOT} = \sum_{i=0}^{NT} \alpha_i \sum_{j=0}^{BS} \gamma_j \mathcal{L}_{[T_i],j} \tag{2.13}$$

Beyond objective functions, the other key aspect to consider is the choice of the optimizer and the related strategies and hyperparameters. Without delving too deeply into the intricacies of how optimizers work, which is beyond the scope of this thesis, the Stochastic Gradient Descent (SGD) is the straightforward implementation of the mini-batch gradient descent algorithm, where the gradients of the loss function with respect to the model's parameters are computed using the back propagation algorithm, and the weights are then adjusted in the opposite direction of the gradient to approach a minimum of the loss. With the introduction of the momentum, in addition, a weighted average of past gradients is accumulated to accelerate convergence to a stationary point. Adam, on the other hand, which is likely to be the most popular optimizer in the field of deep learning, adapts the learning rate for each parameter based on the first and second moments of the gradients, allowing for adaptive and per-parameter optimization. It is generally believed that Adam is faster but tends to lead to sharp local minima, instead SGD with momentum is believed

to lead to flatter regions of minima, resulting in improved generalization ability [49].

Stochastic Weight Averaging (SWA) is a technique used in the field of deep learning that has been repeatedly shown to be useful in improving generalization capabilities [50]. SWA is orthogonal to the optimization algorithm, as it works by averaging multiple snapshots of the model weights collected during the training phase. With respect to using a single set of weights, this is shown to help smoothing the optimization landscape, mitigating the effect of sharp minima and encouraging the exploration of wider optimality regions, ultimately leading to more robust and stable models.

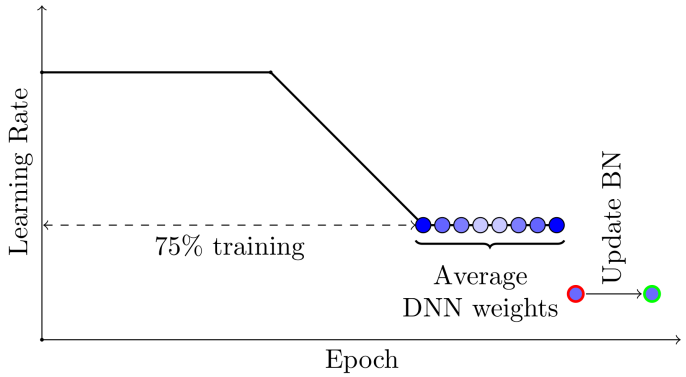


FIGURE 2.16: Typical Learning Rate strategy used for SWA

Typically, (Figure 2.16), SWA is implemented by training with SGD, normally using a learning rate schedule, for the most part of the epochs. Snapshots are only collected in a final portion of the training, setting a sufficiently high constant learning rate or using a dedicated schedule to incentivize exploration of the solution space, in the end the average weights are calculated, which will be used for the inference. When Batch Normalization (BN) is used in the model, the running mean and variance statistics of the BN layers are typically not averaged with the weights, instead a forward pass on the training set is performed to update the activation statistics at the end of the training process.

2.4 Experiments and Evaluation

2.4.1 Experimental Setup

This chapter is devoted to the presentation and discussion of the results of the experimental analysis conducted. First, the protocol for validating the results and the methodologies used to corroborate the qualitative observations are introduced.

Metrics

Evaluating and comparing the performance of such a multi-task model is not a trivial task, first because the appropriate metric must be defined for each sub-task, not being universally clear when comparing what the appropriate trade-off might be in case a worsening on one task accompanies an improvement on another. Moreover, in this specific case, no metrics could be obtained for the distraction classification task, since an adequate validation set could not be isolated due to the very limited data availability. Therefore, for the latter scenario, it was opted to carefully evaluate the metrics for the main tasks, since the first objective is to minimize any performance gap anyway, then a qualitative analysis is carried out through the study of saliency maps.

$$\text{NME}(p, g) = \frac{1}{N} \sum_{i=1}^{N} \frac{\|p_i - g_i\|}{\text{IOD}(g)} \tag{2.14}$$

The most popular metric for the evaluation of facial landmarks detectors is the Normalized Mean Error (NME) (Equation (2.14)). It calculates the average Euclidean distance between the predicted facial landmarks and the ground truth annotations, normalized by a characteristic scale of the face, such as the interocular distance. Normalization is critical to fairly compare faces at different scaling factors, particularly for the 98-landmark scheme used is the interocular distance (IOD) obtained as the Euclidean distance between the pupil landmarks. Lower NME values indicate better accuracy in landmark localization.

$$\text{Accuracy}(y, \overline{y}) = \frac{1}{N} \sum_{i=0}^{N} 1(y_i = \overline{y}_i) \tag{2.15}$$

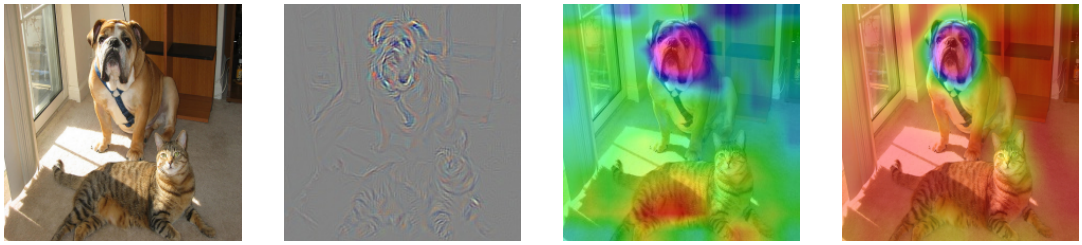
$1(\cdot)$ denotes the indicator function.

For the four binary classification sub-tasks for eyes (visibility and openness), Accuracy is calculated, defined (Equation (2.15)) as the ratio of correctly classified samples to total samples. For convenience of interpretation and display it is preferred to have a single value that summarize the performance of the eye related sub-tasks, the average of the 4 accuracy weighed against the support of each label is calculated. Similarly, for the multi-class mouth classification task the accuracy value is computed for each class and the weighted average is reported. For the head orientation tasks finally, for each degree of freedom

(yaw, pitch and roll), the circular offset (expressed in degrees) between the predicted and the ground-truth angle is computed. The three values are reported separately without averaging.

Saliency Maps

As for the distraction classification trained using the pseudo-annotated images, a validation set for calculating the metrics could not be obtained. In fact, evaluating the most desirable property of the model, the ability to generalize in the deployment scenario, would be hard to assess even if a validation set would be available. The straightforward qualitative testing campaign, conducted by experts, can provide a valuable feedback, but it is not an objective system of judgment. An important contribution to trying to evaluate predictions objectively comes from the field of Interpretable Machine-Learning (IML), which focus on elucidating the process behind a model's specific prediction. Unlike other classifiers (such as decision trees), the predictions provided by neural networks are not inherently interpretable and are generally considered to be a black-box predictor after training. Thus, a field of study of particular interest is that of attribution methods, and in particular the pixel attribution methods that aim at explaining the contribution of pixels (or group of pixels) for the classification results produced by a trained neural network. Typically, pixel attribution methods are used to produce a saliency map, which highlight regions of an input image that significantly contribute to the model's prediction, offering transparency into which features the network deems most relevant for classification.



(A) Input (B) Backprop (C) Grad-CAM (D) Score-CAM

FIGURE 2.17: Output of saliency methods for the class Mastiff.
Source [51]

One of the simplest, though scarcely reliable, methods of obtaining saliency maps given a trained convolutional neural network output O, an input image I and the target class i is to compute the gradient of the output for the target class with respect to the image, which can be achieved trough backpropagation.

$$\mathrm{SM}_{BP}(I) = \mathrm{ReLU}\left(\frac{\partial O_i}{\partial I}\right) \tag{2.16}$$

where the Rectified Linear Unit (ReLU) function sets negative values to zero and preserves positive values.

A more modern family of methods to produce saliency maps is the family of Class Activation Maps (CAM) methods, these so not directly provide information about positive or negative contributions of pixels but rather focuses on the importance of regions in the image. In the more general formulation (Equation (2.17)), these methods leverage an intermediate feature map F from the convolutional part of the CNN classifier, and defines the saliency map as a weighted sum of the k channels of F. The striking difference between two of the most popular CAM methods, respectively GradCAM [52] and ScoreCAM [53] is the way the feature maps weights α are defined. In the former, the gradient of O_i with respect to F^k, averaged across the spatial dimensions, is used.

$$\mathrm{SM}_{CAM} = \mathrm{ReLU}\left(\sum_k \alpha_k F^k\right) \tag{2.17}$$

ScoreCam instead ditch the computation of the gradient entirely, instead for each channel k the forward pass is computed using a linear combination of the input image I and the k-th channel of F. More precisely

$$\alpha^k = \mathrm{softmax}(C(F^k))_c \tag{2.18}$$

where c denotes the target class. The following are defined:

$$C(F^k, I) = \Psi(I \circ H^k)$$
$$H^k = s(Up(F^k))$$

where $s(\cdot)$ is a normalization function used to force the output values in $(0, 1)$ and $Up(\cdot)$ is the function used to Upsample the spatial dimension of F^k to match the size of the input image I.
Following a preliminary study, it was decided to use only ScoreCAM to derive a possible justification for the results of the qualitative analysis of the performance of the distraction classification. It must be said that these attribution methods are not a panacea, are not free of limitations and unresolved issues, and do not replace the use of validation metrics. Nevertheless, as reported in Section 2.4.3, the use of saliency maps provided valuable insights, which help to confirm qualitative observations.

2.4.2 Evaluation

For all experiments performed, the model was trained to process input images of (160×160) pixels. Training sessions were conducted using 2 Nvidia V100 GPUs, with a mini-batch size of 64 per GPU, totaling 128. A fixed training duration of 300 epochs was applied across all experiments. Additional hyperparameters and configuration specifics were tailored to the respective experiments and will be elaborated upon in dedicated sections as required.

Landmarks Regression Loss

The very first aspects to be evaluated are the definition of an appropriate loss for the landmarks regression task and an initial evaluation of the choice of optimizer. Therefore, we start for simplicity from a single-task configuration where only the landmarks regression task is trained, using only the images from the LaPa dataset, without including the pseudo-labeled images. Therefore, hereafter the options introduced in Section 2.3.4 are evaluated in a controlled setup, leveraging NME metric (Equation (2.14)) to compare the results.

In Table 2.3, a first subset of experiments is dedicated to compare the performance of baseline regression losses (L1, L2 and Smooth-L1) in this scenario. As can be appreciated, both L2 and Smooth-L1 ended up performing the worst; That these perform similarly is somewhat expected, since they behave identically in the range $[-1, 1]$, into which after the very early stages of training most of the errors fall. The fact that they perform so poorly can be traced back to the quadratic behavior, which ends up under-penalizing relatively small localization errors. The L1 in fact, not exhibiting this behavior, turns out to perform extremely competitively.

Loss	**Optim**	**LR**	**Landmarks**
L1	Adam	2^{-4}	2.4
L2	Adam	2^{-4}	3.67
Smooth-L1	Adam	2^{-4}	3.79
Wing	Adam	2^{-4}	**2.36**
Wing	SGD	4^{-4}	2.97
Wing	SGD+SWA	4^{-4} (2^{-4})	**2.36**

TABLE 2.3: Comparison of landmark losses and optimizers

For the second sub-set of this preliminary evaluation, the specialized Wing-Loss is evaluated with different optimizer configuration. With the baseline setup of optimizer and hyperparameters, the WingLoss outperforms all the other options. Alternatively, SGD with momentum set of 0.9 has been tested, this alternative turned out performing worse than Adam baseline; To this regard is in fact believed that Adam tends to reach sharp minima quickly, while SGD with momentum is slower and tends to reach broader minima, which, however, tend to be farther from the optimum. The best of both words can be obtained by leveraging the SWA technique introduced in Figure 2.16, in the used configuration the baseline learning rate ($4e-4$) is decayed to $2e-4$ starting from epoch $\text{swa}_{start} = 250$ of 300 over 10 epochs· Starting from swa_{start} till the end of training, a snapshot of the weights is collected at the end of each epoch weights, which are averaged at the end of training. With this technique, the NME value obtained with Adam is again reached, the difference (unfortunately difficult to measure), is that in theory a larger minimum is reached, with a benefit in terms of generalization ability.

In conclusion, WingLoss was obviously opted for, and it will be used for all future experiments in multi-task configuration. Regarding the optimizer, on the other hand, albeit not clearly appreciable at this stage, the use of SGD with SWA proved to be crucial when the distraction classification task is introduced (Section 2.4.3).

Multi-Task Regression on LaPa

It has been mentioned several times how training with a multi-task setup could tend to improve the performance of individual tasks compared to the single-task case, in this section therefore is devoted to evaluate and weigh this statement. To this end, the following strategy is proposed: for each of the base LaPa tasks (for whose validation data is available), a pair of models are trained, one with only the target task (single-task) and the other adding the landmarks regression task. Moreover, as reported in Table 2.4 the configuration with all tasks is trained and evaluated, and for completeness also a model with all tasks except landmarks regression is tested. All the experiments are performed using Adam optimizer and leveraging the WingLoss for the landmarks regression, for all the other tasks the loss configurations detailed in Section 2.3.4 is used.

Task	**Lm (↓)**	**Eyes (↑)**	**Mouth (↑)**	**Head (YPR) (↓)**
Baseline	2.36	-	-	-
Lm + Eyes	2.357	**0.975***	-	
Eyes (Single)	-	0.98	-	-
Lm + Mouth	2.44	-	0.934*	-
Mouth (Single)	-	-	0.917	-
Lm + Head	2.4	-	-	(2.637, 2.546, **2.972***)
Head (Single)	-	-	-	(**2.425***, **2.29***, 3.53)
Lm + All	**2.35***	0.978*	**0.935***	(2.622*, 2.793*, 3.186*)
All	-	0.98	0.9225	(3.051, 3.762, 3.327)

TABLE 2.4: Ablation study. The results in **bold** are the overall best score obtained, the use of (*) denote the best results between the pair of configurations under evaluation.

A first observation that can be derived from these results is that a single-task configuration never outperforms the dual-task equivalent, with the partial exception of the head-pose estimation obtains a slightly lower error on the regression of Yaw and Pitch angles, while severely underperforming in the Roll estimation. In contrast, the dual-task configuration with the 'addition of landmarks estimation has a more consistent trend in the error obtained on the 3 corners, which is indicative of a greater robustness of the feature set extracted from the convolutional backbone.

The configuration with all tasks active results in the absolute best performance for landmarks regression and mouth opening classification, and it is interesting

to note how this consistently outperforms the counterpart trained without the landmarks regression. Intuitively, this may be justified by the fact with the regression of landmarks in particular incentivizes the search for a rich feature set that captures the structure of the face and connotations. Training with a single task o the other hand, might lead to the learning of an overspecialized feature set which could capture unintended biases and nuances of the training images, leading to overfitting. In conclusion then, the initial hypothesis was confirmed, the good performance achievable in LaPa tasks was observed, and a baseline was also established with which to compare the trained models by introducing distraction detection. In the next section for that matter, this very aspect is investigated.

2.4.3 Qualitative Analysis

Since there are insufficient images available for the distraction classification task to define a test set for calculating the metrics, a qualitative analysis is presented in this section. In particular, a small number of highly representative images were used, some of which are shown in Figure 2.18. Specifically, for each of the 2 added classes (cell phone use and smoking), the typical situations in which false positives are evidenced were captured; in particular, bringing the fingers close to the mouth can trigger cigarette detection, while bringing the hand close to the face or ear can cause cell phone detection. Sample images where the action was actually performed were also used, but these proved to be less important, because the main problem highlighted is that of false positives.

FIGURE 2.18: Some samples used for qualitative analysis. The first five images for each class are special cases selected to trigger false positives.

The useful tool to enrich and justify the results of qualitative analysis is the study of Saliency Maps; In particular, the Score Cam method introduced in Section 2.4.1 is used, the target class for ScoreCam is defined as the class that is intended to trigger, therefore c in Equation (2.18) is set to the label index of the trigger class, not the ground-truth class (that would be the normal class for false positive inducing images), nor the predicted class. The rationale behind this choice is to elucidate which regions of the image are investigated in the prediction of classes of interest. Specifically, this analysis aims to justify the choices discussed in Section 2.3.4 regarding the choice of objective function, class balancing mechanisms, and the choice of optimizer and related strategies.

Configuration	Landmarks	Eyes	Mouth	Head
CE+Adam	3.14	0.9743	0.925	(3.37, 2.76, 3.148)
CE+SGD	2.77	0.97	0.925	(2.634,2.17,2.948)
FL+SGD	2.63	0.976	0.933	(2.542,2.157,2.781)
FL+SGD+W1	2.515	0.977	0.926	(2.473, 1.938,2.714)
FL+SGD+W2	2.518	0.978	0.927	(2.30, 1.966, 2.773)

TABLE 2.5: Results on Base tasks of models trained with distraction classification

Choice of Optimizer

The first comparison concerns the choice of the optimizer, from a preliminary stage of this work it has emerged that this aspect is critical to the generalization ability of the model trained with the addition of the distraction classification task. In order to clarify this aspect, two versions of the model trained using the same setup, with the cross-entropy objective function without class weights, were compared using the Adam optimizer or SGD with moment together with the SWA strategy, respectively. Only the optimizer-related hyperparameters differ; for Aadam a learning-rate of 2^{-4} was maintained, while for SGD+SWA it starts from 4^{-4} which is then linearly decayed in 10 epochs to 2^{-4} starting at epoch 250 of 300 total. Moreover, a momentum value of 0.9 has been used for SGD.

FIGURE 2.19: Cross Entropy loss with Adam

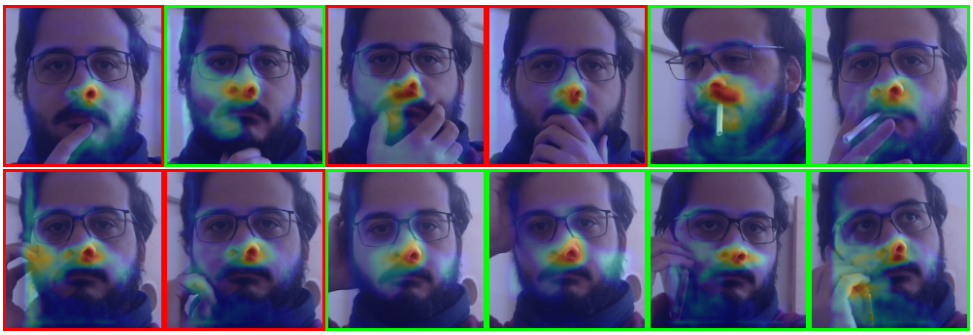


FIGURE 2.20: Cross Entropy loss with SGD+SWA

From a purely qualitative point of view, it is observed that the model trained with Adam is extremely sensitive to false positives, even if only by simply placing the hands near the face. This aspect is not fully appreciable from the images in Figure 2.19 and Figure 2.20, from where we see instead that the saliency maps produced with the model trained with Adam are much more sparse and hazy, likely an indication that the feature maps obtained are unrepresentative. The results in Table 2.5, on the other hand, show how there is also substantial performance degradation on basic tasks, likely attributable to a negative-transfer phenomenon. Consequently, it was chosen to continue exclusively using SGD com momentum and SWA optimizer combination.

Choice of Objective Function

The second aspect to be clarified concerns the choice of objective function; In Section 2.3.4 the Focal loss function was considered as a potential alternative to Cross Entropy. This allows reducing the contribution of easy-to-classify examples, theoretically incentivizing the model to focus on hard-to-classify samples. Two identical models were then trained by varying only the objective function for distraction classification; the gamma hyper parameter for focal loss (Equation (2.12)) was set to $\gamma = 2$

FIGURE 2.21: Focal loss with SGD+SWA

Qualitatively, it was observed an increase in robustness to false positives; Comparing the saliency maps of the model trained with focal loss in Figure 2.21 with the analogues using Cross Entropy Figure 2.20, it is observed that especially for the cell phone use class, more emphasis is placed on features in the correct region near the ear. Also, the numerical results on the base tasks go in favor of focal loss, with a slight overall improvement.

Class Weighting

In the end, it is important to clarify the controversial aspect introduced in (), which would see counterintuitively as a preferable option to underweight the minority classes (cell phone and cigarette) in order to mitigate overfitting on the peculiarities of the training images improve generalization altogether. For this purpose, two equal models were trained again, using focal loss and SGD optimizer with SWA. In the first case (referred to as **W1**), the weights chosen for the 3 classes (normal, phone, cigarette) are $(1.0, 0.5, 0.5)$, respectively, thus

going to penalize the minority classes. On the contrary, in the second experiment, (referred to as **W2**), the weights are instead $(0.5, 1.0, 1.0)$.

FIGURE 2.22: Focal loss with Minority class down weighting (W1)

FIGURE 2.23: Focal loss wit Majority class down weighting (W2)

W1 was by far the most stable and robust to false positives configuration, and it is in conclusion possible to consider it as the final release version of the developed DMS multi-task network. The saliency maps in Figure 2.22. The saliency maps in Figure 2.22 appear highly consistent with the objective of the classification, while for W2 they appear poorly correlated (Figure 2.23). In contrast, the numerical results on the basic tasks (Table 2.5) are relatively on par.

2.4.4 Deployment

To conclude the experimental section, it is essential to evaluate and compare the performance in inference obtained in a realistic deployment scenario. First, the performance of the trained model as the architectural configuration employed changes is analyzed; As introduced in Section 2.3.3, three possible configurations of the mobilenet-style backbone have been considered (`tiny`, `small`, `large`), with increasing computational demands. All the previous experiments have been conducted with the `small` configuration, therefore hereafter in Table 2.6 the results obtained on the LaPa baseline tasks are compared across the three configurations. All the models are trained with the configuration of "Lm+All" from Table 2.4, using the Adam optimizer with fixed learning-rate (2^{-4}) and the Wing-Loss for the landmarks regression.

Model Configurations

Model	Params	Landmarks	Eyes	Mouth	Head
Tiny	124,566	3.815	0.955	0.899	(3.855, 2.975, 3.604)
Small	705,422	2.35	0.978	0.935	(2.622, 2.793, 3.186)
Large	2,330,318	2.163	0.983	0.983	(2.078, 2.323,2.458)

TABLE 2.6: Comparison of model configurations

As can be appreciated, performance scales proportionally with the number of configuration parameters used; moreover, the observed variation is uniform across tasks. This is not obvious, in fact a drastic reduction in model footprint can often completely obliterate learning capabilities, requiring at the very least careful fine-tuning of hyperparameters or the use of specialized techniques such as distillation [54].

Inference Performances

Estimating and comparing the cost of inference is a nontrivial problem. Only measuring inference times is influenced by the particular choice of hardware and software, which by design and implementation characteristics may favor a particular architecture, but by varying the inference platform the conclusions can be subverted. Unlike in the next chapter, where it is necessary to compare profoundly different backbones, here the problem is partially mitigated by comparing different configurations of the same architecture.

Regarding the computing platform, there are several aspects to consider. With regard to server-side training and inference, the de facto standard today is the use of Graphics Processing Units (GPU). Unlike traditional CPUs, GPUs are specifically designed to handle vast amounts of data in parallel, making them exceptionally well-suited for the matrix and vector operations inherent to deep learning algorithms. The goal of this work, however, is inference on an embedded computing unit, without exchanging data with an inference server in any way, which must therefore burden only the local platform. The embedded platform must balance a trade-off between computational power, device power consumption, economic impact, and physical device footprint. There are a myriad of options in this regard based on CPUs, GPUs or increasingly on accelerators designed especially for deep model inference, an exhaustive exploration of the available options is beyond the scope of this thesis, however. Indeed, experimenting on different platforms requires becoming familiar with the hardware and software tools provided by device manufacturers; these are often proprietary tools and mastering them is extremely time-consuming.

Therefore, it was chosen to conduct this experimental phase based on the Jetson family devices produced by Nvidia. NVIDIA Jetson hardware represents a family of embedded computing platforms designed for high-performance AI and computer vision capabilities at the edge. The jetson platforms are designed

Feature	Jetson Nano	Jetson Xavier NX
GPU	Maxwell	Volta
CUDA Cores	128	384
CPU	4-core ARM Cortex-A57	8-core ARM Cortex-A57
CPU Clock Speed	1.43 GHz	1.9 GHz
RAM	4 GB LPDDR4	8 GB LPDDR4
Tensor Cores	No	Yes (48)
AI Performance	0.47 TOPS	21 TOPS
Power Consumption	5-10W	10-15W

TABLE 2.7: Comparison between NVIDIA Jetson Nano and Jetson Xavier NX

around the Nvidia Tegra System on Chip (SoC), which integrate a variety of processing units, including ARM CPU cores, NVIDIA GPU cores, and dedicated accelerators, into a single chip. This architecture is specifically designed to provide high-performance computing capabilities while maintaining low power consumption, making it well-suited for embedded and edge computing applications. In particular, the Nano and Xavier (NX version) platforms were used, the technical specifications of which are given in Table 2.7. From the software standpoint, Nvidia provides the JetPack Software Developer Kit (SDK), which includes the tools and libraries required to build and execute software for the Jetson platform. This includes the TensorRT inference engine, the proprietary runtime used for accelerating the execution of deep learning workload; The evaluated models are therefore trained in the regular setup and then imported in TensorRT for evaluation.

Model	Params	MACs	Nano (ms)	NX (ms)
Tiny	124,566	76.61 $\times 10^6$	5.583	1.382
Small	705,422	447.23 $\times 10^6$	14.743	2.32
Large	2,330,318	2.40 $\times 10^9$	76.451	8.296

TABLE 2.8: Evaluation on deployment

In Table 2.8 the inference time is reported for the three model configuration considered, the reported inference time (expressed in milliseconds) is the average execution time of 1000 inference steps. All the deployed model are built using the 16-bit floating point precision provided by TensorRT version 8.2, included in JetPack SDK version 4.6.1.
In addition to the measured inference times, it is useful to have a qualitative measure that can serve as a proxy of the computational requirements, agnostic to the specific hardware and software configuration used. Beyond the number of parameters, it is also reported the number of Multiply-Accumulate (MAC) (expressed in Tera Operation per Seconds (TOPs)), which quantify the number of multiplication and addition operations required to compute the output of a neural network layer. This value is directed correlated with the computational

workload and runtime requirements, and can be easily estimated as the sum of the MACs required by the individual operations of each layer.

Chapter 3

Advanced Driver Assistance System

3.1 Motivations

As anticipated, the work presented in this chapter is closely related to the work in the previous chapter on DMS and can largely be considered its complementary as far as this first part of the thesis work presented is concerned. This work focuses on Advanced Driving Assistance System (ADAS) systems, this umbrella term identifies a broad variety of automotive technologies designed to enhance driver safety, improve overall vehicle performance, and augment the driving experience through the integration of sensors, cameras, radar, and intelligent algorithms. In particular, by providing real-time assistance, ADAS mitigates the risk of accidents, reduces human errors, and paves the way for a safer and more efficient road transportation ecosystem. In essence, the DMS seen above is a special case of an ADAS system applied to the analysis of driver activity. In the more general sense, on the other hand, ADAS systems can be used to monitor various driving conditions and report any detectable dangerous situations that may arise from interactions with other vehicles or the surrounding environment. Popular applications include Forward Collision Warning (FCW), which aims at detecting an impending collision with a vehicle or obstacle in the vehicle's path and Blind Spot Detection (BSD), which monitor the vehicle's blind spots—areas not easily visible to the driver, helping prevent lane-change collisions.

It is easy to see how the implementation of these systems requires very precise perception of the surrounding environment, which may include knowing the relative position of other road users, knowing the position of the ego vehicle relative to the roadway, decoding road signs, and interpreting environmental conditions. More and more vehicles are being equipped with an increasing range of sensors, first and foremost radar, GPS, and in some cases LiDar, which are being used to implement ADAS features. Using specialized sensing equipment, however, is not always an optimal solution, primarily because of the costs that can greatly affect the final price of vehicles. The additional limitation, which is less often talked about, is the high degree of specialization of the information provided by this type of sensor: in fact, a radar, for example, is capable of providing an extremely precise measurement from the object placed in front of

the vehicle, but it is hardly usable for different applications such as accurate recognition of different vehicles or identification of road signs. Otherwise, building a vision-based perception system, attempting to mimic human perception at least in part, can be helpful in alleviating these issues, primarily the cost, which is orders of magnitude less than the alternatives.

Moving to an exclusively camera-based system, especially one relying on a monocular camera, introduces increased complexity on the algorithmic front, as it is necessary in this case to use computer vision techniques. In the landscape of recent deep learning-based methods, there are a wide variety of techniques that can be applied as fundamental building blocks to construct powerful and robust perception pipelines. Notably, there exists an extensive body of literature focusing on object detection, lane detection, and image classification tasks, offering a rich repertoire of methodologies to enhance visual understanding. However, state-of-the art deep learning models usually implies computational demands that are far beyond what can be obtained by onboard computing units. Therefore, a significant challenge lies in reconciling the ambition for advanced computer vision capabilities with the inherent limitations posed by the modest computational resources available on hardware realistically installable aboard motor vehicles. Another aspect to consider when building computer vision models for automotive perceptions is the availability of appropriate labeled dataset suitable for the intended task. Automakers are investing billions in acquiring training data through fleets of sensorized vehicles, meanwhile the research community instead is lagging behind for the lack of the availability of appropriate datasets available for research purposes. Therefore, while researchers are proposing increasingly advanced perception models, the scarcity of training and validation data poses a constraint on the possibilities of deploying these models in real use cases. In fact, training a model that can generalize to arbitrary and extremely varied driving scenarios is a problem that is poorly addressed, and often excellent results obtained on the little data available or on simulation environments are not reproducible in realistic scenarios.

As foreseeable, the development of camera-based ADAS systems is at the core of the investigation proposed: the work presented in this chapter stems from the requirements of the CEMP Project (Section 3.1.1), which focuses on the development of a new generation of assistance systems suitable for motorcycle applications. The need for deployment on a motorcycle further accentuated the mentioned issues, since the choice of computing hardware is further narrowed by the stringent weight, size, power consumption that are inherent in the intrinsic characteristics of this family of vehicles. the scarcity of data is also exacerbated, since there is no record of available datasets intended for motorcycle applications. Taking into account the requirements of the project, the fundamental vision tasks to be accomplished are Object Detection (in particular, detection of vehicles) and detection of lane lines and road boundaries. In addition, it is useful to also classify the occlusion state of the detectable vehicles (useful to identity the foreground vehicles) and to perform a full image classification, which is useful to calibrate the safety system to the current driving

scenario and environmental conditions. In this scenario, the advantages of a multi-task learning approach are expressed at the highest level; in fact, it would be almost impossible to think of implementing each of the necessary tasks and subtasks as a separate single-task neural network. It was therefore planned to group all vision tasks under a single perception neural network, combining a powerful but efficient shared backbone for feature extraction with specialized heads to produce the outputs needed for individual tasks.
This idea is not necessarily new, and has already been partly applied to similar use cases, however, the work presented is innovative in several respects: first, computational efficiency, and the ease of deployment on different embedded hardware platforms are fist-class citizens. Moreover, the robustness and ability to generalize to a broad variety of operational conditions is also a strong requirement. To achieve these ends, the models developed are all built from convolutional backbones, which do not require the implementation of exotic layers and are easily implemented on all desired hardware platforms. The heads for individual tasks, in stark contrast from what has been done by other works, are defined by exploiting a modern bottom-up formulation for both object detection and lane estimation tasks. Unlike other popularly used formulations, such as the famous anchor-base formulation for object detector that requires onerous and inefficient post-processing steps to obtain the desired predictions, the bottom-up formulation does not involve exoteric operations and the decoding is simple and efficient to implement. Second, exploiting a similar formulation for the two main tasks, although essentially different, simplifies the definition of objective functions and the balancing operation of individual contributions in the training phase, which is notoriously a nontrivial phase of optimizing multi-task models.

In summary, the proposed work is aimed at developing a new family of highly efficient and robust camera-based perception models intended for the development of ADAS. In the remainder of this chapter, the following contributions and results are introduced and discuss:

- The architectural choices, which reflects the requirement for the maximum computational efficiency.
- The proposed bottom-up representations for the tasks of object detection and lane estimation, together with the novel objective function proposed.
- Numerical experiments are presented, highlighting both the good performances on the intended task and the inference latency on the selected embedded hardware.

The remainder of this introductory section is therefore devoted to introduce the CEMP project, with set the requirements for the proposed work, which was ultimately deployed in the ADAS use cases required by the project. Part of the work presented in this chapter is derived from the original work [2] presented in October 2022 at the Workshop on "Perception and Navigation for Autonomous Robotics in Unstructured and Dynamic Environments" (PNARUDE)[1] of the

[1] https://iros2022-pnarude.github.io/

prestigious IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) conference.

3.1.1 The CEMP Project

The CEMP (Connected Electric Modular Powertrain) project was funded by the Lombardy Region under the 2014-2020 ESF Operational Program, as part of the investment aimed at strengthening research, development, and innovation. The project involved top industrial partners in the automotive industry, such as Dell'Orto, OCTO Telematics and Energica Motorcompany, along with the University of Modena and Reggio Emilia (UniMoRe). The latter has always been active in technology transfer projects to national and international automotive companies, industrial automation and high-performance embedded systems. In summary, the research objective of the project was focused on the development of a next-generation Electric Propulsion System intended for light 2-, 3-, and 4-wheeled vehicles (international category L), characterized by high energy efficiency. The prerogatives of the CEMP project include integration with high value-added connectivity and driver assistance systems aimed at reducing accidents and improving the efficiency of urban travel through vehicle sharing features.

FIGURE 3.1: Sensorized vehicle used for the CEMP project

In the development phase, project partners collaborated on the development of an electric-powered motorcycle with advanced connectivity and safety features. UniMoRe's involvement in CEMP has been primarily aimed at the development and testing of the active safety algorithms, which constitute the ADAS subsystem, or more properly ARAS (Advanced *Rider* Assistance System). The developed ARAS Package consists of three subcomponents: 1. Active speed limiter 2. Forward collision warning 3. Blind spot detection. The complete design and implementation of the ARAS system required successive stages of analysis, design, implementation, and testing, a detailed description of which is definitely beyond the scope of this thesis, therefore, only choices relevant to the presented work are given below. The FWC and BSM subsystems were implemented utilizing a vision based approach, so the vehicle was equipped with a pair of cameras (front and rear), the computation unit chosen is instead

based on the Xavier NX SoC, produced by NVidia. The chosen SoC is part of NVidia's Jetson family of embedded processors, equipped with GPUs and specially developed for running AI workloads, the Xavier NX it's a mid-range solution in the Jetson family, capable of 21 Tera Operation per Seconds (TOPs) with a peak power consumption of up to 20W. As for the ASL system, this was implemented using the GPS system with which the vehicle is equipped, so it is not further discussed.

3.2 Background

3.2.1 Object Detection

Object detection is a prominent computer vision task, in its general definition this task identifies the automatic extraction of regions of an input-supplied image that contain objects, typically limiting to one or more classes of interest. Typically, an object within an image is defined by means of a bounding box, i.e., a quadrilateral inscribing the object of interest, the bounding box can be encoded as a 4-uple of valued using several conventions, the main ones being the coordinates (x_1, y_1, x_2, y_2) of the top-left and bottom-right corners or as the center point coordinate (cx, xy) along with the box width and height (w, h). Thus, the object detection problem is reducible to regression from the image of a bounding box set $(b_1, .., b_n)$, each of which typically comes with a label c_i indicating its object class and often also a confidence score $s_i \in (0, 1)$. As many computer vision problems, object detection has existed since before deep learning models were introduced, but clearly today the latter dominate this field unchallenged.
Classical non-deep detectors, for which an in-depth discussion is beyond the scope of this work, were typically based on a two-stage approach (also common in the earliest deep learning detectors): first, a feature descriptor is computed for all the image, typically at varying scale factors, a prominent example being the Histogram of Oriented Gradients (HOG) features [55]. Then, a classifier like Support Vector Machines (SVM) [56] is used to classify the feature vectors in a sliding window fashion over the entire image. The sliding window moves across the image, and at each position, the features within the window are fed to the classifier, which is trained on positive samples (containing the object of interest) and negative samples (containing background or other non-object regions). Finally, to remove duplicate or highly overlapping detections, a post-processing step called non-maximum suppression (NMS) is often applied. While this family of approaches has demonstrated success for several years, it is not without limitations. Hand-crafted features like HOG rely on predefined rules for extraction, often failing to capture high-level semantic information effectively. Furthermore, the sliding window approach is highly inefficient, requiring multiple classifier inferences. Additionally, the fixed scales of the window pose challenges in detecting objects at various scales within an image.

Two-Stage detectors

deep learning based detectors, as mentioned above, have in a short time superseded all other approaches in every respect, and after around a decade of development, there is now a very wide choice of methodologies to suit virtually any application. The success of deep learning methods for object detection, as well as for all computer vision tasks, is mainly due to the use of convolutional networks, which are specifically designed for applications in this field. Casting object detection with a formulation that can be used for training a convolutional network is not straightforward and during the years a taxonomy

of approaches has emerged, although each method clearly has inherent characteristics that make it unique. The first and foremost branching of deep object detectors is between two-stage and single-stage methods, the former are the oldest approaches, originating from the typical formulations of traditional models, while the latter include the later more modern methods that, with different strategies, attempt to perform direct bounding-box regression from the image. The Region-Based CNN (R-CNN) [57] can be considered the progenitor of all the deep object detectors, this archaic two-stage detector leverage a region proposal module based on the Selective Search algorithm [58] to extract a set of 2000 candidate regions of the image that are likely to contain objects, then each region of the image associated with a proposal is cropped and warped to a square of (227×227) pixels. The second stage leverages a pre-trained CNN to extract a 4096-dimensional feature vector for each of the 2000 patches, then an SVM (for each class) uses the extracted features to classify the presence of the object within that candidate region proposal. In addition, a separate bounding-box regressor outputs four offsets value for the proposed bounding box, that should improve the precision of the detector. To train the second stage, first the convolutional feature extractor is pre-trained in 1000-classes classification dataset (ImageNet) [59], and fine-tuned in the domain-specific task to classify the extracted regions from the detection dataset. Finally, for each object class the SVM is trained for binary classification, using the ground-truth bounding boxes as positive examples and randomly samples regions as negative (actually the definition of negatives is slightly more complex). Clearly, this approach has a long list of anachronisms mutated from classical methods, one of the prominent limitations being the need to have to perform feature extraction with CNN separately for each of the proposed 2000 regions, which makes detection extremely inefficient. A subsequent work, Fast R-CNN [60], introduces several improvements to this framework: first, while the region proposals are still obtained separately from with the selective search, a single global feature map is extracted from the whole image, and the Region of Interest Pooling (RoI Pooling) method is introduced to obtain fixed-size feature vectors for each proposal from the shared feature map. From each RoI feature vector, additional layers are applied to perform classification and regression of the refined box coordinates, unlike previous work, the classifier and regressor are joined in a deep network and are trained end-to-end. Faster R-CNN [61] introduces further improvements, the first among them being the introduction of the Region Proposal Network (RPN), used to replace the slow and inefficient selective search algorithm used to produce the object proposals. The RPN generates region proposals directly from the shared feature maps, these proposals are then processed by the object detection network for classification and bounding box refinement. Faster R-CNN unifies the training of both the RPN and the object detection network. The model is trained end-to-end, making the entire process more efficient.

Single-Stage Detectors and Beyond

The other main branch is single-stage object detector, as the name suggests the main characteristics of this class of models is that they do not rely on an

explicit region proposal stage, rather all the bounding boxes are regressed in a forward pass. In the early years these models emerged because of their absolute computational superiority, since it typically results an order of magnitude lower than the two-stage models, but on the other hand they have long been considered less performant on the task at hand. Today, the performance gap has been abundantly closed, and single-stage detectors achieve state-of-the-art performance, and in addition, detectors with extremely low execution cost have been developed that can also be used for real-time applications on embedded devices. The YOLO (You only look once) family of single-stage models, although not the first incarnation of this approach, is certainly its best-known exponent. These models, specifically fall into the sub-class of anchor-based detectors, as seen for R-CNNs, yolo over the years has also undergone revisions and improvements, while maintaining the same basic pattern. In the most popular version [62], yolo operates as a single convolutional neural network that simultaneously produces several outputs: the input image is associated with an output grid of $(N \times N)$, with each grid cell being responsible for predicting bounding boxes and class probabilities for objects that fall within it. Instead of directly regressing the bounding box coordinates, the box center (c_c, c_y) is regressed as two offsets vector (t_x, t_y) with respect to the cell's center, and the box width and height (w, h) is regressed as two n_a offsets vectors (t_w^i, t_h^i) for $i = (1, ...n_a)$ with respect to a set of n_a anchor boxes. The anchor boxes introduce a prior for the shape of the bounding boxes, and are usually defined by performing k-means clustering on the set of ground-truth boxes. In addition to the 4 box offsets, for each cell, an 'objectness' score is predicted, which encodes the confidence that exists a corresponding object, together with a C-classes classification vector. Most implementations produce multiple similar outputs at different output scales (usually 3), which leads to an overall output of size $3 \times [(N_s \times N_s) * (1 + C + 2 + 2n_a)]$ for $s = (1, .., 3)$. A large number of candidate bounding boxes are therefore produced in a single inference step, however, only a fraction of these actually correspond to valid predictions. Thus, a complicated process is required to obtain the desired predictions, first to obtain the actual coordinates from the regressed offsets and anchors used, and then it is necessary to apply the NMS algorithm or remove duplicate or highly overlapping detections.

A more modern sub-family of single-stage detectors is that of anchor-free models, those eliminate the reliance on predefined anchor boxes, introducing simplicity and flexibility to the object detection process. This departure from anchor boxes brings advantages such as adaptability to diverse object shapes, improved efficiency, and a more straightforward training pipeline. A prominent example of anchor-free detector is CenterNet [63], on which the object detection branch of the multi-task network proposed in this chapter is based. CenterNet, operate the center-ness of a candidate point, indicating how likely it is to be the center of an object. In summary, Centernet-style detectors directly localize objects by predicting the coordinates of the bounding box's center, in addition, they incorporate a regression head to predict the width and height of the bounding box directly from the features associated with the predicted center. This is in

contrast to anchor-based detectors, where bounding box dimensions are often determined by anchor box templates. In Section 3.3.1 an in-depth description of this strategy, which has been extensively reposted for this work, is detailed.

3.2.2 Lane Estimation

Lane estimation, is a computer vision task that involves predicting the lane markings on a road from an image, or even a sequence of images. Like for object detection, this task was already being handled by traditional methods before the advent of neural networks, again, the state-of-the-art methods are entirely deep methods. As one can imagine, however, it is a more niche task, since it is basically relegated to the automotive domain and in particular applications related to autonomous driving. The formalization of the regression problem is not unique, since there is no universally accepted representation for lines (as there are bounding boxes for object detection).
A typical pre-deep learning approach is one based on the Hough [64] transform, which assumes that every line is representable by the equation of a straight line, an option that is obviously not always verified when thinking about the typical geometry of a road. A classic lane detection pipeline consists of an initial edge detection phase, which can be implemented with Canny's algorithm [65] or similar, then the Hough transform is applied to identify candidate lines in the edge-detected image. The Hough Transform transforms the image space into a parameter space, where each point in the parameter space corresponds to a possible line in the image. By applying the Hough Transform, straight lines in the image are detected as peaks in the parameter space. Post-processing steps involve filtering and clustering these lines to differentiate between left and right lane boundaries. While the Hough Transform is effective for detecting straight lanes, its limitations become apparent in scenarios involving curved or discontinuous lane markings, where more advanced techniques are often required for robust and accurate lane detection.

Regarding deep learning-based models, a macroscopic distinction is between segmentation-based and detection-based approaches. In segmentation-based methods [66]–[68], a Convolutional Neural Network (CNN) can be trained to classify each pixel in an image as either belonging to a lane or background. The output is a pixel-wise segmentation map, where each pixel is labeled as part of a lane or not, thus providing a detailed understanding of lane boundaries. If segmentation is at the semantic level, then it will only be possible to indicate which pixels are part of a line, possibly by specifying its class (e.g., continuous white line, double white line...), but it will not be possible to distinguish between points belonging to different lines of the same class, which would require instance-level segmentation [68]. Segmentation-based models heavily rely on clear and distinguishable lane markings, and may struggle in complex scenarios where lane markings are ambiguous, faded, or occluded or in environments where lanes are not marked or roads lack standard lane markings (e.g, lane boundaries or curbs). Moreover, high-resolution segmentation maps can

be computationally intensive, making real-time implementation challenging, especially in resource-constrained environments like embedded systems in vehicles.

In contrast, detection-based models, similarly to object detection, attempt to model lane detection as a regression problem of specific geometric primitives that model the shape and position of the lanes within the image. In perfect analogy, again a prominent sub-family is that of anchor-based detectors, among these Line-CNN [69] represents pioneering work. The used method is based on Faster-RCNN: first, a ResNet-family convolutional backbone is used to compute a shared feature map, then a Line-Proposal Unit (LPU) is introduced to produce a set of proposals for the detected lanes at predetermined locations on the feature map. More in detail, only the bottom and the top/left boundaries of the feature map are leveraged by the RPU, at each location a set of k lane proposed are produced with respect to a set of k anchor boxes. Each prediction consists of a binary classification score for the presence of an anchor line and a set of S offsets from the ground-truth line to the anchor line, evaluated at a set of S row anchors. In detail, anchor lines are defined as a ray emitted from the considered boundary location with a specific angle, again, the k-means algorithm applied to ground-truth annotations can be used to generate the set of anchors. This type of approach has been taken up and improved in subsequent contributions [70]–[73], however, in general they suffer from some limitations, the most obvious being the large set of candidate detections that are output, which require the NMS refinement in order to obtain the final set of precision. On the same basis as center-net style object detectors, some anchor-free lane detection methods set the task as a direct coordinate regression problem. In particular, each lane instance can be represented as a set of keypoints sampled along the curve, therefore a heatmap-based method for keypoints regression, as detailed in Section 3.3.1, is an effective strategy on which to build an anchor-free detector. The problem with direct keypoint regression is that, in some ways analogous to segmentation, it is not obvious how to group the predicted keypoints in the individual lane instances. To this end, the flexible bottom-up structure of heatmap-based keypoint regression is optimal for associating additional outputs with each keypoint to simplify the association phase. Notably, PINet [74] predict a feature embedding for each keypoint, using a particular loss to encourage embeddings associated with keypoints on the same line to be close together in embedding space. The use of embeddings is optimal in inference to perform clustering, but as also experienced in the early stages of the proposed work, their training is problematic and often creates instability. Inspired by the domain of human pose estimation, which heavily exploits heatmap-based regression, FOLOLane [75] and GANet [76], to perform association by predicting a spatial offset at each keypoint location, which can be effectively optimized with a standard regression loss. In particular, FOLOLane associates each keypoint with the displacement vector with respect to the immediate next point in the sequence, while GANet instead simply associates the offset with respect to the starting point of the sequence. Both of these choices are effective and clearly interpretable, but the GANet choice tends to be more robust to errors in decoding: for this region, this encoding has been adopted for the proposed

work, with the difference that the offsets are computed with respect to the midpoint. In Section 3.3.1 the details of this type of approach on which this work was based are given.

3.2.3 Multi-Task Approaches

In the literature, to the best of available knowledge, there is no model exactly comparable with the one proposed, which is understandable given that this one was created specifically for an extremely specific design requirement. There are a few contributions that are similar enough, however, to be useful in making interesting observations about the diversity in possible solutions for the development of a multi-task model of perception specifically for automotive applications.

Model	**Tasks**	**Dataset**	**Backbone**
MultiNet [77]	[DET, DA, SCN]	KITTI [78]	VGG16[41]/ResNet[79]
DLT-Net [80]	[DET, DA, LL]	BDD [81]	VGG16 [41]
YOLOP v1 [82]	[DET, DA, LL]	BDD	CSP-Darknet53 [83]
YOLOP v2 [84]	[DET, DA, LL]	BDD	E-ELAN [85]
HybridNets [86]	[DET, DA, LL]	BDD	EfficientNet-B3 [87]
A-YOLOM [88]	[DET, DA, LL]	BDD	CSP-Darknet53 [83]
TwinLiteNet [89]	[DA, LL]	BDD	ESPNet-C [90]

TABLE 3.1: Comparison of Multi-Task models. Tasks: Traffic Object Detection (**DET**), Drivable Area Segmentation (**DA**), Lane Line Estimation (**LL**), Scene Sclassification (**SCN**).

An overview of the identified contributions is presented in the Table 3.1, all of which are characterized by a basically similar structure, consisting of a convolutional feature extractor obtained from existing classification (VGG, Resnet or EfficientNet), object detection (Darknet, Extended-Elan) or segmentation (EspNet) models. On top of the convolutional backbone, all contributions analyzed except for MultiNet (which is the oldest), all models use a fusion strategy to combine feature maps at different scales to produce aggregate feature maps that contain information at different scales. For this component (which can be called a neck block), DLT-Net utilize the Feature Pyramid Network (FPN) [91], YOLOP (v1 and v2) combines Spatial Pooling Pyramid (SPP) [92] and FPN, Hybridnets adopts the Bi-Directional FPN (BiFPN [93]) and TwinLiteNet utilizes the Dual Attention Module [94]. As for A-YOLOM, their choice is unclear, because they claim to use a Spatial Pyramid Pooling Fusion (SPPF) which they assume is superior to the classical SPP, but without providing a reference or further details.
Regarding the solved tasks, all the compared approaches include the Driveable Area Segmentation task, by which is meant the segmentation (thus pixel-level classification), of the image in order to distinguish areas that are considered drivable or navigable from those that are not. In the work proposed in this chapter, this task was not addressed, because it is not of direct use for the purposes

of the developed ARAS module. For the main remaining tasks (Traffic Object Detection and Lane Line Estimation), they all follow very similar approaches in particular for the object detection task, all the contributions that provide for it use a single-stage anchor-based detection head, among them YOLOP, A-YOLOM and HybridNets use YOLO, while DLT-Net soi differs slightly by using RetinaNet [48], also single-stage anchor based. For lane estimation, on the other hand, all use a segmentation-based approach, with all the limitations of the case that were presented in the previous section.

As this concise review of the literature shows, to date, no multi-task perception models using anchor-free keypoint-based approaches have yet been proposed for object and lane detection tasks. The remainder of this chapter therefore presents work that, among other objectives related to design requirements, aims to fill this gap.

3.3 Proposed Approach

3.3.1 Problem Formulation

Object Detection

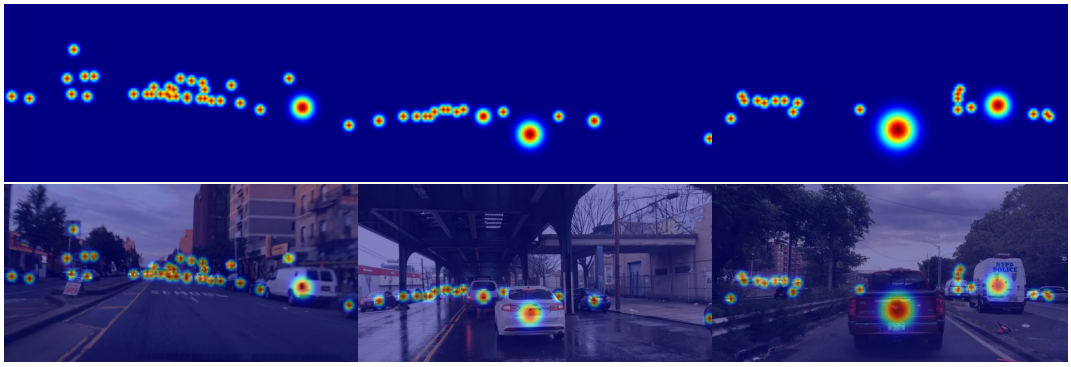


FIGURE 3.2: Sample heatmaps for Object Center. Top: Heatmap (all classes merged), Bottom: Heatmap superimposed to input image

The desired output for the object detection task is a set of N_O detections in the form $(x_1, y_1, x_2, y_2, c, o)$ where x_1, y_1 (resp. x_2, y_2) define the image space coordinate of the top-left (resp. bottom-right) corner of the bounding box, c characterizes the object class and o is the binary label for the object's occlusion state. The bottom-up approach leveraged is heavily based on the anchor-free detector described in CenterNet ([63]). The idea behind the anchor-free detector is to represent each bounding box as the coordinates of its center point and two offset vectors from the center to the top-left and bottom right corners of the box, the center points and the offsets are regressed from the image with a fully convolutional approach, leveraging a heatmap-based encoding. For each object class $k \in C_O$ with $|C_O| = 10$, a ground truth heatmap is defined $H_D^k \in \left(\frac{w}{S} \times \frac{h}{S}\right)$ (with S being output stride) that encodes the center points of all the objects in I belonging to class k. Given a set of detection ground truths, first the target keypoints in output space $c^i = \left(\frac{x_2^i - x_1^i}{2S}, \frac{y_2^i - y_1^i}{2S}\right)$ for $i \in \{0, \ldots, N_O^k\}$ are computed as the geometrical center point of each box and rescaled with the output stride (rounding to the nearest integer), as in Figure 3.3 (left). Then the target heatmap H_D^k for the k-th object class is obtained by computing for each keypoint a Gaussian centered at the keypoint's location and taking the element-wise maximum:

$$H_D^k(x, y) = \max_j \left\{ \exp\left(-\frac{(x - c_x^j)^2 + (y - c_y^j)^2}{\sigma^2} \right) \right\} \quad \text{for } j \in \{0, \ldots, N_O^k\} \tag{3.1}$$

the variance term σ can be fixed to a constant value, or it can be computed for each object being encoded accordingly to the box shape as defined in [95].

The final ground-truth heatmap $H_D \in \left(|C_O| \times \frac{w}{S} \times \frac{h}{S}\right)$ is obtained by concatenating the single-class heatmaps along the class dimension, thus H_D represents the target for center point regression. In addition, a pair of offset maps, $\overline{O}_D \in \left(\frac{w}{S} \times \frac{h}{S} \times 4\right)$, is regressed. This encodes, at the center point coordinates of each ground-truth object, the pair of offset vectors from the object center to the top-left and bottom-right corners of the target bounding box (depicted in Figure 3.3 (right)). Unlike the center point heatmaps, there is no need to define a ground-truth offset map. During the training phase, the regression loss is evaluated only at the ground-truth center points. Importantly, this means that the values of $\overline{O}_D$ that do not correspond to any ground-truth center point are not explicitly forced to a default value (i.e., zero).

$$\overline{O}_D(c_x, c_y) = (c_x - x_1/S, c_y - y_1/S, c_x - x_2/S, c_y - y_2/S). \tag{3.2}$$

In addition, the occlusion state for each object detected is also regressed: for this scope, an occlusion map $\overline{V}_D = \left(\frac{w}{S} \times \frac{h}{S} \times 1\right)$ is also regressed, also in this case the loss function is evaluated only at the ground truth center points (c_x, c_y).

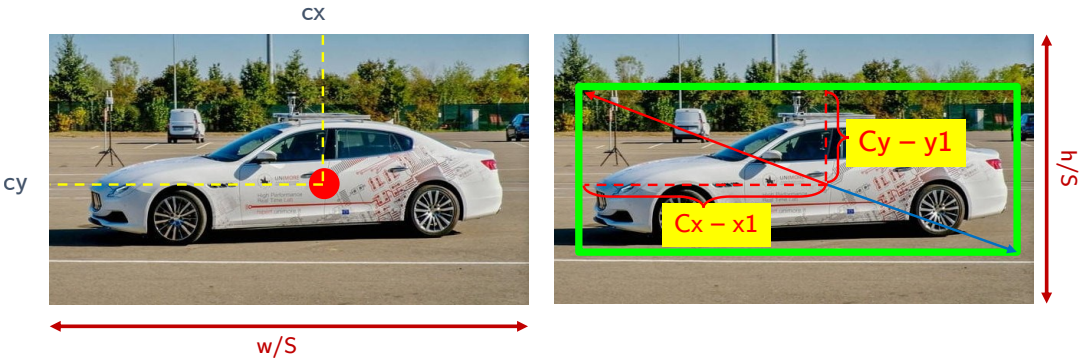


FIGURE 3.3: Definition of the object detection's targets: (left) object center-point (right) box-corners offsets: top-left (red) and bottom-right (blue).

At inference time therefore, the model outputs the predicted center point heatmaps $\overline{H}_D$, offset maps $\overline{O}_D$ and occlusion maps $\overline{V}_D$: with the chosen formulation, the process to recover the predicted bounding box is simple and efficient, and does not require refinement operations as the Non-Maxima suppression required by many detectors. First, the predicted center points $\overline{P}^k$ for each class k are retrieved by first taking the local maxima (the peaks of the Gaussians) from the keypoint heatmaps $\overline{P}^k = [(\overline{c}_x^1, \overline{c}_y^1)^k, ..., (\overline{c}_x^{\overline{N}_O^k}, \overline{c}_y^{\overline{N}_O^k})^k]$, then, for each candidate center point $(\overline{c}_x, \overline{c}_y)$ (the class index is omitted for ease of notation), the top-left and bottom-right offsets are retrieved by evaluating the predicted offset map at each candidate location:

$$\overline{O}_D(\overline{c}_x, \overline{c}_y) = (\overline{c}_x - \overline{x}_1/S, \overline{c}_y - \overline{y}_1/S, \overline{c}_x - \overline{x}_2/S, \overline{c}_y - \overline{y}_2/S) \tag{3.3}$$

Finally, the 4-uple $(\overline{x}_1, \overline{y}_1, \overline{x}_2, \overline{y}_2)$ that defines the predicted bounding box is easily obtained by summing the predicted offset to the recovered center points, and can then be denormalized to the input image size by accounting for the

stride factor. In addition, the predicted occlusion label is retrieved by evaluating the occlusion map at the center point location: $\overline{o} = \sigma(\overline{V}_D(\overline{c}_x, \overline{c}_y))$, where the function $\sigma(\cdot)$ (not to be confused with the Gaussian variance), defines the sigmoid function. The value $s \in (0, 1)$ of the Gaussian heatmap at $(\overline{c}_x, \overline{c}_y)$ also serves as a confidence score for the detected box, which is useful to implement a confidence threshold to be effectively set for detections during decoding.

Lane Estimation

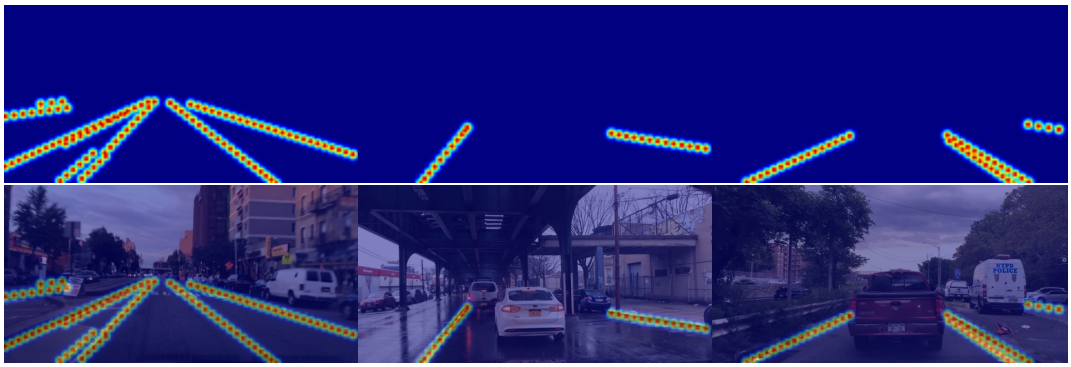


FIGURE 3.4: Sample heatmaps for Lane Estimation. Top: Heatmap (all classes merged), Bottom: Heatmap superimposed to input image

The desired output of the lane estimation task is a set of lane instances for each lane class l among the $N_l = 8$ possible ones. The generic i-th lane instance $L_i^l = [(x_1, y_1), ..., (x_{n_i^l}, y_{n_i^l})]$ is a list of keypoint coordinates of arbitrary length n_i^l (different for each lane instance). All the keypoints in L_i^l should belong to a single lane and can be easily used to fit a polynomial curve representing the lane line boundaries in image space coordinates. Therefore, the objective of regressing the lane instances is twofold: to identify the points in the image where the lines of the road are represented and to group the sets of points belonging to the same line into a single instance. Similarly to the object detection task, a keypoint-based approach, in particular the method, is derived from [76] and [74]. As shown in Figure 3.5 this method allow to simultaneously recover all the keypoints belonging to any lane visible for the l-th lane class and regressing an offset vector for each keypoint that is used to effectively cluster the keypoints in distinct lane instances. To begin, it is necessary to preprocess the annotations provided by the dataset into a compatible representation with the chosen formulation: in particular at the outset the available annotations encode each line as a Bézier curve, consequently the first step is to sample the provided curve with a fixed sampling step so that each line is encoded as a set of keypoints.

In analogy to the approach used for object detection, a ground-truth heatmap H_L^l is defined for each lane class l. This heatmap encodes the position of all visible keypoints belonging to class l, disregarding specific lane instances. The ground-truth heatmap is obtained by rescaling the keypoints' coordinates to the output space, accounting for the stride factor S, and then centering a Gaussian

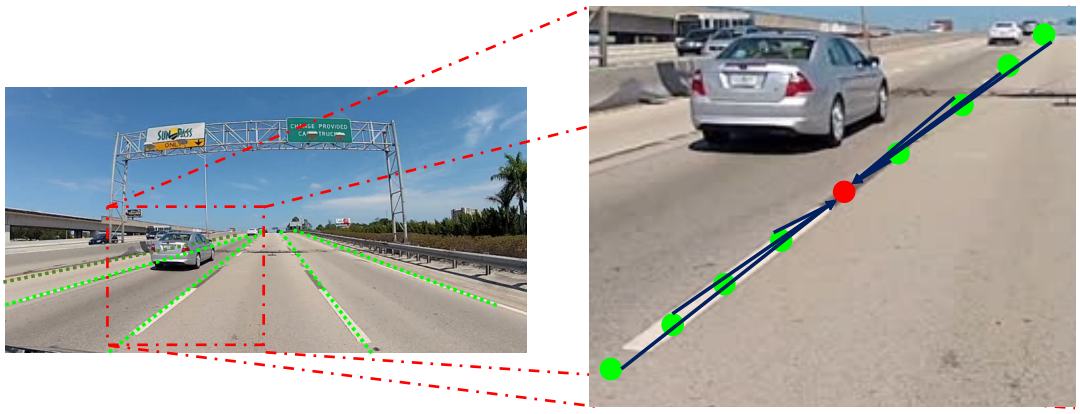


FIGURE 3.5: Representation of the lane estimation head used: each keypoint belonging to a line cast a vote offset for the mid-point of the lane that it belongs to.

at each lane keypoint, as described in Equation (3.1). The complete heatmap H_L is then obtained by concatenating the single-class heatmaps. The regression of the heatmaps allows for retrieving the positions of the keypoints belonging to the lines of each class, but it does not provide useful information for grouping the points into individual instances. To address this, in addition to the keypoint heatmaps, a single offset map $\overline{O}_L \in (\frac{w}{S} \times \frac{h}{S} \times 2)$ is also regressed. $\overline{O}_L$ is used to associate each keypoint with an offset vector pointing to the midpoint of the line to which it belongs. To achieve this, at each j-th keypoint location $p_j^i = (x_j^i, y_j^i)$ of the i-th lane instance, the offset vector to the midpoint is regressed, defined as $L_i[|L_i|/2] = (xm_i, ym_i)$ of the lane instance. The offset vector is regressed only at the location corresponding to the keypoint locations, as explicated in Equation (3.4).

$$\overline{O}_L(x_j^i, y_j^i) = (xm_i - x_j^i, ym_i - y_j^i) \tag{3.4}$$

At inference time, the model outputs the predicted keypoints heatmaps $\overline{H}_L$ and offset map $\overline{O}_L$, in order to decode the predicted lane instances, first for each lane class the candidate lane keypoints $[(x_1, y_1), ..., (x_{N_l}, y_{N_l})]$ are retrieved by selecting the Gaussian peaks from the predicted heatmap $\overline{H}_L^l$. Then, at each candidate keypoint location $(\overline{x}_i, \overline{y}_i)$ (omitting the class label l), the resulting candidate mid-point $(\overline{xm}^i, \overline{ym}^i)$ is retrieving by evaluating $\overline{H}_L$ at the keypoint's coordinates. Finally, after each point has cast a "vote" for it's mid-point, the candidate midpoints are used to cluster the keypoints in individual lane instances using an agglomerative clustering algorithm [96] with a fixed distance threshold.

3.3.2 Model Architecture

The proposed convolutional model follow a modular approach, that allows different options to be effectively explored to identify the solution best suited to the computational and performance requirements of the application. The general model architecture, schematized in Figure 3.6, follows a simple and robust approach common to several single-stage detector: first, the convolutional part

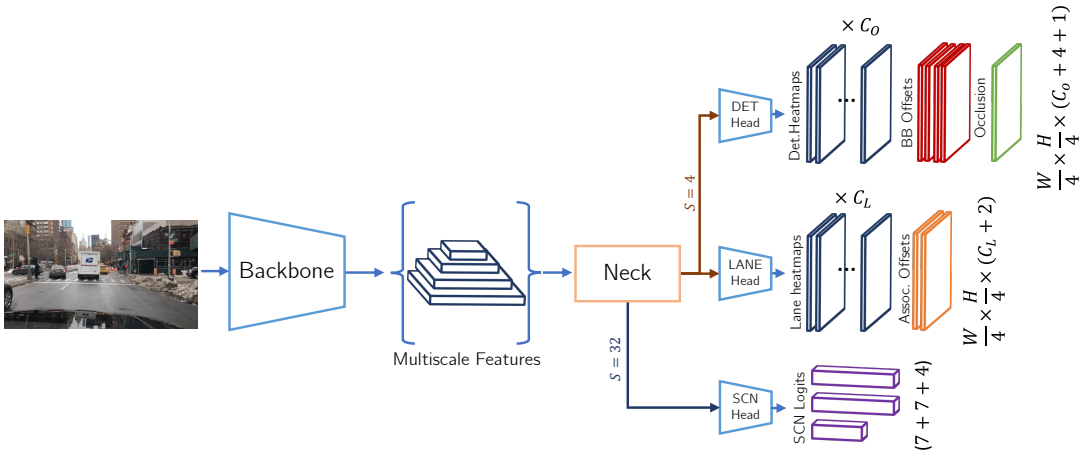


FIGURE 3.6: Overview of the proposed model. The outputs are produced by three distinct heads for object detection (DET), lane estimation (LANE) and scene classification (SCN).

of an off-the-shelf classification model is used to extract a set of n_F multiscale feature maps:

$$\mathcal{F}_i \in (C_i, w/S_i, h/S_i) \text{ for } i \in \{0, \dots, n_F\}$$

With C_i denoting the number of channels (increasing) and S_i denoting the output stride (also increasing). For all the evaluated model configurations, $n_F = 4$ output feature maps are always used, with fixed output strides of $(4, 8, 16, 32)$, while the number of output channels C_i is specific to the chosen backbone.

Subsequently, a *neck* block further process the feature maps, increasing the output resolution and (optionally) aggregating the multiscale features. The neck provide as output a small resolution feature map ($S = 32$), to be used exclusively for the image tagging task and a high resolution feature map ($S = 4$) to be used for both the object and lane detection tasks. Finally, a distinct *head* for each individual task apply a final convolutional layer in order to produce the task-specific output. This structure is therefore extremely flexible, allowing for experimenting with different backbones and different necks, as well as adding or removing individual tasks heads.

Backbone

In order to evaluate the flexibility of the proposed approach, three different model families are evaluated as convolutional backbones: 1. Resnet [79], with its simple and well understood architecture, represents the most conservative option 2. Mobilenet [46] instead represent a well established example of convolutional model with reduced memory and computational footprint 3. finally, EfficientNet [87] is a more recent example of a model with low inference cost, leveraging modern solutions.

The decision was made to purposely avoid tinkering with backbones that were either too computationally intensive to be effective in the intended deployment target or were built with uncommon layers and architectural choices [97], which could make the conversion to the inference framework more challenging and

potentially inaccurate. All the tested backbones had been pretrained on the ImageNet classification dataset.[59].

Necks

To process the feature maps obtained by the convolutional backbone, two different options for the implementation of the neck lock are evaluated.

The **simple** neck only takes the last feature map (stride 32) and gradually increase the spatial resolution up to a stride of 4, required for object and lane detection, with alternating layers of regular 2D Convolution and Transposed convolution with stride of 2 and kernel size of 4 (implying a 2-time upsampling factor for each layer). The feature map returned for the image tagging task is the output from the first convolutional layer. Furthermore, all the Transposed convolutional layers are initialized as bilinear interpolation kernels.

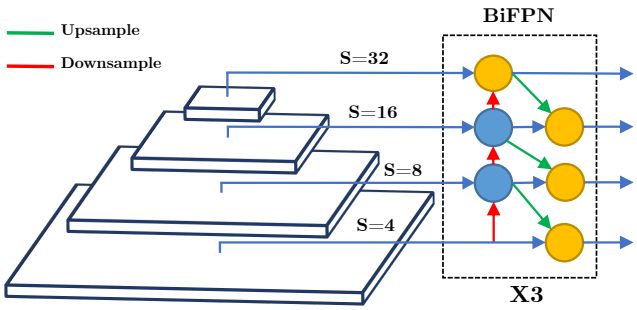


FIGURE 3.7: Overview of the BiFPN based head. In all the experiments, all the up sample operations are implemented as nearest neighbor interpolation and the down sample operations as MaxPooling layers.

The more sophisticated, yet lighter, neck implementation is based on the Bi-directional Feature Pyramid Network (**BiFPN**) introduced in [93]. This block is an evolution of the Feature Pyramid Layer (Fpn, [91]), it allows fusing feature maps ad different scales leveraging both top-down (up sampling) and a bottop-up (down sampling) paths to allow the information to flow in both directions, as depicted in Figure 3.7. Given a set of multiscale feature maps $\mathcal{F}_S$ with S denoting the stride, $(\mathcal{F}_4, \mathcal{F}_8, \mathcal{F}_{16}, \mathcal{F}_{32})$ the intermediate output for the top-down path is:

$$\mathcal{T}_S = \begin{cases} \mathcal{F}_S & \text{if } S = \min_S \\ Conv_{2D}(\mathcal{F}_S + up(\mathcal{F}_{S/2}) * w_{\mathcal{T}}^S) & \text{otherwise} \end{cases}$$

Then the bottop-up path is defined as follows:

$$\mathcal{B}_S = \begin{cases} \mathcal{T}_S & \text{if } S = \max_S \\ Conv_{2D}(\mathcal{T}_S + down(\mathcal{T}_{2S}) * w_{\mathcal{B}}^S) & \text{otherwise} \end{cases}$$

The trained weights $w_{\{\mathcal{B}|\mathcal{T}\}}^S$ are added to the feature fusion to allow the network to model the importance of each feature. The feature map $\mathcal{B}_4$ is used as input

for the object and lane detection tasks, and $\mathcal{B}_{32}$ for the task of image tagging, respectively.

Heads

The heads follow a minimalistic design aimed at keeping the complexity overhead of each individual task to a minimum. The head for the lane estimation and object detection head share also the same input features and an identical structure: each of the desired outputs (i.e, heatmaps, offsets etc. . .) is obtained by applying a dedicated 3×3 convolutional layer to the feature map, followed by a ReLU activation, followed by a final 1×1 convolution to produce the desired number of output channels. A sigmoid activation is applied only to the heatmap's output layer and, for training stability, has been proven beneficial to initialize the bias value of the last convolution to -4.6, since after the sigmoid layer ($\sigma(-4.6) = 0.01$) this leads to an initial estimate reasonably close to the average heatmaps value (which has a significant dominance of near-zero values). The image tagging head is as simple as it gets, a single 3×3 convolution is applied to the smallest feature map produced by the neck block, followed by activation and max-pooling, finally three fully-connected layer are applied to obtain the image level classification outputs.

3.4 Experiments and Evaluation

3.4.1 Experimental Setup

As anticipated, for this work it has been chosen to conduct the experimentation leveraging exclusively the dataset BDD100K [81], this publicly available dataset consists of 70K train images and 10K validation images sampled at 10Hz from a large set of 100K driving videos (around 40s each). Videos are captured in a wide range of scene types and conditions, allowing to train robust models able to generalize to real driving conditions. Because of the large differences in the type of annotations provided by the few similar datasets (e.g., OpenLane), experimenting on multiple datasets would result in an extremely onerous workload. The annotation provided by BDD100K for the object detection task already comes in the desired format, where each bounding box is represented in image space as the coordinates of top-left and bottom-right corners plus an object label class and the binary label for the occlusion state. Hence, the heatmap and offset targets can be easily computed as detailed in Section 3.3.1, for the object detection task the variance σ used for the encoding of the center-points heatmaps formalized in Equation (3.1), is computed as a function of the bounding box geometry. Thus, as can be appreciated in Figure 3.2, larger boxes (which typically correspond to closer objects), are encoded with a larger radius: as explained in [95], this choice aims at reducing the penalty given to negative locations around a positive location, which leads to a tangible performance benefit. Regarding the object classes, all the 10 labeled classes provided by the dataset are kept, which includes 6 vehicles classes (car, truck, bus, train, motorcycle, bicycle) and other 4 classes (pedestrian, rider, traffic light and traffic sign). In comparison, similar contribution instead ([82], [86]), collapses all the vehicles in a single super class.

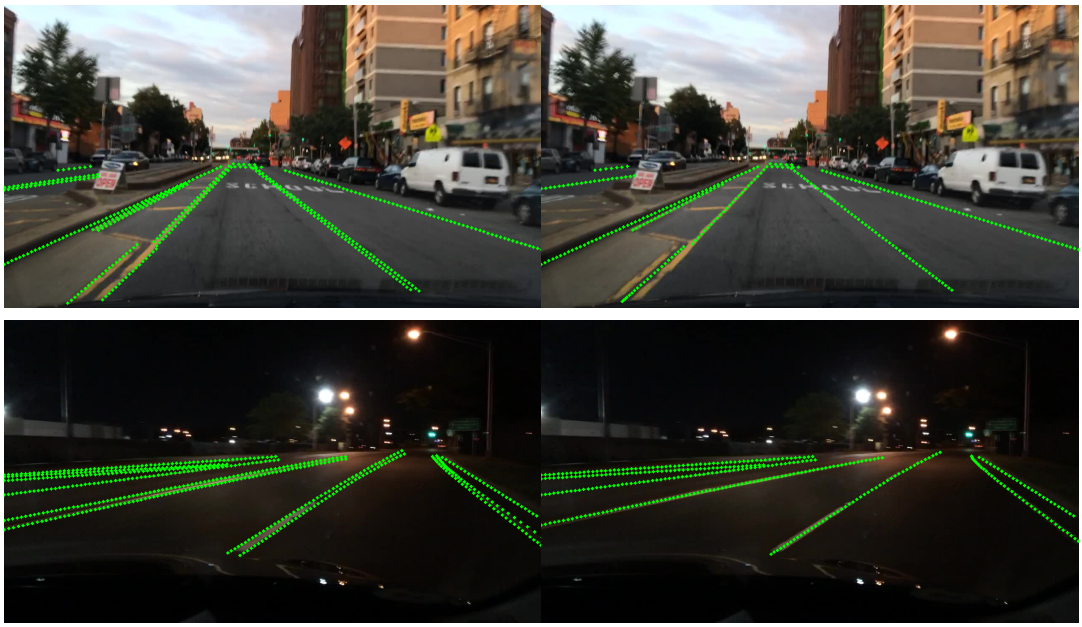


FIGURE 3.8: Labels for the lane lines. Before (left) and after (Right) pre-processing.

For the lane estimation task, some additional pre-processing is required: the dataset authors opted to label each lane with two distinct segments, denoting the two edges of each line (Figure 3.8 left). For the scope of this work it is preferable to regress a single segment in the center of each visible line, for this reason a deduplication strategy is proposed to merge the two distinct labels. First, a distance measure is calculated between all pairs of annotations of the same class: this measure is implemented by first sampling the Bézier curves defining each annotation so that it is representable as a list of points, after which Dynamic Time Warping (DTW) is used, which is useful for calculating the distance between unaligned sequences. Then, the calculated distance is used by comparing it with a threshold value to determine whether two annotations refer to the same line; if so, the alignment obtained via DTW is exploited to calculate the average between the two point sequences, which is then used as the new ground-truth lane annotation (Figure 3.8 right). After this pre-processing, which only needs to be done once, the annotations can be used to generate ground-truths as explained in Section 3.3.1, in this case, the radius of the Gaussians used to encode the coordinates of the keypoints is instead fixed as a constant $\sigma = 2$. Finally, for the image tagging task the dataset provides 7 weather classes (rainy, snowy, clear, overcast, undefined, partly cloudy, foggy), 7 scene classes (tunnel, residential, parking lot, undefined, city street, gas stations, highway) and 4 time-of-day labels (daytime, night,dawn/dusk, undefined).
To keep the computational demands low, the RGB images, with an original resolution of 1280×720 pixels, are first down scaled with a factor of 2 and then cropped to a final resolution of 640×320. The output stride $S = 4$ for all the experiments, therefore all the outputs have a spatial resolution of (180×90). During training several data augmentation strategies are adopted in order to improve generalization against changes in appearance and illumination (random noise, RGB shift, random brightness and contrast) and camera positioning (random affine transforms, random crop, random horizontal flips).

3.4.2 Objective Functions

Multi-task deep learning models are often non-trivial to train due to instability and some tasks dominating the gradient of the loss. Hence, careful weighting of the individual tasks and often elaborate multi-stage training strategies are commonly required. One advantage of the proposed formulation is that the formulation of the objective function for individual subtasks is simple, using mainly basic regression losses (except for the loss of heatmaps). Since the lane and object detection tasks share a very similar formulation, it is possible to the same losses for the heatmap and offset regression subtasks.

Heatmap Regression Loss

The heatmap regression sub-task is used to regress the keypoint coordinates of the boxes center-points in the object detection task and for regression of the lanes keypoints in the lane estimation task. Therefore, the very same formulation is used for both tasks: commonly used objective function for heatmaps regression include the standard $L1$ and MSE ($L2$) distance functions, other loss

function explicitly engineered to boost heatmap regression include the Wing Loss [17], [47]. Lately the Focal Loss introduced in [48] has been found to be useful for this matter [63], [95], [98], the focal loss adds a weighting term to the Cross-Entropy Loss (usually used in binary classification problems) to avoid the contribution from the large amount easy to classify values (i.e, background pixels) to dominate the gradient magnitude. During this experimental analysis, it was observed a severe training instability when using the Focal Loss (as well as with Generalized Focal Loss [99]), that led to an unusable model. Driven by a similar principle, a weighted version of $L2$ distance is proposed, whose formulation is reported in Equation (3.5)

$$\mathcal{L}_H(H, \overline{H}) = \frac{1}{N_K} \sum max\{(1+H)^{\alpha}, (1+\text{sg}(\overline{H}))^{\beta}\}(H-\overline{H})^2 \tag{3.5}$$

The term $max\{(1+H)^{\alpha}, (1+\overline{H})^{\beta}$ introduces an exponential weighting of the $L2$ loss. This means that locations where either the target heatmap or the predicted heatmap are closer to the maximum value of 1 (obtained at the Gaussian peaks) will have an exponentially larger relevance in the loss. More in detail, the term $(1+H)^{\alpha}$ introduces an exponential weight that focuses on hard values where a peak is present in the target heatmap. This allows the loss to prioritize locations with actual keypoints. Instead, the term $(1+\overline{H})^{\beta}$ is responsive to penalizing false positives in the predicted heatmap. It ensures that locations with high predicted values, even if not corresponding to keypoints, contribute significantly to the loss. For all the experiments, the values of the exponential weighting factors are experimentally set to $\alpha = 4$ and $\beta = 2$. The stop-gradient operator $\text{sg}(\cdot)$ is used to prevent the gradient from being propagated through the weighting term of the loss.

Offsets Regression Loss

As introduced in Section 3.3.1, the box-corner offsets required for the object detection task, as well as the mid-point association offsets of the lane estimation task are evaluated only at ground-truth keypoints locations. To keep the formulation simple, the $L1$ loss is used in both cases for the regression of offsets. Therefore, for the object detection task, let (cx_i, cy_i) being the center point coordinates for the i-th ground-truth bounding box and $(\delta_i^{x1}, \delta_i^{y1}, \delta_i^{x2}, \delta_i^{y1})$ being the corresponding top-left and bottom-right offsets, defined in Equation (3.3), the regression loss is computed as:

$$\begin{aligned} &\mathcal{L}_{OD}(\overline{O}_D, [(cx_1, cy_1), ..., (cx_n, cy_n)]) = \\ &= \frac{1}{n} \sum_{i=0}^{n} |\overline{O}_D(cx_i, cy_i) - (\delta_{x1}^i, \delta_{y1}^i, \delta_{x2}^i, \delta_{y2}^i)| \end{aligned} \tag{3.6}$$

where n, for ease of notation, define the numbers of ground-truth objects among all classes.

Similarly, let (mx_j, my_j) being the coordinates of the mid-point for the j-th lane instance and $(\gamma x_j^k, \gamma y_j^k)$ be the offsets from it's k-th keypoint, the corresponding regression loss is obtained as:

$$\begin{aligned} &\mathcal{L}_{OL}(\overline{O}_L, [(mx_1, my_1), ..., (mx_m, my_m)]) = \\ &= \frac{1}{m} \sum_{j=0}^{m} \sum_{k=0}^{k_j} |\overline{O}_L(mx_j, my_j) - (\gamma x_j^k, \gamma y_j^k)| \end{aligned} \tag{3.7}$$

where m denotes the number of individual lane instances and k_j denotes the number of keypoints that defines the j-th lane instance.
Alternative approaches, such as employing advanced metrics like Intersection over Union (IoU) [100] or exploring specialized loss functions like DIoU loss [101], have been suggested in the literature to enhance object detection performance. However, these alternatives have not been extensively explored within the scope of this study. Although these sophisticated objective functions could potentially be integrated into the adopted formulation, the complexity of the proposed multi-task scenario introduces challenges in managing and resolving potential convergence issues. Practical implementation and investigation of these alternative metrics and loss functions are left for future exploration due to the focus and constraints of the current work.

Classification Loss

the classification subtasks, which would be the occlusion classification in the object detection task, and the three scene classification tasks, use cross-entropy as the loss function. For the occlusion classification, the loss is evaluated only af the locations of the occlusion map $\overline{O}_V$ that corresponds to object keypoints, therefore, using the same notation introduced in the previous paragraph, the calculated loss is obtained as:

$$\mathcal{L}_{OV}(\overline{O}_V, [(cx_1, cy_1), ..., (cx_n, cy_n)]) = \frac{1}{n} \sum_{i=0}^{n} BCE(\overline{O}_V(cx_i, cy_i), v_i) \tag{3.8}$$

where v_i denote the ground-truth visibility label for the i-th object and BCE denotes The Binary Cross entropy objective function. Instead, for scene classification tasks, the standard cross-entropy with softmax used in multi-class classification problems is applied.

Multi-Task Loss

Summarizing what we have seen, the overall loss for the multi-task model consists of three main loss terms for the object detection, lane estimation and scene classification sub-tasks. In practice, each term consists of several subterms, which have been detailed in the previous paragraphs and for the sake of clarity are not repeated further. The interesting aspect to note is that normally in a problem of this type it is necessary to carefully balance the individual loss contributions with scaling terms, in this case this work proved straightforward.

This is likely due to the fact that the formulations are very similar across tasks, and share the same loss functions. Mainly, it was only necessary to rescale the overall loss tel scene classification task (sum of the 3 classification subtasks) by a value of 0.1.

3.4.3 Evaluation

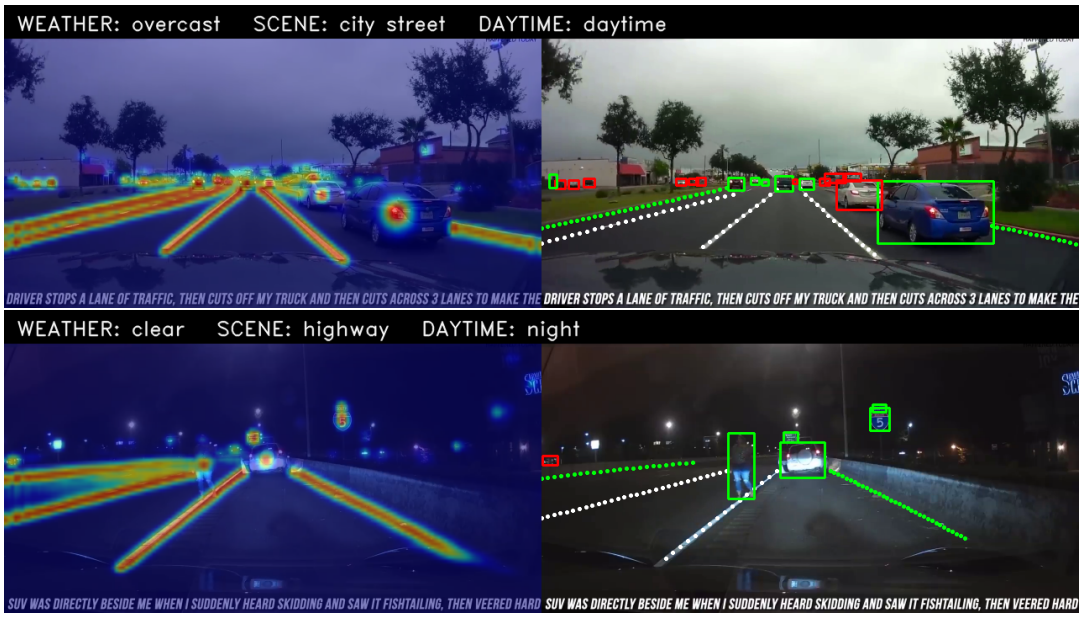


FIGURE 3.9: Qualitative results of Multi-Task Perception model

All the experiments reported in Table 3.3 are run with the same set of hyperparameters: in particular each model is trained for 75 epochs with a batch size of 64 (for a total of over $160K$ steps) using the optimizer Adam, the learning rate is linearly increased from zero to 2.5^{-4} during 3500 steps, then is halved after $100K$ steps. For comparison, the results reported in Table 3.3 are obtained with six different architectures, testing five different backbones: Resnet (34, 50 and 101), MobilenetV2 and Efficientnet-b2 and two necks (BiFPN and Simple, introduced in Section 3.3.2). Only a single experiment using the simple version was sufficient indeed to prove the obvious superiority of the BiFPN Neck.

Performance evalutation

Assessing the performance of the proposed model requires evaluating individual subtasks separately, which involves some absolutely nontrivial choices. The object detection and occlusion classification tasks are evaluated separately for the object detection task, for the former the Mean Average Precision (mAP) metric is used, of which there are several versions used in the literature. In order to compute the mAP, fist the set of ground-truth and predicted boxes needs to be associate, which is achieved by computing the Intersection over Union (IoU) measure among the possible pairs of true and predicted boxes, if this measure is greater than a threshold values the two boxes are matched and defines a true

Model	**Backbone**	**Neck**
RN34-sim	Resnet34	Simple
RN34-bifpn	Resnet34	BiFPN
RN50-bifpn	Resnet50	BiFPN
RN101-bifpn	Resnet101	BiFPN
EnB2-bifpn	Efficientnet b2	BiFPN
Mn-bifpn	MobilenetV2	BiFPN

TABLE 3.2: Evaluated configurations

positive detection (TP), unmatched boxes are instead respectively defined false positives (FP) for detection boxes and false negatives (FN) for ground truth boxes. These results can be used to compute Precision (PR) and Recall (RE), which are defined as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

To compute the mAP a precision-recall curve is constructed by varying the confidence threshold, which can be achieved by sorting the predictions by confidence and computing the AP considering the first n predictions, from $n = 1$ to $n = N_{det}$, finally to compute the area under the curve it is usually sampled at fixed recalls values. For this work, the formulation of CoCo is used [102], which defines a 101-point interpolated AP, moreover the mAP@0.5 metric is chosen, where 0.5 is the IoU threshold used in the association step. Predictions a with a confidence less than a threshold value, set at 0.25, are eliminated before calculating the metric. The success of the occlusion classification task is obviously dependent on the success of the detection task, in fact for FP detection the ground-truth of the occlusion is not definable and vice versa for FN detection the occlusion prediction is not available. Therefore, the object occlusion sub-task is evaluated only for TP detections: for this purpose, matching is first performed between the set of detection and ground-truths, using the Hungarian algorithm [103] to solve the minimum weight matching problem, employing the IoU as the pairwise distance measure. For the set of successfully matched detections, the predicted occlusion is retrieved from the occlusion map evaluated at the center-points and the Accuracy metric is computed.

Evaluating the performance of the lane estimation task is a tougher task because, as mentioned above, while a task such as object detection is universally formulated as a bounding box prediction problem, for lane estimation the formulations are varied and different from each other. For the chosen formulation, where each lane is a set of points sampled from the curve representing the line, at the time of writing there appears to be no shared evaluation methodology in the literature, therefore the option was chosen to generate binary segmentation

Model	**Object Detection**		**Lane Est**	**Scene (F1)**		
	Boxes	Occl		Wtr	Scn	ToD
RN34-sim	0.210	0.913	0.640	0.815	0.790	0.932
RN34-bifpn	0.265	0.910	0.644	0.817	0.791	0.932
RN50-bifpn	0.274	0.911	0.648	0.822	0.794	0.935
RN101-bifpn	0.281	0.912	0.65	0.825	0.794	0.934
EnB2-bifpn	0.272	0.907	0.647	0.821	0.790	0.937
Mn-bifpn	0.230	0.9	0.633	0.813	0.790	0.934

TABLE 3.3: Results of various configurations for all the tasks evaluated on BDD100K

masks from the prediction, to be using for evaluating the task in a more common segmentation setup. After the clustering phase of the prediction decoding process, a polynomial curve fitting is performed for each set of lane keypoints obtained, in this way a smoother estimate is obtained. Subsequently, the fitted lanes are sampled and plotted as polygonal chains in a binary segmentation mask, the similar ground-truth segmentation map is instead obtained by plotting the ground-truth lane keypoints. Finally, the IoU metric between the map obtained from the prediction and the map obtained from the gt is computed, note how this metric as implemented is agnostic to the lane class as a single binary map is generated.
For the three scene classification tasks, the F1-score metric, commonly used in these scenarios, is calculated for each of the 3 multi-class classification subtasks. To calculate the multi-class F1-score, TP, FP, and FN are counted individually for each class, in the end they are all collected and used for the calculation of Precision and Recall, which are in the end used for the calculation of F1 as:

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

which corresponds to the harmonic mean of precision and recall.

Profiling

As introduced in the previous chapter, runtime is not an exhaustive measure, the former because it is strictly dependent on the combination of hardware and software that is used for inference, at the variation of which any comparison between models could be subverted, and for example, the best performing model on a given hardware could be mediocre on a different hardware. For these reasons, it is useful to report, in addition to run times, objective measures that are as independent of inference hardware as possible. Here the number of parameters and the number of Multiply-Accumulate (MAC) operations are reported, the former clearly being a strong proxy for memory occupancy, while the latter should give an unbiased hardware estimate of computational cost. In

Model	**Params** ($\times 10^6$)	**MACs** ($\times 10^9$)	**FP16 Infer (ms)**	
			Xavier	Nano
RN34-sim	29.1	61.67	14.34	205.64
RN34-bifpn	22.2	19.77	8.85	103.98
RN50-bifpn	24.6	21,8	11.09	130.06
EnB2-bifpn	8,7	7.45	14.11	135.30
Mobbv2-bifpn	2.7	5.9	5.86	68.37

TABLE 3.4: Inference results: inference times reported account only for the forward pass computation, without accounting for the decoding phase.

reality, these measures are not perfect, and there is no absolute consensus in the literature as to which is preferable.

Specification	**Jetson AGX Xavier**	**Jetson Nano**
GPU Architecture	Volta	Maxwell
GPU Cores	512 CUDA cores	128 CUDA cores
CPU Architecture	NVIDIA Carmel ARM v8.2	ARM Cortex-A57
CPU Cores	8	4
CPU Clock Speed	Up to 2.26 GHz	1.43 GHz
RAM	32 GB 256-bit LPDDR4x	4 GB 64-bit LPDDR4
Storage	32 GB eMMC	microSD card slot
Tensor Cores	Yes (64)	No
AI Performance	32 TOPS	0.47 TOPS
Max Power	30W	5W

TABLE 3.5: NVIDIA Jetson AGX Xavier vs Jetson Nano Hardware Comparison

Regarding the inference times, these are collected in a realistic deployment scenario on two different embedded platforms of the NVidia Tegra family: he AGX Xavier represents the high-end segment, being capable of up to 32 Tera Operation per Seconds (TOPs) with a power consumption of up to 20W. Meanwhile, the Jetson Nano represents the lowest end of the spectrum, with only 0.5 TOPs and 5W of power consumption. A more detailed hardware comparison between these two systems is shown in Table 3.5. Since these two devices are based on the same platform, the same software configuration can be replicated on the two, specifically, the 4.6.1 version of the Jetpack SDK (Nvidia's software development package) is used, which includes TensorRT (version 8.2.1). TensorrRT is Nvidia's proprietary high-performance deep learning inference library that enables optimized execution on desktop and embedded NVidia GPUs, among the optimizations provided on the platforms of choice, the inference with 16-bits floating-point precision has been enabled for all the experiments.

Part 2: Modern AI meets Traditional Compression

Chapter 4

Efficient Self-Attention

4.1 Motivations

The analysis of variable-length sequences constitutes a pivotal area of study within the realm of artificial intelligence, with a particular focus on neural networks. There is a wide variety of data that are defined by means of a sequence comprising continuous or discrete values, distinguished by inherent correlations between their elements.
To illustrate, a sequence exhibiting temporal correlations, commonly referred to as a time series, can be useful in describing the dynamics of a particular phenomenon over time. In numerous scenarios, unraveling the concealed structures within a time series may be desirable, as a mean to unveils the inherent characteristics, trends, patterns, and interdependencies of the underlying phenomenon. This makes time series analysis an indispensable tool for tasks such as forecasting, informed decision-making, and comprehensive data comprehension. For instance, one might be interested in understanding or even predicting the performance of financial indices starting from historical data records, or one might be interested in analyzing sensor acquisitions of an industrial facility to detect or anticipate irregularities or malfunctions.
Conversely, semantic correlations of a high-level nature are an intrinsic characteristic of natural language, where discrete elements, such as words, are intricately interconnected through the language's structural rules, specific to the language in use. While in this case one might leverage the known rules of the language in order to automatically analyze the meaning of a sentence, mapping a text to another language or performing automatic text generation, this approach falls short of the complexity of hand-creating a model of the language that is comprehensive enough to consider both elementary constructs and high-level semantic elements and even cultural nuances of the language. Again, therefore, it is useful to have generic models that can infer the statistical correlations of the sequences, hence the structure of the language, from a set of human generated text.

The development of data-driven paradigms for sequence modeling has gone for several years in parallel with models for structured data or image analysis. Recent years have also seen the predominance of deep learning models, which has led to a substantial rapprochement, if not outright unification, between the realms of computer vision and sequence modeling, particularly in the domain

of Natural Language Processing (NLP). Historically, the family of deep learning models developed explicitly for sequence modeling are the Recursive neural Network (RNN), that operates by maintaining hidden states that capture information from previous time steps. Transformer-based models, have lately revolutionized sequence modeling by introducing a self-attention mechanism that allows to more effectively capture dependencies between distant elements in a sequence. At the architectural level, transformers can be defined as the meeting point between computer vision and NLP; in fact, this approach, although originally introduced precisely in NLP, has now been widely deployed in the vision domain as well.

While Transformer-based models have greatly advanced the field of sequence modeling, they are not without their limitations. One of the primary challenges is their computational intensity and memory requirements. The self-attention mechanism, which is central to Transformers, involves quadratic time and space complexity concerning the input sequence length: as a consequence, using Transformers for very long sequences can become impractical, especially in real-time applications or on devices with limited resources. To address this problem, researchers have explored several solutions. One strategy is to use various techniques for approximate or sparse attention mechanisms, which reduce the number of pairwise comparisons during self-attention. Methods like sparse attention, kernelized self-attention, or fixed-size windows limit the quadratic growth in an attempt to make Transformers applicable to longer sequences while maintaining reasonable computational demands. Nevertheless, each of the options in the literature has disadvantages and limitations.

This chapter is devoted to presenting the work done in order to optimize the computational cost and memory utilization of the self-attention layer in the transformer family models. The proposed and tested approach is inspired by the world of lossy compression: an approximation technique is proposed for the attention layer that exploits Discrete Cosine Transform (DCT). DCT has been broadly leveraged in the data compression domain to represent a signal in a more compact form by transforming it into a frequency domain and selectively discarding the less relevant information. In the proposed work, DCT is being applied to compress the self-attention values, reducing the dimensionality of the matrix while retaining the most important information.
This approach can significantly alleviate the quadratic complexity problem associated with self-attention, making Transformers more computationally efficient. However, the challenge lies in choosing an appropriate trade-off between compression levels and the quality of the approximation to ensure the model's overall performance remains satisfactory. Moreover, the use of the proposed compression technique requires a major relaxation of the attention layer formulation, which will be carefully described and evaluated in the continuation of the discussion.

The proposed methodology is assessed based on both algorithmic complexity and performance within a standard NLP benchmarking scenario: although this

is by no means the only area of application for transformers it is certainly to date the most interesting, and is among those most affected by its limitations. The experimental analysis is conducted by defining a conventional NLP task, involving unsupervised pre-training on a substantial unlabeled text corpus followed by fine-tuning on supervised downstream tasks. Sentiment classification, as outlined in [104], serves as the benchmark task.
It should be emphasized that the objective is not to introduce a novel language modeling technique, aiming to rival the current state of the art, in particular, given the available academic-level computational budget.

The contribution can be summarized as follows:

1. A simple yet effective self-attention approximation is proposed, leveraging the properties of the DCT.

2. Experimental evidence demonstrates that the formulation enables a reduced memory footprint and faster inference, while maintaining competitiveness in NLP tasks.

3. Comparison with prominent competitors in the literature reveals that the method presented here provides the optimal trade-off between inference time and model accuracy.

The methodology and the results presented in this chapter have been already published as an Article [1] on the Journal of Scientific Computing [1], generating a fair amount of interest.

[1] https://link.springer.com/journal/10915

4.2 Background

4.2.1 Neural networks for Sequence Modeling

To begin with, while a detailed traction of the whole sequence models scenario is beyond the scope of this thesis, it is important briefly recap the core problem of applying deep learning techniques to data of a sequential nature.
A standard FF network is a nonlinear function $\Psi : \mathbb{R}^d \to \mathbb{R}^k$, which maps an input $x \in \mathbb{R}^d$ into an output $y \in \mathbb{R}^k$. In general, the function Ψ takes the form of a stack of L Fully-Connected layers, with the i-th layer being expressed as:

$$y_i = \sigma_i(W_i^T y_{i-1} + b_i)$$

where $y_0 = x$ and $y_L = y$.

Each layer is therefore defined by a weights matrix $W_i \in \mathbb{R}^{k_{i-1} \times k_i}$ plus a scalar bias term b_{i-1} with $i = 1, \ldots, L$, with $k_0 = d, k_L = k$. In addition, a nonlinear activation function $\sigma_i(\cdot)$ is applied to each layer. Formally, the total function can be written as:

$$\Psi(x) = \sigma_L(W_L^T \sigma_{L-1}(W_{L-1}^T \ldots \sigma_0(W_0^T x + b_0)) + b_L).$$

Clearly, from this definition it is clear that x must be of fixed size, hence the feed forward model cannot be readily applicable to data with sequential nature. The idea behind Recursive neural Network (RNN) is to process each element of $x_t \in \mathbb{R}^d$, referred to as a token, of a sequence of inputs $X \in \mathbb{R}^{n \times d}$ individually, while maintaining a knowledge of the elements already processed. This is achieving by introducing a hidden-state h that is feed as an additional input and is updated at each time step to incorporate the information of the current input. The i-th layer of an RNN, often termed as Recurrent Cell, is therefore defined as:

$$y_i(t) = \sigma_i(W_i^T y_{i-1} + O_i^T h_i(t) + b_i) \quad t = (0, 1, ..., n)$$

$$h(t) = \begin{cases} y_i(t-1) & \text{if } t > 0 \\ Id & \text{otherwise} \end{cases}$$

in addition to the input's weight matrix W, the hidden state weight matrix $O \in \mathbb{R}^{k_{i-1} \times k_i}$ is introduced, the hidden state $d(t)$ is initialized to a fixed value before the first token is processed, and it's subsequently updated as a function of both the current input and the previous value of d. After the final token is processed, $y_i(t)$ can be considered as a fixed size latent representation of the whole input sequence up to t, and can be then used for supervised training of the RNN. While RNNs were groundbreaking in their ability to model sequential dependencies, they have some limitations, such as difficulties in capturing long-range dependencies and a lack of parallelism in training. Long Short-Term Memory (LSTM) networks [105] and Gated Recurrent Unit (GRU) [106], which are variants of RNNs, introduces specialized gating mechanisms that improve their ability to capture long-term dependencies and mitigate the vanishing gradient problem.

4.2.2 Transformers

Transformers [107] represent the current state-of-the-art in DL models for sequence modeling tasks. This family of architectures replaces the recursion mechanism of RRNs with the introduction of the mechanism of *self-attention* to effectively process sequences of arbitrary length in a single forward operation. As depicted in Figure 4.1, a standard transformer is made of $\mathbf{n_{blocks}}$ identical blocks, each composed of two sub-blocks: a self-attention module and a feed-forward layer, each one followed by a layer normalization [108] operation and a residual connection. The feed-forward layers, as well as the normalization operations, are applied independently to each token of the input, the self-attention instead is used to aggregate the information carried by each token.

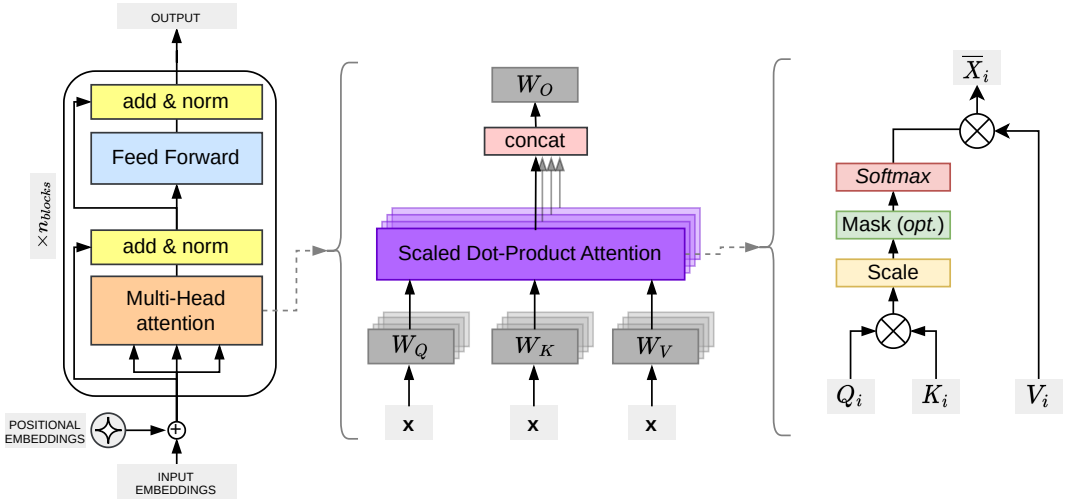


FIGURE 4.1: Overview of the general architecture of the standard Transformer Encoder.

Self attention

In the context of sequence processing, the self-attention mechanism operates on a set of tokens denoted as $X \in \mathbb{R}^{n \times d}$. This mechanism yields a transformed set of tokens, $\mathcal{X} \in \mathbb{R}^{n \times d}$, where each new row token element in $\mathcal{X}$ is derived through a weighted average of the entire original set X. At its core, self-attention assigns weights to each pair of token in X. These weights capture the importance of each token's contribution to the representation of the target token. Consequently, the self-attention mechanism excels in capturing intricate dependencies within the sequence by dynamically adjusting the contributions of different tokens, providing a nuanced understanding of contextual relationships. The resulting $\mathcal{X}$ thus embodies a refined representation that emphasizes the significance of individual tokens in relation to one another, enhancing the model's capacity to discern complex patterns and dependencies in the input sequence.

The affinity in the attention mechanism is computed by projecting the input sequence X onto three distinct sets: a set of Queries $Q \in \mathbb{R}^{n \times d_q}$, a set of Keys $K \in \mathbb{R}^{n \times d_k}$, and a set of Values $V \in \mathbb{R}^{n \times d_v}$. Each projection employs a dedicated matrix: $W_Q \in \mathbb{R}^{d \times d_q}$, $W_K \in \mathbb{R}^{d \times d_k}$, and $W_V \in \mathbb{R}^{d \times d_v}$. The three projections are expressed as follows:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{4.1}$$

The dot product between Q and K^T (with $d_q = d_k$ and $d_v = d$ in self-attention) yields an Energy score for pairs of tokens. This score is normalized and processed through a row-wise nonlinear softmax operation [109] to obtain the final weight matrix:

$$E(Q, K) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right) \tag{4.2}$$

The energy is then multiplied by V to produce the final attention output:

$$Atn(X) = E(Q, K)V = \text{softmax}\left(\frac{XW_Q(XW_K)^T}{\sqrt{d_q}}\right) XW_V \tag{4.3}$$

In most transformer implementations, a number $\mathbf{n_{heads}}$ of self-attention heads, each with its own set of projection matrices $W^j_{Q,K,V}$ (where $j = 1, \ldots, n_{heads}$), are applied in parallel, defining the Multi-Head Self-Attention (MhSA). The output of the multi-head attention is obtained by concatenating the results of individual attention heads, usually followed by an additional projection layer $W_O \in \mathbb{R}^{md_v \times d}$:

$$\mathcal{X} = \text{MhSA}(X) = [Atn_1(X) \oplus Atn_2(X) \oplus .... \oplus Atn_M(X)]W_O$$

For the purpose of this investigation, the multi-head aspect of transformer attentions can be from now on ignored, as the problem at hand is not dependent on the number of attention heads but is related to a single attention term.

Quadratic complexity of Attention

It is clear from (4.2) that being n the sequence length, the complexity of calculating the attention's weight matrix $E \in \mathbb{R}^{n \times n}$ scales quadratically ($O(n^2)$), in both memory and time. This quadratic dependence imposes significant constraints on the practical applicability of the self-attention mechanism, especially when confronted with very long input sequences. The increase in computational demands becomes prohibitive as the sequence length grows, thereby limiting the efficiency and scalability of models utilizing self-attention. To overcome the limitations of the quadratic dependence, several options have been already proposed in the literature, some of which are discussed in Section 4.3

4.2.3 Transformer-based Language Modeling

In order to process a natural language sequence (i.e, a sentence) with transformers or other DL models, tokenization is the initial step, breaking down raw

text into discrete units, or **tokens**, each typically corresponding to a word or subword. The tokens are then mapped into continuous, high-dimensional, embedding space, often represented as $E \in \mathbb{R}^{m \times d}$, transforming the discrete tokens into d-dimensional **embedding** vectors, where d denotes the dimensionality of the embedding. Hence, the sentence of n tokens is transformed into a sequence of embeddings $X \in \mathbb{R}^{n \times d}$. The embedding matrix E can be fixed like Word2Vec [110] or GloVe [111], or it can be optimized along the model parameters during the training process, in any case, the embedding space should capture the semantic and contextual information of the tokens.

Given the property of the self-attention mechanism, since their introduction, transformers have been popularized as powerful language modelers. However, transformer-based language models are known to be extraordinarily hard to train by relying only on labeled data for supervised tasks. For this reason, the scheme of adopting a pre-training strategy, already popular in previous language modeling techniques [111], [112], has become of great importance for transformers based modeling.

Pre-training of Transformers

Pre-trained transformers can be then effectively fine-tuned for downstream supervised tasks, usually with little to none architectural changes. The strength of BERT [113] and its derivatives [114]–[118] comes from the *bidirectional* pre-training strategy, which leverages a huge amount of unlabeled text in an unsupervised fashion. From an architectural standpoint, BERT simply employs the original transformer architecture adding a WordPiece tokenizer [119] to split an input sentence in a sequence of dictionary entries, which are then mapped to token embeddings. The unsupervised pre-training is carried by simultaneously optimizing for two tasks:

- Masked Language Modeling (**MLM**), where a percentage of the input tokens is masked at random, by replacing those with a placeholder `[MASK]` token, and then asking the model to predict back the masked tokens.
- Next Sentence Prediction (**NSP**) task, where a pair of sentences (sentence `A` and sentence `B`) are fed together to the model, divided by a separation token `[SEP]`, and the model is tasked to classify whether the sentence `B` is the actual next sentence that follows `A` or is a random sentence from the training corpus.

The training corpus is obtained by combining BooksCorpus [120] and English Wikipedia in order to obtain over 3,5M words of document-level corpus which include long sequences of sentence-level text required for the pre-training objectives.
Pre-trained transformers like the BERT and GPT (Generative Pre-trained Transformer) series have achieved remarkable success in natural language understanding and generation tasks, paving the way for the disruptive success of Large Language Models (LLMs): transformers with billions of parameters with have been trained on massive text corpus of text, with a computational cost in the

order of tens of millions dollars (based on the current cloud GPU prices).

In the experimental validation, a BERT-like model was trained using the methodology outlined in [121]. While the language modeling ability of such a model cannot be compared with that of a fully pre-trained BERT, it is instead well-suited for illustrating the advantages of the approximated attention in a fair comparison scenario. The experimental setup and the adopted training scheme are elaborated in Section 4.4.

4.2.4 Data Compression in Frequency Domain

The Fourier Transform, a fundamental tool in signal processing, decomposes a signal into a sum of sinusoidal components, exposing its frequency spectrum. While the Fourier Transform is a continuous-domain operation, its discrete counterpart, the Discrete Fourier Transform (DFT), is commonly used for digital signal processing. The DFT, however, poses computational challenges due to its complex values and quadratic complexity.
In response to these challenges, the Discrete Cosine Transform (DCT) emerges as an alternative that simplifies computation. The DCT expresses a finite sequence of elements (a discrete signal) in the form of a sum of cosine functions at distinct frequencies. Notably, the DCT, discrete and real-valued unlike the DFT, boasts invertibility denoted as IDCT and embraces properties such as energy compaction and decorrelation, where coefficients remain uncorrelated. These features render the DCT a widely employed mechanism in signal processing, particularly within lossy data compression algorithms like JPEG (images) [122], MPEG (video), and MPEG Layer III or MP3 (digital audio).
Various DCT variants exist, with the most prevalent, including its application in this work, being the type-II DCT [123], also serving as the initial DCT version. For a finite length sequence of N real-valued elements $\hat{x} \in \mathbb{R}^{N\times 1}$, the Type-II DCT defines a sequence $\hat{X}$ of the same length:

$$\begin{aligned} \hat{X}_K = \alpha_k \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad \text{for } k = 0, 1, ..., N-1 \\ \text{where } \alpha_k = \begin{cases} \sqrt{1/N} & \text{if } k = 0 \\ \sqrt{2/N} & \text{if } k \neq 0 \end{cases} \end{aligned} \tag{4.4}$$

Given its linearity, the forward DCT can be conveniently expressed through a dot-product operation between the sequence x and a transformation matrix $D \in \mathbb{R}^{N\times N}$. Formally, $\hat{X} = DCT(\hat{x}) = D\hat{x}$, where:

$$D_{n,k} = \alpha_k \cos\left(\frac{\pi(2n+1)k}{2N}\right) \tag{4.5}$$

Owing to the normalization term α, the matrix D attains orthogonality, facilitating the expression of the inverse transform as $D^{-1} = D^T$. Thus, $IDCT(\hat{X}) = D^T\hat{X} = \hat{x}$, effectively circumventing the computational challenges associated

with the inverse calculation.

In the realm of lossy compression algorithms, the computation of DCT coefficients precedes the retention of only a subset of the most relevant values. A straightforward approach involves defining a matrix $\overline{D} \in \mathbb{R}^{M \times N}$ (where $M < N$) by preserving solely M rows of the transformation matrix D. Leveraging $\overline{D}$ allows the derivation of a compressed representation $\overline{x} \in \mathbb{R}^{M \times 1}$ through the forward DCT. Subsequently, a lossy reconstruction of the original $\hat{x}$ becomes achievable with the inverse transform. For transformations involving both rows and columns, the observations made earlier can be generalized using 2D-DCT. Given a finite sequence of $N \times N$ real elements $\hat{x} \in \mathbb{R}^{N \times N}$, the 2D-DCT computation follows the formula $\hat{X} = DCT(\hat{x}) = D\hat{x}D^T$. Employing the same methodology, the compression procedure extends seamlessly to the two-dimensional case.

4.3 Related Works

4.3.1 Efficient Attention Heads

As briefly mentioned in the introduction, the quadratic complexity of the attention is a well-studied issue in the deep learning community. A variety of solutions have so far already been proposed, which can be roughly categorized in three classes: (i) methods that try to approximate or factorize the attention as defined in the original formulation [124]–[130] (ii) methods that reformulate the definition of attention (e.g., by introducing locality constraints) to avoid the complexity bottleneck [131]–[136] (iii) contributions which entirely remove the self-attention, usually by proposing an alternative paradigm [137]–[139]. The proposed methodology clearly falls in the first category, therefore hereafter a brief description of the principal competitors is provided, a few of which will be used for comparison in the experimental Section 4.4.

Attention Matrix Reduction

Reformer [126] introduces an attention approximation method called Locality-Sensitive Hashing (LSH) attention. LSH attention works by dividing the sequence into smaller segments and hashing each segment into a bucket. Each query position then attends only to the keys in the same bucket as itself, rather than attending to all keys in the sequence. This reduces the computational cost from $O(L^2)$ to $O(LlogL)$, where L is the length of the sequence. The approximation becomes more accurate as the number of hashes increases, and the computational cost can be adjusted depending on the available compute budget. Additionally, the Reformer model uses reversible residual layers instead of standard residuals, which allows storing activations only once in the training process instead of N times, where N is the number of layers. This further reduces the memory usage of the model.
In Linformer [124] try to achieve linear complexity in self-attention starting from the empirical observation of the attention matrix being low-rank. The authors introduce a set of learnable linear projection matrices $\bar{\epsilon}_i(i = 1, .., n_{heads})$ to project Q and V in a lower dimensional space, the, The projected query, key, and value matrices are then used to compute smaller attention matrices, each of which is a scaled dot-product attention between a subset of the queries and keys. The resulting attention matrices are then concatenated and multiplied by a trainable projection matrix to obtain the final attention output.
Performers [125] introduce a kernelizable attention mechanism (FAVOR+) to approximate the softmax attention with a complexity of $O(n)$, claiming to achieve a nearly-unbiased estimation of the attention matrix. The mechanism is composed of two parts: the FA-part and the OR+ part. The FA-part approximates the softmax kernel using a low-rank factorization of the query-key dot product matrix, while the OR+ part approximates the orthogonal random features used in the computation of the softmax kernel.
More recently, Nyströmformer exploited the usage of the Nyström approximation, which is commonly used in kernel methods to approximate the Gram matrix (positive semi-definite) with a low rank matrix. To avoid computing

the full attention, the authors exploit a relaxation of the Nyström method by individually computing the softmax operation of the three decomposition submatrices before the dot product operation.
SOFT [128] builds on top of [127] by replacing the dot-product operation with a Gaussian kernel, thus entirely removing the softmax operation from the formulation allowing, for a proper application of the Nyström method. Moreover, they propose a Newton-Raphson based method to approximate the pseudoinverse operation, in contrast with the less efficient Moore-Penrose pseudoinverse used in [127].

Differences with Model Compression

Understandably, the topics covered in this discussion may be confused with prevalent techniques designed to diminish the inference cost of generic deep learning models. Quantization methods [140], leverage reduced precision arithmetic, typically employing 8-bits integers or 16-bits floating-point representations. Pruning techniques, [141], involve the removal of less important weights or nodes from the network, thereby reducing the model's complexity. Additionally, knowledge distillation [54], is a strategy to transfer the knowledge of a large model to a smaller one, facilitating more efficient inference.
However, it is crucial to note that the formulation presented in this chapter, along with the previously introduced competitors, diverges from these model compression strategies, and they are in most cases of sense orthogonal to them. While quantization, pruning, and knowledge distillation primarily target the reduction of inference times, often tailoring their approaches to the specific hardware capabilities used for inference [142], efficient attention formulations address a distinct challenge. These formulations center on mitigating the quadratic complexity associated with dot-product attention used in the transformer architectures. Unlike compression strategies, which are geared toward hardware efficiency, efficient attention formulations focus on enhancing the computational efficiency of attention mechanisms, in order to handle increasingly large input sequences effectively. The goal is to streamline the attention operation rather than reducing the model's size or complexity, providing an alternative avenue for enhancing the scalability and efficiency of attention mechanisms in deep learning architectures.

4.3.2 Frequency Domain signal approximation

Signal approximation techniques play a crucial role in various fields, ranging from signal processing to data compression. One powerful approach to signal approximation leverages the frequency domain, utilizing transforms such as the Fourier Transform and, more specifically, the Discrete Cosine Transform (DCT). These techniques provide a means to represent signals in a more compact and efficient manner, revealing the underlying frequency components that contribute to the signal's structure. One prominent application of the DCT is in image compression, where it forms the basis for widely used standards such as JPEG (Joint Photographic Experts Group). The DCT's ability to concentrate signal energy in a limited number of coefficients facilitates efficient image

representation, reducing data size while preserving essential visual information. This transformative process allows for significant compression ratios without substantial loss in image quality.

4.3.3 Neural Network in Frequency Domain

deep learning methods that incorporate frequency domain operations, still a niche topic in this field. An earlier contribution by Gueguen et al. [143] presents an alternative use of frequency domain techniques in deep learning. Their work involved operating a Convolutional Neural Network (CNN) on the Discrete Cosine Transform (DCT) coefficients of JPEG-compressed images. The motivation behind this approach was to circumvent the necessity of running the complete JPEG decoding algorithm, thereby improving computational efficiency.
Several other contributions have explored concepts related to frequency domain operations for computer vision problems. Among those: Dziedzic et al. [144], explores the concept of band-limiting, a compression scheme that artificially constrains the frequency spectra of convolutional filters and input data during training in Convolutional Neural Networks (CNNs), to control resource usage which requires no modifications to existing training algorithms or network architectures. Rajesh et al. [145] introduces the DCT-CompCNN architecture: by modifying the input representation to include DCT coefficients, the CNN classifier achieves improved performance compared to the conventional approach.
Xu et al. [146], proposes a learning-based frequency selection method to address the downsampling challenges in neural networks operating in the spatial domain. Inspired by signal processing theories, the method identifies trivial frequency components that can be removed without accuracy loss.
Dos et al. [147] investigates the impact of spatial resolution and JPEG quality on CNN performance. The study designs a Frequency Band Selection (FBS) technique to reduce the computational complexity of CNNs operating directly on the JPEG compressed domain.
While these works share the common theme of integrating frequency domain principles into deep learning architectures, each approaches the task with its own unique perspective, resulting in varied degrees of success across different computer vision applications.

Limiting the analysis to the domain of transformers, one notable and recent advancement in this direction is the F-Net [138]. The F-Net stands out as a pioneering work that goes beyond mere utilization of Fourier-related transforms in deep learning; instead, it boldly replaces the self-attention mechanism in the transformer architecture with a two-dimensional DFT operation. Although this may appear superficially akin to the methodology presented in this work (see Section 4.3), a fundamental distinction lies in the formulation. F-Net does not seek to approximate the dot-product operation within self-attention, but rather opts for a comprehensive substitution with the DFT, showcasing a unique and innovative approach to handling attention mechanisms in transformers.

Together, these works contribute to the nascent but promising intersection of deep learning and frequency domain operations, expanding the horizons of signal processing within the realm of neural network architectures.

4.4 Proposed Method

4.4.1 Baseline Approach

The objective of this investigation is to exploit the DCT introduced in Section 4.2 to formulate an approximation method that mitigates the quadratic growth of the attention matrix in (4.2) with respect to the input sequence length n for an input $X \in \mathbb{R}^{n \times d}$.
Given the three $(n \times d)$ matrices Q, K and V defined in (4.1), all functions of input X, a straightforward approach involves obtaining three compressed representations $\overline{Q}$, $\overline{K}$ and $\overline{V}$, each of length $\overline{n} << n$, by computing the DCT of each matrix over the dimension n and retaining only $\overline{n}$ DCT coefficients. For ease of understanding, the forward DCT can be expressed using on the matrix formulation of (4.5), enabling the straightforward derivation of a transformation matrix $\overline{D} \in \mathbb{R}^{\overline{n} \times n}$ to compute the required $\overline{n}$ DCT coefficients.
Denoting the transformation matrix as $\overline{D}$, the formulation is as follows:

$$\overline{Q} = \overline{D}Q, \quad \overline{K} = \overline{D}K, \quad \overline{V} = \overline{D}V \tag{4.6}$$

Upon substitution in Equation (4.2), the result is:

$$\overline{E}(\overline{Q}, \overline{K}) = softmax\left(\frac{(\overline{D}Q)(K^T\overline{D}^T)}{\sqrt{d}}\right)$$

Considering the numerator inside the softmax operator, it is evident that:

$$\overline{D}(QK^T)\overline{D}^T = DCT_{2D}(QK^T)$$

Leveraging the associative property of the dot-product, $\overline{E} \in \mathbb{R}^{\overline{n} \times \overline{n}}$ is obtained without explicitly computing the original $E \in \mathbb{R}^{n \times n}$. Subsequently, the compressed attention output is computed by multiplying with $\overline{V}$:

$$\overline{Atn}(X) = \overline{E}(\overline{Q}, \overline{K})\overline{V} \tag{4.7}$$

Finally, the resulting approximated attention $\widetilde{Atn}(X)$ is obtained with an inverse DCT:

$$\widetilde{Atn}(X) = IDCT(\overline{Atn}(X)) = (\overline{D})^T\overline{Atn}(X) \tag{4.8}$$

From (4.7), (4.8):

$$\widetilde{Atn}(X) = (\overline{D})^T[\overline{E}(\overline{Q}, \overline{K})]\overline{D}V = IDCT_{2D}(\overline{E})V$$

To reiterate, the approximated attention experiences growth in memory and complexity with $O(\overline{n}^2)$. Choosing a sufficiently small $\overline{n}$ enables an approximation of linear growth with respect to the original input length n. A notable relaxation introduced in the proposed method is the effective utilization of:

$$\widetilde{Atn}(X) = IDCT_{2D}(softmax(DCT_{2D}(QK^T)))V \tag{4.9}$$

where the normalization term $\sqrt{d_e}$ is omitted. It is important to note that $softmax(DCT(x)) \neq DCT(softmax(x))$, hence, the introduction of a relaxation is implicit when computing the Inverse DCT. Similar relaxations involving the softmax function have already been proposed in [124] and [127]. In Section 4.4, a detailed discussion on its implications is presented, along with the definition of a strategy to experimentally evaluate the performance degradation caused by the utilized relaxation.

4.4.2 Complete formulation

A first possible improvement to make the proposed formulation more efficient from a computational standpoint is to avoid the calculation of three distinct forward DCT transforms as in (4.6). Recalling the formulation for Q, K and V in (4.1), computation can be saved by computing only the DCT of X ($\overline{X} = \overline{D}X$) and then utilizing the compressed $\overline{X}$ in place of X in the attention formulation of (4.3). This can be easily proven to be equivalent to the approximated attention defined in (4.7). The improved formulation can be then formalized as in Algorithm 1.

Algorithm 1 Efficient attention with DCT

Input $X \in \mathbb{R}^{n \times d}$
Output $\widetilde{X} \approx Atn(X)$
Require: $\overline{D} \in \mathbb{R}^{\overline{n} \times n}$
1: $\overline{X} = DCT(X) = \overline{D}X$
2: $\overline{Q} = \overline{X}W_Q, \quad \overline{K} = \overline{X}W_K, \quad \overline{V} = \overline{X}W_V$
3: $Atn(\overline{Q}, \overline{K}, \overline{V}) = E(\overline{Q}, \overline{K})\overline{V} = softmax\left(\frac{\overline{Q}\overline{K}^T}{\sqrt{d}}\right)\overline{V}$
4: $\widetilde{X} = \overline{D}^T \left[Atn(\overline{Q}, \overline{K}, \overline{V})\right]$
5: **return** $\widetilde{X}$

From an efficiency standpoint, the choice of the matrix formulation to compute the DCT is optimal: a single $\overline{D}$ can be precomputed, memorized and shared across all the attention modules of the transformer architecture of choice. This in stark contrast with [124] and similar methods, where each attention head requires its own learnable projection matrix, resulting in a total of $(n_{heads} * n_{blocks})$ matrices to be stored in memory.

Moreover, relying on a known linear transformation matrix has its own set of advantages: (i) it reduces the total number of trainable parameters, making for a lighter and more efficient training (ii) learning a transformation matrix $E \in \mathbb{R}^{n \times \overline{n}}$ as in [124] implies that the input sequence length must be exactly n, taking away the option of model input of arbitrary lengths. When using the DCT instead, $\overline{D}$ can be easily recomputed as needed, or it is possible to exploit an algorithm for fast cosine transform without explicitly relying on $\overline{D}$. For the latter, in the fine-tuning experiments (Section 4.4.2) the Makhoul's algorithm

[148] is employed, which leverages the Fast Fourier Transform (FFT) to efficiently compute the DCT of a $N-$point real valued signal.

To conclude, among the different transforms available in the literature, DCT was chosen because it is an efficient way of compressing information, it can be expressed as a matrix product, its inverse calculation is also linear, and operates in the real numbers field.

4.4.3 Softmax relaxation

As introduced in Section 4.4.1, a significant relaxation exploited by the proposed formulation, wherein the inverse DCT is computed for the result of a nonlinear function $softmax(x) : \mathbb{R}^n \to \mathbb{R}^n$ applied to the outcome of the forward DCT, as indicated in (4.9).
For the sake of completeness, the softmax function is defined as:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=0}^{n} e^{x_j}} \quad i = 0, 1, ..., n$$

This function is commonly used in the deep learning domain to highlight larger values and hide the ones significantly smaller than the maximum; moreover, it constrains the output of a layer to sum to 1 and returns values between 0 and 1. Ideally, to avoid the introduced relaxation, a function $\overline{s}$ would be required such that:

$$\overline{s}(DCT(x)) = DCT(softmax(x)).$$

This function is trivially $\overline{s}(DCT(x)) = \overline{s}(\hat{x}) = D(softmax(D^T\hat{x}))$, which is unsuitable since it implies passing through a higher-dimensional space, precisely what the proposed method aims to avoid. With the relaxation introduced, the computation of the matrix $E \in \mathbb{R}^{n\times n}$ can be circumvented.

When leveraging the proposed formulation, two potential sources of error are introduced when compared to the standard attention definition: (i) an *approximation* error induced by lossy compression using $\overline{n} < n$ DCT coefficients and (ii) a *relaxation* error induced by the usage of the aforementioned softmax relation. The approximation error is intrinsic to the definition of lossy data compression, while the relaxation error needs to be carefully evaluated to establish the mathematical validity of the proposed methodology.

A simple yet effective strategy is proposed to experimentally evaluate the contribution of the softmax relaxation on the overall error degree: the full matrix $E \in \mathbb{R}^{n\times n}$ is explicitly obtained as in (4.2), then its forward and inverse DCT are computed to obtain a lossy reconstruction $\widetilde{E} \in \mathbb{R}^{n\times n}$ which is then used in the following steps to obtain the attention output.
With this setup, the relaxed formulation of (4.9) is never used, hence only the approximation error is added: a simple way of quantifying the relaxation error is to compare the experimental results obtained with this formulation with those of the efficient attention formalized by Algorithm 1. It is worth clarifying

Algorithm 2 Evaluation of DCT-induced error

Input $X \in \mathbb{R}^{n \times d}$
Output $\widetilde{X} \approx Atn(X)$
Require: $\overline{D} \in \mathbb{R}^{\overline{n} \times n}$
1: $Q = XW_Q, \quad K = XW_K, \quad V = XW_V$
2: $E \Leftarrow E(Q, K) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)$
3: $\overline{E} = \overline{D}E\overline{D}^T \in \mathbb{R}^{\overline{n} \times \overline{n}}$
4: $\widetilde{E} = \overline{D}^T E\overline{D} \in \mathbb{R}^{n \times n}$
5: $\widetilde{X} = \widetilde{E}V$
6: **return** $\widetilde{X}$

that the above setup it is only intended for evaluation and comparison, since the quadratic attention is explicitly computed in line 2 of Algorithm 2 there would not be any benefit in using this formulation in a real use scenario.

4.5 Experiments and Evaluation

The purpose of this experimental section is to evaluate the capabilities of the proposed approximation method to mitigate the computational cost and memory usage of the self-attention layer, at the same time quantifying any degradation in the expressive capabilities of a transformer model equipped with the proposed attention variant. Therefore, inference times (milliseconds - ms) and peak memory consumption (Mega Bytes - MB) are evaluated in a first step: it is important to remember that these measurements are completely agnostic to the task on which the model is trained. For the second part, on the other hand, as metrics are evaluated on a specific task (NLP), it is critical to standardize the training methodology so that the comparison is fair. Therefore, in the first part of this section, the experimental setup used for the rest of the analysis is defined.

4.5.1 Experimental Setup

The experimental setup follows the transfer learning scheme common in NLP: first the model is trained on a large dataset of unlabeled corpus data, then it is fine-tuned on a downstream supervised task. In Section 4.3 the results both on the pretrain and the downstream task are reported, while in Section 4.2the results in terms of inference speed and memory occupation are presented and discussed.

Model Architecture

The transformer architecture adopted for the experiments is inspired by $\textbf{BERT}_{\textbf{small}}$ introduced in [149]. The model architecture follows the same structure of the original transformer [107] while only using $n_{blocks} = 4$ instead of the 12 of $BERT_{base}$ in order to keep a reasonable memory footprint even when training with the standard attention head. Each multi-head attention uses 8 heads, the embedding dimension d is 512 and the hidden dimension of the feed-forward layer is 2048. For the input tokenization, the same pretrained WordPiece tokenized used in BERT is employed, using the implementation *"bert-base-uncased"* provided by the Transformers library [150].

Pretraining

The workflow used for the evaluation is based on the pipeline proposed [121], combining various techniques to train a BERT-style language model within a reasonable computational budget. Following this setup, optimization is exclusively done for the masked-language model (MLM) task with a sparse token prediction head [117], omitting the next sequence prediction (NSP) objective. Only English Wikipedia text is used as the training corpus, and for increased training throughput, 10 masked copies of the dataset are precomputed with a masking probability of 0.1. Additionally, the maximum sequence length n is capped at 128 tokens to accommodate larger batch sizes. The utilized setup

largely mirrors the one proposed, utilizing the AdamW optimizer [151] with ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e{-}6$) and a weight decay of 0.01. To allow for an unbiased comparison of models with significantly different training speeds, the fixed time-budgeted scheduler is discarded from the training recipe. Instead, the total number of optimization steps is fixed to $100k$, and the learning rate is linearly increased from 0 to the Peak-lr with a warm-up proportion of 0.06, followed by a linear decay for the remaining steps. The peak learning rate (LR) is set at $1e{-}3$ with a minibatch size of 4096, achieved through two gradient accumulation steps.

Fine-tuning

In the spirit of keeping the experimental setup simple and understandable, the pretrained models are also evaluated by fine-tuning on the downstream task of sentiment classification of IMDb movies reviews [104]. This dataset comprises 50.000 movie reviews in plain English text, evenly divided between training and testing sets. Each review is manually labeled for sentiment classification, categorized as positive or negative based on the writer's opinion of the movie. Both positive and negative labels are equally distributed with a 0.5 ratio in both test and train splits, resulting in a perfectly balanced classification task. The average length of sequences in the training set is approximately 298 tokens (*min.* 13, *max.* 3055). To manage memory during training, the maximum sequence length is restricted to 1024 tokens, with longer sequences being truncated.
For fine-tuning the model, the Masked Language Model (MLM) head used for pre-training is substituted with a classification head. The first token $C \in \mathbb{R}^d$ of the transformer's output $\mathcal{X} \in \mathbb{R}^{n \times d}$, corresponding to the special `[CLS]` input token, is feed into two feed-forward layers with a $tanh(\cdot) : \mathbb{R} \to \mathbb{R}$ activation function to generate the binary classification output. The model is optimized using a Binary CrossEntropy objective function, employing the AdamW optimizer, as in the pretraining phase, but with the learning rate fixed at $1e{-}5$. In total, each pretraining model to undergoes fine-tuning on the supervised task for 10 epochs, utilizing a batch size of 64 with no gradient accumulation steps.

Implementation

From an implementation standpoint, the optimization engine DeepSpeed [152] is utilized, with mixed precision training provided by the APEX[2] backbone. To prevent potential interference with the efficient attention formulation, fused linear-activation-bias layers and the APEX LayerNorm implementation, commonly used for training acceleration, are avoided. All experiments are conducted on two 32GB Nvidia V100 GPUs, leveraging model-level parallelism.

4.5.2 Inference Performances

For a fair comparison, the transformer model defined in Section 4.2.2 the is used for all the tests, replacing only the attention head with the implementation of

[2] https://github.com/nvidia/apex

the efficient attention considered for the experiment. This experiments are conducted with randomly generated sequences of length $n \in \{128, 512, 1024, 4096\}$ adapting the batch size accordingly to fit the model in memory: to adjust for the non fixed batch size, both the inference time and the memory occupation are normalized for the current batch size. All the measurements are taken accounting only for the forward propagation.

TABLE 4.1: Inference performance of the efficient attention compared to other attention heads.

Attention Head	Sequence length (N) - Batch size (BS)							
	128 - 256		512 - 32		1024 - 16		4096 - 1	
	MB	ms	MB	ms	MB	ms	MB	ms
Vanilla	5.1	0.391	28.75	1.99	89.37	5.03	1250.0	45.6
DCT-0.25	4.55	**0.312**	22.62	**1.34**	**44.5**	**2.85**	**326.0**	**15.75**
Linformer-0.125	**4.23**	0.374	**21.0**	1.62	46.75	3.52	612.0	19.3
Nyström-0.125	4.73	0.41	24.87	1.83	55.5	4.18	488.0	47.71
Performer-0.125	4.91	0,425	23.87	1.89	59.5	4.2	548.0	28.93

In Table 4.1 the notation $\{\textbf{Model}\} - \{\textbf{scale}\}$ is adopted, where **scale** indicates the (fixed) ratio of the input sequence length used to instantiate the efficient attention. For the proposed method it defines the number of DCT coefficients, for Linformer the dimension of the learnable projection $\bar{\epsilon}$, for Nyströmformer the number of selected landmarks and for Performer the number of random features. While in principle scale could be defined as a constant, instead of a proportion of the input length (i.e, $DCT - 0.25$ for $n = 128$ implies $\overline{n} = 32$), it would be mathematically unfounded to assume that is possible to obtain a constant complexity for an arbitrary input length, whatever efficient attention head is used. From the reported results it is clear that the transformer model, equipped with the proposed DCT based efficient attention, outperforms all the competitors. As expected and discussed in Section 4.4 the savings in memory and inference times, from the usage of the proposed attention head, are directly proportional to the sequence length. In the next paragraph this important aspect is discussed in more details.

Scalability with Sequence Length

To obtain the inference results presented in the last paragraph, the batch-size (BS) had to be reduced when increasing the sequence length (N) in order to fit the model in memory. While this approach is perfectly suitable to compare different models - for a fixed sequence length - it does not allow to truly appreciate how each model scale with the sequence length. A new experiment is proposed to evaluate the growth in memory occupation and inference times when the sequence lengths are varied. In fact, as reported in Figure 4.2, only the multi-head attention modules are benchmarked, using a small fixed batch

size. For this experiment, for each attention, the same scale factors of Table 4.1 is used.

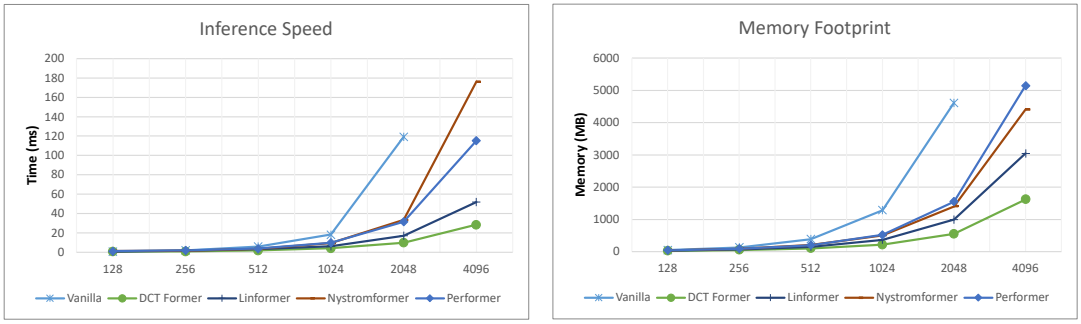


FIGURE 4.2: Plot of Inference Time (ms) and Memory Footprint (MB) for input sequences of different lengths. The Batch Size is fixed to 16 to allow for comparison.

As deducible from its formulation, the vanilla attention scales proportionally with the square of the input's length. The proposed efficient attention outperforms all the competitors in terms of scalability, both in inference times and memory footprint. In particular, for the longest input sequences the benefit of the efficient attention reflects in a memory reduction of up to 80%, thus successfully enabling to work with significantly longer sequences.
In the following section instead, the results are presented for the trained models on both the pretraining and downstream tasks, showing that the proposed attention can perform competitively when compared to significantly heavier formulations.

4.5.3 Performance on NLP benchmarks

Following the configurations detailed in Section 4.1, multiple settings of the proposed model are evaluated. For the pretraining stage, both the best loss (Cross Entropy) on the validation set and the Accuracy score for the MLM task are reported. Notably, the MLM can be evaluated as a multilabel classification problem, since for each masked token of the sentence it aims at predicting the correct vocabulary entry index (which in this case is 30522 entries long). In addition, to make the comparison fair the *normalized* accuracy score is also reported, which is obtained by dividing the accuracy score by the normalized average inference time obtained by the same model with a bath size of 256 and a sequence length of 128 (divided by 10^2 for readability). For the fine-tuning, the averaged Precision, Recall and F1-score obtained on the test split are reported.

All the sequences of the pretraining data are close to 128 tokens, requiring only a minimal amount of padding to be fixed to exactly 128 tokens: the finetuning sequences instead, while being truncated to 1024 tokens, presents a significant length variation. For this reason, for this part of the valuation it is used the Makhoul's method to compute the forward and inverse DCTs in the finetuning phase. With respect to the usage of the $\overline{D}$ matrix, this allows to work with any sequence length, while otherwise it would be limited to only work with

TABLE 4.2: Results of transformer models with different attentions

	Pretraining			Finetuning ↑		
Attention	Loss ↓	Accuracy (%) ↑	Normalized ↑	Pr	Re	F1
Vanilla	2.07	59.7	1.52	0.9	0.9	0.9
DCT-16	2.58	51.6	1.73	-	-	-
DCT-32	2.36	54.7	**1.74**	0.87	0.87	0.87
IDEAL-32	2.26	56.6	-	0.88	0.88	0.88
DCT-48	2.28	56.0	1.68	0.86	0.85	0.85
DCT-64	2.24	56.6	1.61	0.85	0.85	0.85
Linformer-16	2.29	56.2	1.49	0.80	0.80	0.80
Linformer–32	2.17	57.9	1.49	0.82	0.82	0.82
Linformer–48	2.13	58.5	1.49	0.83	0.83	0.83
Nystrom–16	2.25	56.6	1.37	0.88	0.87	0.87
Nystrom–32	2.13	58.8	1.26	0.88	0.88	0.88

sequences of exactly $\overline{n}$ tokens. Finetuning the Linformer it is instead far more problematic: the matrices $\bar{\epsilon}_i$ learned during the pretraining are only suitable to work for sequences of 128, the only way to perform the finetuning is hence to reinitialize the transformation matrix to work with sequences of 1024 and zero-pad all dataset elements to the maximum length. With Nyströmformer a simular issue is encountered, since also in this case the sequence length is required to be known and be evenly divisible by the number of landmarks, hence it was decided to pad the sequences in the same way as for Linformer.
For the results reported in Table 4.2, the models trained with the efficient attention formulation of Algorithm 1 are reported as DCT-$\{\overline{n}\}$, while the experiment IDEAL-32 follow the formalization of Algorithm 2 to evaluate the approximation error induced by the DCT compression, without leveraging the relaxation on the softmax operation. It is fundamental to understand that, while the Ideal setup clearly outperforms the efficient setup, the Ideal setup needs to compute all the matrices onto the $\mathbb{R}^n$ space, therefore losing all relevance to both memory and speed efficiency. Exploring alternatives to the softmax relaxation is a potentially interesting topic on its own, and can represent a future research direction.

Chapter 5

Latent Generative Models

5.1 Motivations

Artificial intelligence applied to image generation stands out as a captivating domain within the realm of machine learning studies, captivating not only enthusiasts, but also garnering significant attention and efforts from the broader scientific community. Generative AI refers to a class of artificial intelligence models and algorithms designed to produce new, often realistic, data instances that resemble or capture the patterns present in a given dataset.
To grasp the essence of generative models, consider the intriguing concept that, at a given image resolution, every conceivable natural and non-natural image at that resolution is representable as a point in ultra-high dimensionality pixel space. The rationale behind AI for generating natural images is based on the concept that a manifold exists within the space of all possible pixel combinations, encapsulating the realm where natural images reside. Therefore, when focusing on a specific class of natural images (people, cats, cars...), this concept suggests that the space of meaningful or natural images of the chosen class is not randomly scattered throughout all possible pixel combinations, but rather lies on a lower-dimensional structure or manifold. This notion can be intuitively understood by observing that natural images exhibit shared characteristics, patterns, and regularities. Take, for instance, images of faces, where common features like eyes, noses, and mouths are consistently arranged in specific ways. The inherent regularities and structures within the data find representation in a lower-dimensional manifold amid the vast high-dimensional space encompassing all conceivable pixel combinations. Thus, the pursuit of generative AI involves training models to navigate and learn these underlying patterns, enabling the creation of new and meaningful content that aligns with the observed regularities in the training data.

The problem underlying generative AI is thus to attempt to mathematically model, or rather approximate, the hidden distribution inherent in natural images by exploiting a formulation such that new and original images can be sampled from the approximate distribution. Generative models, particularly those based on deep learning and neural networks, attempt to model this manifold from a set of sample images that have been selected to represent the domain or class of interest. Training a deep learning model for this purpose requires a radical paradigm shift from the supervised learning cases seen earlier. Indeed, in

the most general case, generative models are built from an unsupervised learning process, where the model is exposed to a set of input data without explicit labels. During this process, the model tries to learn the intricate structures and hidden relationships present in the data, without relying on external information. More properly, the unsupervised training process attempts to learn a nonlinear mapping from the high-dimensional pixel space to a significantly lower dimensional embedding space. One of the simplest strategies to obtain a latent space that reflects the properties sought and to simultaneously learn to reconstruct the original image from the latent representation, which can be achieved by minimizing a reconstruction loss between the original and the recovered image. Models trained in this way fall into the category of Autoencoders. The definition of the latent space alone is insufficient for the generation part; indeed, the distribution of images in the reduced-dimensional space is generally unknown, rendering explicit sampling of new images impossible. One approach to address this limitation involves explicitly constraining the learned distribution of the latent space to follow a known distribution, typically a normal distribution. This constraint allows for the explicit sampling of the latent space. This could be achieved by simultaneously minimizing not only the reconstruction loss but also an additional divergence loss between the learned distribution and the target distribution. This is the approach behind the Variational Autoencoders, which are one of the first successful approaches to image generation with deep networks. The underlying concept has been taken up in subsequent developments, albeit with substantial variations, leading to the definition of new paradigms for training generative models, most notably Generative Adversarial Networks (GAN) and more recently Denoising Diffusion Probabilistic Model (DDPM), which are the focus of this chapter.

Diffusion models, in essence, identify a family of generative models, which, as previously discussed, seek at establishing a relationship between the unknown distribution of training data and a more manageable distribution, typically assumed to be Gaussian. The fundamental approach involves implementing a forward diffusion process, characterized as a Gaussian process, by progressively introducing Gaussian noise to a dataset sample until reaching a state of pure noise. Subsequently, the reverse process of gradually reconstructing the image from pure noise can be modeled by a neural network, which, when trained with this objective, implicitly acquires the capability to generate novel images that align with the sought-after distribution. One notable distinction is that VAEs employ an encoder-decoder bottleneck design, introducing a predetermined structure to the latent space. In contrast, DDPMs do not explicitly model the latent space; instead, it is implicitly represented through the diffusion process. This approach can be beneficial when dealing with intricate latent space structures, resulting in high-quality samples due to the model's capacity to capture fine details in the data distribution. However, the drawback of avoiding a predefined latent space structure in DDPMs is that they tend to meticulously model perceptually insignificant details in target images, often at the expense of capturing semantically relevant features. This can lead to a

heightened demand for computational resources, particularly during the training phase.
An insightful observation emphasized in [153] underscores that generative models essentially undertake two key tasks. Firstly, there is the task of perceptual compression, involving the modeling of perceptually relevant components and the abstraction of imperceptible details. The subsequent task consists in the actual generation process, which imply the modeling of relationships and conceptual compositions among the various semantic elements constituting the image. This observation presents an opportunity for optimization by separating these two components of the generative process. Initially, a low-dimensional latent space is defined to achieve perceptual compression, followed by training the generative DDPM on this new space instead of the RGB space. The spatial compression induced by the chosen latent space thus prunes unnecessary degrees of freedom corresponding to perceptually irrelevant details, introducing a strong bias in the training of the generative process. This bias results in much more efficient training and a significantly higher quality of the generated images. This constitutes the foundational methodology employed by LDM, representing the current state-of-the-art in generative diffusion models. In practical terms, LDM begins by training an Autoencoder to accomplish perceptual compression. Subsequently, the frozen encoder is leveraged to encode the training set in a lower-dimensional latent space. The Denoising Diffusion Probabilistic Model (DDPM) is then trained on this latent space, focusing on generating encoded images. During inference, the frozen decoder is utilized to retrieve generated images from the latent space. This integrated process exemplifies the efficacy of LDM in achieving both perceptual compression and efficient generative training.

The concept presented in this chapter is rooted in an intriguing observation: considering that the purpose of the latent space is to introduce perceptual compression, akin to a *lossy compression*, could an alternative non-deep compression strategy achieve similar results without relying on the costly training of an autoencoder? If such an approach were feasible, it could obviate the need for the resource-intensive training of an autoencoder, replacing it with a lightweight and efficient lossy compression algorithm, leveraging decades of existing research and understanding. Beyond training, the generation of new samples would also benefit from reduced memory and computational demands, stemming from the omission of the resource-intensive decoder model. Additionally, but of equal significance, this decision has led to a more profound comprehension of the inner workings of generative models within latent spaces. It has unveiled the requisite properties for an effective replacement of the autoencoder and shed light on the associated limitations. In pursuit of this understanding, a straightforward lossy compression framework is introduced, leveraging Vector Quantization (VQ) techniques to approximate image patches with a finite set of representative elements. The resulting latent space manifests as a discrete distribution across a finite set of indices. To address this discrete nature, a practical approximation is introduced to navigate the challenges of a discrete process that emulates a representation in a continuous domain. Lastly, a refinement of the discrete inverse diffusion process is proposed, aligning with the

established framework.

To sum up, the main contributions in this chapter can be summarized as:

- A shallow encoding for the perceptual compression stage is proposed, characterized by a simple definition and predictable behavior.
- A straightforward strategy is proposed to enable the training of a continuous space diffusion model on the discrete latent space. This is accomplished by enforcing an originality property in the latent space through a sorting of the finite set of latent values along the direction of maximum covariance.
- Finally, the reverse diffusion process is proposed to be redefined in a categorical framework, explicitly modeling the latent image representation as a set of categorical distributions over the discrete set of latent variables used for the lossy encoding.

The results and methodologies proposed below have been recently published as an article [3] in a specialized journal.

5.2 Background

5.2.1 Generative Models

The problem of image synthesis can be formalized as the search from a generative model G which approximates the unknown distribution $\mathcal{P}$ of "real-world" images, sometimes conditioned with respect to a certain class (cats, churches, faces, etc.). This task has already been studied in literature for other domains (text generation, music files...), and since its inception has found large use relative to long sequences or, in this case, to high-resolution images. These problems are related to both the computational power required to generate such models and the numerical instability phenomena which lead the process to be unsteady with respect to the choice of hyperparameters and the training process. On the other hand, the power and surprising results have pushed the literature towards the search for a compromise between performance and efficiency. In light of this, there are many recent works that attempt, through various techniques, to stabilize the generative methods and make them efficient.

Evaluation of Generative Models

Assessing the performance of generative image models is notoriously complicated, and different strategies have been proposed thought the years. Inception Distance (ID) [154] measures the dissimilarity between the distribution of generated images and a reference dataset using an InceptionV3 network pretrained on ImageNet [155]. ID is computed by extracting a feature representation from both the real and the generated images, and then computing the Kullback–Leibler (KL) divergence between the related distributions. Fréchet Inception Distance (FID) [156] takes this concept by computing instead the Fréchet Distance between the features distributions.
From a practical standpoint, computing the FID score requires a large set R of images from the real distribution and a comparable size set of generated images G. First, the pre-trained InceptionV3, up to its final pooling layer, is used to extract 2048-dimensional feature vectors from each image in both sets. Afterward, the FID is computed by measuring the Fréchet distance between the two sets of feature vectors, as defined in Equation (5.1). This computation is carried out after determining the mean vectors (μ_R, μ_G) and covariance matrices (Σ_R, Σ_G) for both sets of feature vectors:

$$\mathrm{FID}(G, R) = ||\mu_G - \mu_R||_2^2 + \mathrm{Tr}\left(\Sigma_G + \Sigma_R - 2(\Sigma_R^{1/2}\Sigma_G\Sigma_R^{1/2})^{1/2}\right), \tag{5.1}$$

where $\mathrm{Tr}(\cdot)$ defines the trace operator. The metric defined with this formulation ideally should quantify the perceivable difference in appearance between the generated and real images; in fact, as the FID score approaches zero, it should indicate that the two sets of images are indistinguishable. However, is still active investigated how the FID and similar metrics might fail at capturing semantic aspects of image content, such as object composition or scene coherence. Moreover, subtle details on the implementation have been shown to invalidate the reliability of these metrics [157]. In addition, a disadvantage

of these types of metrics is that they require the generation of an image set in the tens of thousands, which, depending on the model under consideration, can take quite an onerous amount of computation time. Nonetheless, for the purpose of this work, the assessment by means of the FID score was opted for.

5.2.2 Diffusion Models

Diffusion models are a class of likelihood-based generative models which define the generative process as a mapping from a prior distribution $\mathcal{N}(0, \mathrm{Id})$ to the target unknown distribution $\mathcal{P}$ of the real data. This is achieved by defining forward a degradation (or diffusion) process q which gradually corrupts a data sample x_0 with additive Gaussian noise at each time step t with $t \in (1, ..., T)$ and a reverse degradation process which learns an approximation p_θ parametrized by the weights θ of the denoising process.
The forward process $q(x_t|x_{t-1})$ is modeled by a conditional Gaussian distribution which, at each step, is defined by the mean $\mu_t = \mu_t(x_{t-1})$ and variance β_t:

$$q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t \mathrm{Id}), \tag{5.2}$$

where $0 < \beta_1 < \beta_2 < ... < \beta_T < 1$ are fixed according to a predefined variance scheduler, which in the original work is defined as linearly increasing from $\beta_1 = 10^{-4}$ to $\beta_T = 2 \cdot 10^{-2}$ with $T = 1000$. The reverse process $p_\theta(x_{t-1}|x_t)$ is also a conditional Gaussian distribution

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \beta_t \mathrm{Id}) \tag{5.3}$$

with θ usually being the parameters of a powerful denoising neural network and μ_θ is the mean of the learned inverse process. With the variance being fixed, only the mean of the posterior distribution has to be learned. Once the model is trained, sampling from the approximated distribution of real images can be done effectively by starting with a realization of pure noise $\epsilon \sim \mathcal{N}(0, \mathrm{Id})$, and sampling x_{t-1} from p_θ for $t = T, T-1, \ldots, 1$ to recover the sample $\tilde{x}_0$, which represents the initial generated state.

A suitable objective function to learn the reverse process can be derived by a reparametrization of the mean μ_θ, leading to, as derived in [153]:

$$\begin{aligned} \mathcal{L}_\epsilon(\theta) &= ||\epsilon - \epsilon_\theta(x_t, t)||_2^2 \\ t \sim \mathcal{U}(0, T) \quad \epsilon &\sim \mathcal{N}(0, \mathrm{Id}), \end{aligned} \tag{5.4}$$

where $\epsilon_\theta(x_t, t)$ is the neural network tasked with estimating the noise level on x_t at the uniformly drawn time step t. Since the forward process is a fixed Markov chain, x_t can be directly computed at any noise level t without going through the intermediate levels. Hence, Equation (5.4) can be efficiently optimized at random time steps t by simply predicting the noise realization ϵ added to x_0. An alternative parametrization of the mean can lead to a different definition of the training objective as the learning of the uncorrupted image $\tilde{x}_0^\theta$ given x_t and

t:

$$\mathcal{L}_{x_0}(\theta) = ||x_0 - \tilde{x}_0^\theta(x_t, t)||_2^2. \tag{5.5}$$

Although the two formulations are theoretically equivalent, the former is generally preferred because of its observed better quality of the generated images. In the remainder of the chapter, both expressions are used, and in particular, a combination of them is used to introduce the formulation for the reverse process in Section 5.3.2.

It's worth ending this section by remarking that, starting from [153], DDPMs have been leveraged in both unconditional generation, where the image space is randomly sampled, and conditional generation, where the generative process is guided by a prior, such as the image class in a multi-class scenarios or even complex natural language queries. For the scope of this work, only the unconditional generation is considered.

5.2.3 Latent Diffusion Models

As briefly stated in the introductory section, Stable Diffusion [158] achieved impressive results in image generation with DDPMs by shifting the generation process in latent space. This choice is motivated by the inherent difficulty of training DDPMs for high-resolution image generation, due to the waste of expressive capacity for generating irrelevant image details and the very high computational cost. The shift to latent space introduces a strong GreenAI-boosting perceptual and spatial compression, simplifying training and reducing computational cost. The latent space is defined by an encoder $\mathcal{E}(x) = z$ which maps an image $x \in \mathbb{R}^{w \times h \times c}$ into a latent representation $z \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times c}$ and a decoder $\mathcal{D}(z) = \overline{x}$ which reconstructs $\overline{x} \approx x$ in the image space. The pair $(\mathcal{E}, \mathcal{D})$ defines a Variational Autoencoders (VAE) pretrained on the Open Images dataset [159] (and frozen when training the diffusion model), trained by simultaneously minimizing a reconstruction term $||x - \overline{x}||_2^2$ and a regularization term useful to bound the variance of the latent space. Good results have been achieved with VQ-VAE regularization [160], which introduces a Vector Quantization (VQ) layer to regularize the latent space. This however shall not be confused with the VQ based compression method introduced in Section 5.3.1, especially since z is extracted from the VAE bottleneck before the VQ layer, which is hence embedded in the decoder.

Defining the generative process in latent space means redefining the loss in Equation (5.4) as a function of z:

$$\mathcal{L}_L(\theta) = ||\epsilon - \epsilon_\theta(z_t, t)||_2^2. \tag{5.6}$$

Hence the reverse diffusion model is used to sample $\tilde{z}$ and at the end of the generation process the sampled image $\tilde{x} = \mathcal{D}(\tilde{z})$ is recovered using $\mathcal{D}$. The work presented in this manuscript is conceived as an ablation of this method, by substituting the deep latent space defined by the VAE with a shallow latent space defined by a model-based lossy image compression technique.

5.3 Proposed Method

5.3.1 Latent-Space Encoding

Lossy image compression is a fundamental branch of image processing that addresses the problem of reducing the data required to represent an image by selectively discarding the information that is deemed less relevant for the human perception. Typical lossy compression algorithms encode images in ways that exploit the inherent redundancy, allowing them to be efficiently represented using fewer bits. The first aspect to consider when designing a compression algorithm that can be leveraged for latent-space image synthesis is the convolutional encoder-decoder style of the generative model itself, which can only produce output of fixed size. Conversely, the most popular image compression algorithms, with JPEG on top of all, produce variable-length compressed representations for different images. For this reason, a simple yet effective compression strategy is devised based on VQ techniques. It maintains a constant compressed representation size for each patch by employing a predefined codebook that maps input vectors to fixed-length code words, thus ensuring consistent compression regardless of image content. Similar strategies have been popular since the 1980s (see e.g., [161], [162]) for their simplicity and overall good performances. In the deep learning era, VQ-like approaches have been rediscovered [160], as has been the case with other compression techniques [1], [163].

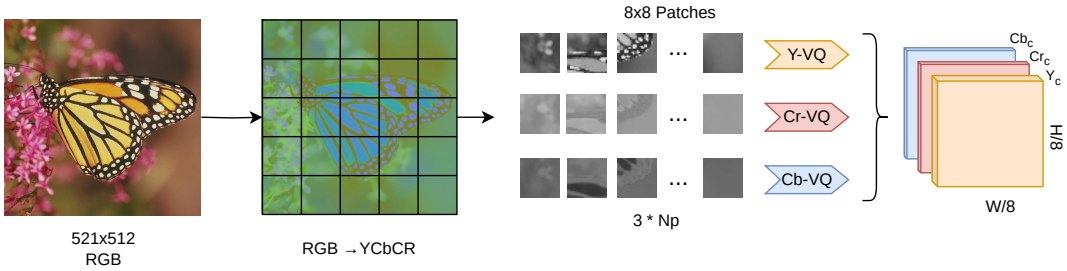


FIGURE 5.1: Image compression scheme

Given an RGB image $x \in \mathbb{R}^{w \times h \times 3}$, the model-based compression algorithm produces a latent representation $z \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times \bar{c}}$. The compression algorithm is designed by first converting the image to the YCbCr color space, this separates the image information into its perceptually relevant components, enabling a better use of compression techniques. Then, for each channel from 1 to 3, the image is divided in non-overlapping patches of $s \times s$ pixels, with each patch capturing a localized region of the image, preserving both spatial and chromatic information. Then, finally, the core of the proposed compression techniques involves vector quantization applied to each individual patch. Vector quantization replaces each patch with one (or more) representative vector, denoted as codewords, from a fixed size codebook, thus allowing the whole ($s^2 * 8$ bit) block to be encoded as the index i of the assigned codeword. Clearly, the strategy of the definition of the codebook and the codewords assignment strategy is crucial

to set the optimal tradeoff between compression ratio and information preservation, therefore the multiple strategies experimented are detailed hereafter. In Section 5.4.1 instead, all the technical details of the practical implementation of the quantization approaches are discussed, together with results on a lossy compression benchmark.

Vector Quantization

Vector quantization [164], [165] is a popular technique in the domain of data compression and signal processing. Its fundamental idea is the approximation of the entire data space X with a finite subset of representative vectors $\mathcal{C}$. This process not only reduces the cardinality of the space, but also enables a concise and efficient representation by assigning a codeword c_i to each element $x \in X$ of the subset. Conveniently, each entry c_i can be uniquely represented by its index $i \in \mathcal{I}$, with $\mathcal{I} \coloneqq \{1, ..., |\mathcal{C}|\}$.

Formally, this procedure is described by a quantizer defined by an encoder function $E(x)$ and a decoder function $D(i)$

$$q\colon X \longrightarrow \mathcal{C} \tag{5.7}$$

$$x \longmapsto c = q(x) = D(E(x)). \tag{5.8}$$

The encoder function $E\colon X \longrightarrow \mathcal{I}$, maps each element of the space into its compact code representation $E(x) = i \in \mathcal{I}$. On the other hand, the decoder function $D\colon \mathcal{I} \longrightarrow \mathcal{C}$ maps the index $i \in \mathcal{I}$ back to the representative vector c_i. The representative vector c_i is chosen by definition in order to minimize the distortion with respect to x, hence the encoder can be defined as:

$$E(x) = i \iff \|x - c_i\|_2^2 = \min_c \|x - c\|_2^2. \tag{5.9}$$

From Equation (5.9) the lossy nature of VQ is clear, since the encoding process implies an approximation error e, or $c_i = x + e$. For convenience, $\mathcal{C}$ is referred to as the *codebook*, and its elements c_i as *codewords* with i being the *code* associated to x. For the scope of this work, only fixed-rate quantizers are considered, where each code is represented with the same number of bits. Hence, the number of bits required to approximate x with i using a codebook $\mathcal{C}$ is $b_{\mathcal{C}} = log_2(|\mathcal{C}|)$, defined as *bitrate* of q. When $x \in \mathbb{R}^n$ with $n >> b_{\mathcal{C}}$, this makes VQ a very powerful compressor.

Defining an optimal quantizer is not straightforward, as it depends on the desired tradeoff between reconstruction error and compression rate, which is strictly dependent on $|\mathcal{C}|$. In the simple case where X is defined as a finite subset of $\mathbb{R}^n$ and $|\mathcal{C}|$ is fixed beforehand, recalling Equation (5.9), the formulation for the error introduced by encoding X with q simplifies to:

$$MSE(q) = \sum_{x \in X} \|x - q(x)\|_2^2 = \sum_{x \in X} \|x - D(E(x))\|_2^2. \tag{5.10}$$

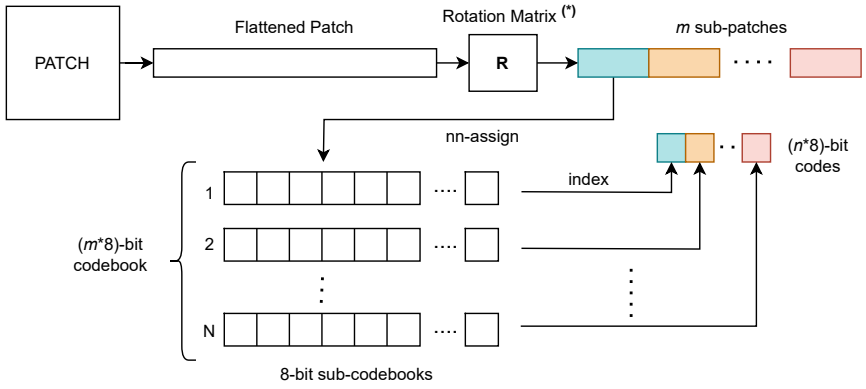


FIGURE 5.2: Product quantization. (*) When $R \neq Id$ this corresponds to OPQ.

From this definition, the choice of the optimal quantizer for X reduces to defining the set of vectors c_i which minimizes Equation (5.10), which are those provided by the k-means algorithm. K-means directly minimizes the MSE by iteratively updating centroids to minimize the sum of squared distances between data points and their assigned centroids. In Section 5.4.1the process of defining the optimal quantizer for the problem of image compression is detailed, where X is defined as a set of representative image patches.

(Optimized) Product Quantization

The base vector quantizer described in subsection 5.3.1, while being a powerful tool, implies some major drawbacks. The number of codewords $|\mathcal{C}|$ required to represent a code $E(x) = i$ using $b_\mathcal{C}$ bits is proportional to the cardinality of the codebook as $|\mathcal{C}| = 2^{b_\mathcal{C}}$. For moderately large x (i.e, an 8×8 8-bit grayscale image patch), a reasonably high compression rate of $1:16$ would imply encoding the block with $512/16 = 32$ bit, which would lead to a codebook $|\mathcal{C}| = 2^{32} \simeq 4B$ possible codewords! Clearly, such a large codebook would be hard to obtain and require large memory usage and intense computation cost, hence more sophisticated quantizers are needed.

Product Quantization (PQ) [166] is an upgrade in this direction. Instead of processing the input vector x as a whole, it is possible to split it into m sub-vectors $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ and consider a sub-codebook for each of them. For simplicity, it is common to have sub-vectors with equal length of $n' = \frac{n}{m}$: $x^{(j)} = (x_{m*(j-1)+1}, \ldots, x_{m*j})$, where m is a divisor of n. At this point, the quantizer is the Cartesian product of m sub-quantizers that act on each sub-vector:

$$q\colon X \longrightarrow \mathcal{C} = \mathcal{C}_1 \times \ldots \times \mathcal{C}_m \tag{5.11}$$

$$x \longmapsto c = (c_1, \ldots, c_m) = (q_1(x^{(1)}), \ldots, q_m(x^{(m)})). \tag{5.12}$$

Since each sub-quantizer q_j is a separate vector quantizer, defined by its codebook C_j, the code for x can be obtained by encoding independently each sub-vector:

$$E(x) = (E_1(x^{(1)}), \dots, E_m(x^{(m)})) = (i^1, \dots, i^m). \tag{5.13}$$

The fundamental difference is that the resulting code $(i^1, \dots, i^m)$ is a vector, therefore the total number of bits required to represent it corresponds to the sum of the bitrates of the sub-codebooks. Hence, encoding x with a 32-bit code will require just 4 8-bit codebooks, for a total of only 1024 possible codewords.

Optimized Product Quantization (OPQ) [167] offers further improvement. Instead of directly encoding the vector x as it is, it first applies an orthogonal matrix R (i.e., a change of basis). This operation is particularly useful in this case, as image pixels have correlations between them that are dictated by the structures present in the images subjects. Hence, simply splitting the images at fixed intervals may not be the best course of action, but it may be necessary to first reorder their pixels, so that correlated pixels end up in the same sub-vector. The problem of finding an optimal quantizer involves finding both the $\mathcal{C}_1, ..., \mathcal{C}_m$ codebooks and the matrix R subject to $RR^T = \text{Id}$. In [167] the authors derive a non-parametric solution as an alternating minimization scheme, where in one step R is given, and the codebook is learned, in the other the optimization of the MSE is carried out w.r.t. R while the codebook is fixed. The former step can simply be done via k-means in the transformed domain. For the latter, considering Equation (5.10), the problem can be written as

$$\underset{s.t.\ RR^T=\text{Id}}{\text{argmin}} \sum_{x \in X} \|Rx - Rc\|_2^2 = \underset{s.t.\ RR^T=\text{Id}}{\text{argmin}} \|RX - Y\|_F^2, \tag{5.14}$$

where in the last formulation, with a slight abuse of notation, X is the matrix containing the practical data samples from the space, and Y is the reconstruction of X in the transformed domain according to the current codewords estimates (i.e, $Y = D(E(X))$). This problem is known as Generalized Procrustes Alignment [168] and has a closed form solution in $R = VU^T$ with V and U being the orthogonal matrices of the Singular Value Decomposition of XY^T.

Residual Quantization

A different strategy to improve VQ is represented by additive quantization [169]. In this case, the quantizer q can be seen as the sum of a set of quantizers $\{q_j\}_{j=1}^m$. Each of these quantizers maps the vector x into its own codebook $\mathcal{C}_j$:

$$q \colon X \longrightarrow \mathcal{C} \tag{5.15}$$

$$x \longmapsto c = \sum_{j=1}^{m} c_j = \sum_{j=1}^{m} q_j(x). \tag{5.16}$$

Similarly to PQ, the code assigned to x is a vector in $\mathbb{R}^m$ and the bitrate of the encoding corresponds to the sum of the bitrates of the sub-codes. Unlike in PQ, here, the centroids are vectors with the same length as x and not just a

fraction of it.

Residual Quantization (RQ) [170] is a specific example of additive quantization. The idea is to give a hierarchy to the quantizers q_j. The first one, q_1, is learned normally without any additional information. The second one, instead, wants to be the optimal quantizer for the vector $x - q_1(x)$, i.e., the residual between x and its quantization. In general, for $j \geq 2$, q_j is meant to quantize the vector $x - \sum_{s=1}^{j-1} q_s(x)$, which effectively places this strategy under the additive quantization umbrella. While RQ demonstrated superior performance in minimizing reconstruction errors, as shown in Section 5.4.1, it exhibits a fundamental limitation. Specifically, it allows for the possibility of multiple valid encodings for a single data point, which undermines the core principles of the generative model training procedure.

5.3.2 Generation

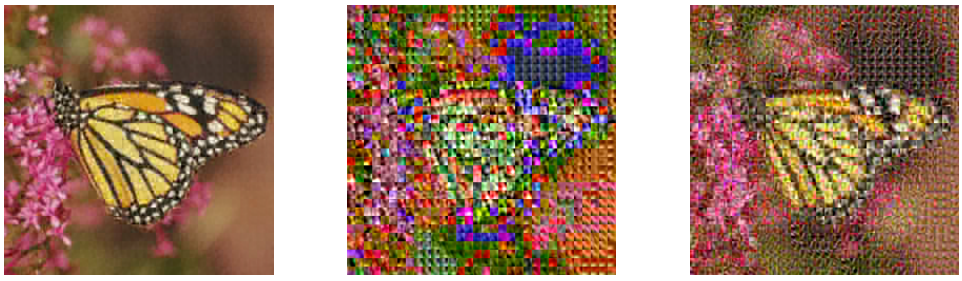
The lossy compression algorithm defines an encoding function $\mathcal{E}$ which maps an image x to a latent representation $z \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times \bar{c}}$. Ideally, redefining the training objective of the generative model to operate in the new latent space should be straightforward:

$$\min_{\theta} \|\epsilon - \epsilon_\theta(z_t, t)\|_2^2, \qquad z_t = \mathcal{E}(x_t). \tag{5.17}$$

This approach has been shown to be valid in the case of a latent space defined by a deep autoencoder [158], [160]. However, when relying on the shallow latent space defined by the proposed compression algorithm, the impossibility of learning anything useful is utterly clear. The causes of this limitation can be traced back to the definition of the encoding algorithm using VQ (and its other embodiments), and especially in the definition of the codewords using k-means as defined in Equation (5.10). Effectively, the encoding of an image patch with the assigned code $i \in \mathcal{I}$ impose a bi-univocal correspondence with the set of positive integers, which implies an inherent total ordering of the codewords $c_1, ..., c_{|\mathcal{I}|}$. The imposed ordering, which by the nature of k-means is fundamentally random, is not reflected in an ordering of the $\mathbb{R}^n$ space where codewords are defined, hence the addition of Gaussian noise to i will result in the random assignment of a new codeword $c_{\lceil i+\mathcal{N}(0,1) \rceil}$, while ideally the additive noise should lead to a codeword $c_j \propto c_i + \mathcal{N}(0, 1)$. The above can be intuitively appreciated by observing the effect of a constant addition on an encoded image in Figure 5.3b.

A useful interpretation is to consider the codewords as a finite and discrete set of categorical data, which brings us to the domain of categorical diffusion models, previously introduced in Section 5.2.2. Recent works on discrete state diffusion models [171]–[175] mainly propose to redefine the forward diffusion process by relying on a transition matrix which defines the probabilities of moving from one state to another. An interesting result of [171] is that it is possible to imitate a continuous space diffusion model by defining the discrete diffusion process in a way that will transit with higher probability to similar

states, introducing an ordinal bias. Along this line, is hereafter shown that the careful choice of an optimal ordering on $\mathcal{C}$ can be leveraged to approximate as a continuous process virtually with no additional work.



(A) $\mathcal{D}(\mathcal{E}(x))$ - only reconstruction noise

(B) $\mathcal{D}(\mathcal{E}(x)+1)$ before rearranging $\mathcal{I}$

(C) $\mathcal{D}(\mathcal{E}(x)+1)$ after rearranging $\mathcal{I}$

FIGURE 5.3: This example is obtained with compression schema OPQ-32-8-8, see Section 5.4.1 for details

.

Enforcing Pseudo Ordinality

For the scope of this work, an ordering of the code space $\mathcal{I}$ is sought, which satisfies an ordinality property on the codewords space $\mathcal{C}$ in $\mathbb{R}^n$:

$$\|c_{i+1} - c_i\|_2 < \|c_{i+j} - c_i\|_2 \quad \forall i \in \{1, \dots, |\mathcal{C}| - 1\},\ j \in \{2, \dots, |\mathcal{C}| - i\}, \tag{5.18}$$

$$\|c_{i-1} - c_i\|_2 > \|c_{i-j} - c_i\|_2 \quad \forall i \in \{2, \dots, |\mathcal{C}|\},\ j \in \{2, \dots, i - 1\}. \tag{5.19}$$

In this way, it is possible to enforce that the addition of noise leads to a transition to a code that is similar to the source code in proportion to the level of noise added. Clearly, with $n > 1$ and in the absence of additional constraints on the codebooks definition, the existence of a suitable mapping $\mathbb{R}^n \to \mathbb{R}$ that defines $\mathcal{I}$ cannot be guaranteed, and in most cases simply will not exist. More specifically, looking at the codewords as vectors in $\mathbb{R}^n$, there could be an arbitrary number of values j which satisfy the right part of the expression in Equation (5.18) and Equation (5.19), i.e., there could be multiple elements of $\mathcal{C}$, identified by different indexes j, which are equidistant from c_i.

Therefore, a relaxation of the desired property is introduced by imposing an ordering along the direction which maximizes the correlation between the elements of $\mathcal{C}$. This approach is borrowed from the technique of Principal Components Analysis (PCA) [176], widely used in the context of dimensionality reduction. Given a generic VQ codebook $\mathcal{C} \subset \mathbb{R}^n$, its principal component $v_0 \in \mathbb{R}^n$ can be conveniently computed as the eigenvector of the covariance matrix of $\mathcal{C}$ corresponding to its largest eigenvalue. The projection of $c_i \in \mathcal{C}$ along the principal component is hence computed as $c_i^0 = c_i v_0$, which also represents a mapping $\mathbb{R}^n \to \mathbb{R}$. Each element of $\mathcal{C}$ is assigned to an index $i \in \{1, ..., |\mathcal{C}|\}$ such that $c_i^0 > c_{i+1}^0$, or in simpler terms, as the indices that sort the vectors through their principal components. This approach does not strictly satisfy the properties in Equation (5.18) and Equation (5.19), nor it assures to maximize

the correlation between adjacent pairs, but it is guaranteed to be the best approximation possible achievable with a linear transformation. The latter claim is secured by the fact that PCA, in its linear version, is the linear component reduction technique that best preserves for explained variance. The additive noise in the diffusion process then causes a displacement along the maximum variance dimension, which in a categorical framework implies moving towards a highly correlated latent state, which has a resemblance with the continuous diffusion process. Figure 5.3c provides a powerful intuition of the achievement.

Categorical posterior distribution

Despite the promising results in training the DDPM in the latent space, thanks to the approximation introduced above by optimizing for the training objective of Equation (5.4), the sample quality of the decoded images is far from ideal. In particular, the presence of residual noise is evident as a consequence of the approximation of the continuum by exploiting the defined ordering, which as already mentioned is not perfect. A plausible overcome is to train a refinement model in RGB space to improve the sample quality. Despite not being optimal, this is far less expensive than training the whole generative model in the image space.

As an additional enhancement, the formulation of the inverse process is revised to better align with the inherent characteristics of VQ-encoded images. Building upon the training objective presented in Equation (5.5), learning the reverse process within the latent space by targeting the uncorrupted sample $z \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times \overline{c}}$ implies predicting the index i corresponding to the suitable codebook entry $c_i \in \mathcal{C}_j$ for each codebook $j \in (1, ..., \overline{c})$ at each spatial location. Instead of directly regressing the value of c_i, an alternative strategy is proposed, wherein the inverse process targets a categorical distribution for each codebook entry across all utilized codebooks. Assuming that all codebooks contain the same number of entries, the output of the reverse process can be parametrized as $\hat{z} \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times \overline{c} \times |\mathcal{C}|}$, so that the recovered sample $\tilde{z}$ could be obtained by sampling the corresponding categorical distribution at each spatial location. The forward diffusion process instead is kept unchanged, with z_t being defined by introducing noise to z in the index space.

Since the sampling operation is non-differentiable, in the training phase the Gumbel-softmax trick [177] is leveraged in order to approximate the sampling of the categorical distribution: this is achieved by perturbing the logits $\hat{z}^j$ for each category with Gumbel distributed noise $G_i \sim Gumbel(0, 1)$, and then approximating the one-hot vector representing the categorical choice by applying the softmax function to the perturbed logits. Mathematically, this can be expressed as:

$$\begin{gathered} \overline{z}^j = softmax\left(\frac{\hat{z}^j + G}{\tau}\right) \qquad j = 1, \ldots, \overline{c} \\ G \sim Gumbel(0, \mathrm{Id}). \end{gathered} \tag{5.20}$$

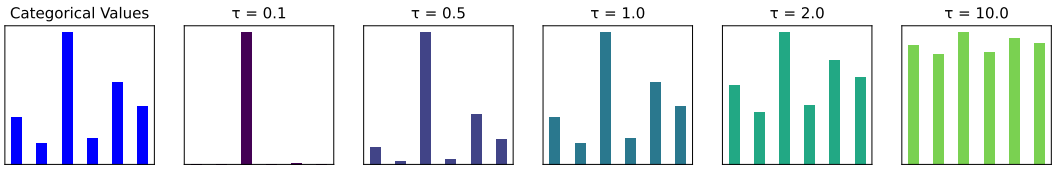


FIGURE 5.4: Effect of temperature smoothing of categorical distribution

Algorithm 3 Differentiable decoding denoised sample

Require: noisy sample z_t, time step t
Ensure: denoised sampled in image space $\tilde{x}_0$

```
predicted_codewords = empty()
```
$\hat{z}_0 \leftarrow$ `nn(`z_t, t`)` ▷ $\hat{z}_0 \in \mathbb{R}^{\frac{w}{s} \times \frac{h}{s} \times \overline{c} \times |\mathcal{C}|}$
for each channel $c \in (1, ..., \overline{c})$ **do**
  for all i,j **do**
    $\overline{z}_0[i, j, c] =$ `gumbel_softmax(`$\hat{z}_0[i, j, c]$`)`
    $\tilde{c} \leftarrow \hat{z}_0[i, j, c]\mathcal{C}_c$
    `predicted_codewords`$[i, j, c] \leftarrow \tilde{c}$
  end for
end for
return `vq_decode(prediced_codewords)`

Notably, this approximation allows us to explicitly sample from the codebook distribution instead of the index distribution. Consequently, the predicted codewords can be decoded in a fully differentiable manner, as demonstrated by the equation

$$\tilde{c} = [\overline{z}^1 \mathcal{C}_1, \cdots, \overline{z}^{\overline{c}} \mathcal{C}_{\overline{c}}] \tag{5.21}$$

and the decoded image $\tilde{x}$ can be recovered from $\tilde{c}$ depending on the utilized VQ specific (the complete procedure is schematized in Algorithm 3). By leveraging this formulation, the training objective can be computed in image space as in Equation (5.5), while still performing the generation and the gradient computation in the latent space. It is worth noticing that, in this way, the latent codes for z are never elicited in the training objective.

In order to sample from the trained model, the only distinction from the standard parametrization of the reverse process in the latent space is that deriving the next sample z_{t-1} requires obtaining $\tilde{z}$ by sampling from the categorical distributions derived from $\hat{z}$, which is achieved by the assigned codebook index from the corresponding categorical distribution. It is worth noticing that simply taking the index with the largest logit (argmax function), corresponds to using the mode of the categorical distribution, which contrasts with the definition of the inverse process.

Algorithm 4 Sampling

Require: noisy sample z_t, time step t
Ensure: $\tilde{z}_{t-1}$
  $\hat{z}_0 \leftarrow \texttt{nn}(z_t, t)$
  for all i,j **do**
    $\tilde{z}_0[i,j] \leftarrow \texttt{sample}(\hat{z}[i,j])$
  end for
  $\tilde{z}_{t-1} \leftarrow \texttt{reverse}(\tilde{z}_0, t-1)$ ▷ implementation of the reverse process parametrized by z_0
  return $\tilde{z}_{t-1}$

5.4 Experiments and Evaluation

5.4.1 Compression

In this section, the implementation detail of the VQ-based image encoding algorithm introduced in Section 5.3.1 is detailed and a qualitative analysis that led to picking the exact configuration used to train the generative models is discussed.

The size of the patch is fixed to $s = 8$ for all models, and the VQ training set is defined by extracting all the possible patches from the 18 high quality images of the McM dataset [178], which are augmented to 36 by vertical flipping each one, leading to just over 8.6 million patches. Once a VQ method (among VQ,PQ/OPQ and RQ) is picked, a distinct VQ encoder is trained for each channel in YCbCr space, by leveraging the implementations of the FAISS library [179] for maximum efficiency. Each channel can be encoded with a different bitrate, hence, the notation {VQ}-{Y_b-Cr_b-Cb_b} is adopted to uniquely identify the used combination of VQ type and assigned bitrates.

Encoding	C.R	Baboon		Peppers		Lenna	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
VQ-{8-8-8}	1:64	28.83	0.40	30.75	0.63	31.23	0.67
PQ-{32-8-8}	1:32	29.07	0.60	31.28	0.69	31.80	0.74
OPQ-{32-8-8}	1:32	29.12	0.62	31.56	0.72	32.35	0.78
RQ-{32-8-8}	1:23	29.22	0.64	31.97	0.75	32.81	0.81
VQ-f4 [158]	n.d	21.43	0.66	29.17	0.77	31.33	0.83
VQ-f8 [158]	n.d	18.31	0.37	26.66	0.68	27.16	0.73

TABLE 5.1: Evaluation of the proposed Lossy compression algorithm.

Each method is tested on three standard images of size 512×512, which have been used in the image compression literature for decades[1] [180]. The

[1] https://sipi.usc.edu/database/

performance is evaluated with two different quality measures. The Peak Signal-to-Noise Ratio (PSNR) is defined as

$$\begin{aligned} PSNR(x_1, x_2) &= 20log_{10}\left(\frac{255}{\sqrt{MSE(x_1, x_2)}}\right), \\ MSE(x_1, x_2) &= \frac{1}{hw}\sum_{i=1}^{h}\sum_{j=1}^{w}\|x_1(i,j) - x_2(i,j)\|_2^2 \end{aligned} \tag{5.22}$$

and takes values on all the real line, where the smaller the number, the worse the quality of the image is. The other is the Structural SIMilarity index (SSIM), which can be computed as

$$SSIM(x_1, x_2) = \frac{(2\mu_{x_1}\mu_{x_2} + c_1)(2\sigma_{x_1x_2} + c_2)}{(\mu_{x_1}^2 + \mu_{x_2}^2 + c_1)(\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2)},$$

where, for $i = 1, 2$, μ_{x_i} is the mean of x_i, σ_{x_i} is its standard deviation and $\sigma_{x_1x_2}$ is the covariance between the two images. The constants c_1 and c_2 are constants placed in order to avoid numerical issues. The SSIM of an image takes values in the interval $[0, 1]$, where a score of 0 implies an heavy dissimilarity between the images, while a score of 1 means the contrary.

The compression ratio is also reported, which, since the patch size is fixed, can be conveniently computed as

$$\mathcal{R}(q) = \frac{3 * 8 * s^2}{Y_b + Cr_b + Cb_b}, \tag{5.23}$$

where Y_b, Cr_b and Cb_b represents the number of bits needed to represent the pixel value for the respective channel.

The results reported in Table 5.1 are in line with what was expected: baseline VQ severely underperforms (especially in terms of SSIM), increasing the bitrate for the luminance channel yields far better results, while still achieving a high compression ratio. The absolute best results are obtained with RQ, which, however, turned out to be unsuitable for the intended application because of the commutative propriety of the sum, also being computational inefficient for the sequential nature of the encoding. In addition, basic PQ really does not make much sense because it produces highly redundant codebooks. For the above reasons, the choice opted for is to proceed with OPQ-32-8-8. The results obtained on the same benchmark by two configurations of the pretrained VAE used are reported in [158]. It is imperative to acknowledge that although the metrics under evaluation may not explicitly highlight this phenomenon, deep encoding tends to yield markedly more visually appealing images. This phenomenon can likely be attributed to the fact that shallow encoding introduces high-frequency and blocky artifacts, which are very noticeable to the human visual system, while deep encoding produces high-level errors that are not obvious at a glance unless a side-by-side comparison is made.

5.4.2 Generation

In this section, an experimental evaluation of the proposed approach is proposed, evaluating the performances in a popular benchmark for unconditional image generation. As introduced in Section 5.2.1, the definition of a metric for evaluating generative models is still an open problem, hence, for the remainder of this section the FID score is used, considering that to the current date it represents the most popular metric in the literature.

FIGURE 5.5: Sampling steps ($t = 1000, 900, ..., 0$) of the model trained with the categorical parametrization of the reverse process.

Experimental setup

The generative performances are assessed in a single-class unconditional scenario on two subsets of the LSUN dataset [181] which are extremely popular in the context: the *cat* subset and the *church* one. The first is a large dataset of over 1.6 million images of cats acquired from a wide variety of sources. This is a very complicated dataset due to the extreme variability in the appearance and composition of the images, caused in part by the nature of the subject. In contrast, the dataset of churches is smaller with only 125 thousand images, and is considerably easier to handle because of the chromatic and geometric regularity of the buildings represented. All the experiments are conducted with an image resolution before compression of 512×512, which means a latent representation of $64 \times 64 \times 6$ using the OPQ-{32-8-8} encoding.

Model	FID Score	
	LSUN-cat	LSUN-church
Continuous	14.01	12.46
Continuous + Refiner	**11.92**	**10.05**
Categorical Rev. ($\tau = 1.0$)	13.97	12.06
Categorical Rev. ($\tau = 2.2$)	13.75	-

TABLE 5.2: Evaluation of unconditional image generation on LSUN datasets

A common setup for all the experiments is fixed with the following characteristics: the denoiser model is defined as a U-Net model with 4 down/up sampling stages and attention only at the bottleneck level, in order to reduce

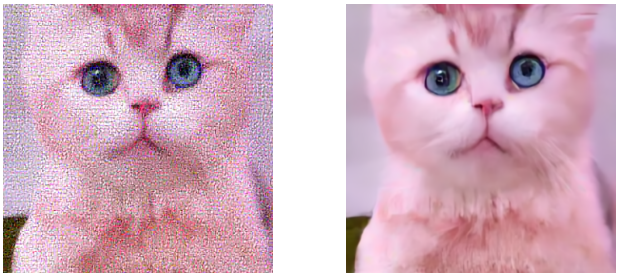
the memory footprint. In total, the model has around 50 Million parameters, which is a small number if compared to what is commonly used. The utilized experimental setup is based on the Pytorch implementation by Phil Wang of the original DDPM paper, with the addition of Min-SNR weighting [182], which, by penalizing the loss terms proportionally to the inverse of the magnitude of the noise added at timestep t,further discourages the model from focusing on minor details. Self-conditioning [183] is also implemented, which condition the denoising process based on the previous estimate of the recovered sample $\tilde{x}_0$. All the evaluated models are trained for 1 million steps with batch size of 64 on 2 Nvidia v100 GPUs, which takes around 10 days. This represents an extremely tight computational budget for this kind of application: in comparison, in the original DDPM a U-Net configuration with either 114 or 256 Million parameters (approximately 2–5 times larger than ours) is leveraged, and train for the equivalent of over 100 v100 days, which is over 10 times the available computational budget for this work. An additional difference is that, thanks to the efficiency of the proposed encoding, it becomes possible to generate samples of size 512×512, which after encoding with the chosen setup (OPQ-{32-8-8}) corresponds to latent codes of size $64 \times 64 \times 6$. Other existing methods, such as the original DDPM [153], typically use samples of size 256×256.

Continuous-style diffusion model

FIGURE 5.6: Samples generated with continuous formulation (after refiner)

This evaluation starts by analyzing the case in which the forward and backward processes are kept unchanged compared to the continuous case. Hence, the objective in Equation (5.4) is used for optimization, while leveraging the codeword-to-index assignment introduced in Section 5.3.2. As anticipated, the generative process is able to infer and reproduce the high-level structure of real images and model the textures and details. Where this approach fails are the last steps of the inverse process, since the ambiguity introduced by the approximation to the continuum seems to limit the overall definition of the finest details in the image, resulting in an extremely noisy generated image.

As an impromptu solution, it is possible to train a refinement model to improve the quality of the generated image $\tilde{x} = \mathcal{D}(\tilde{z})$ after it has been decoded to RGB space. The refiner is based on the same U-Net architecture with a smaller configuration with only 1.7M parameters, and it is trained in fully supervised fashion by synthetically generating noisy images x_n by slightly corrupting their encoded representation: $x_n = \mathcal{D}(\mathcal{E}(x) + \gamma)$, where γ is a scaled-down Gaussian



(A) Generated Image (B) After Refinement

noise. The objective function for the refiner is a combination of $L1$ and SSIM losses, which easily achieves a PSNR of over 30 on both datasets. The clear downside of this solution is the necessity of training a custom refiner for each dataset and the increased cost of generating new images.

Categorical reverse process

FIGURE 5.8: Samples generated with categorical formulation (no refiner needed).

Comparatively, the performances are also evaluated when leveraging the formulation introduced in Section 5.3.2 for the definition of the reverse process. At training time the objective function of Equation (5.5) is used, together with the differentiable decoding of $\tilde{x}_0$ derived in Equation (5.21). From this analysis, it emerges how it is preferable to add a second term to the loss in order to condition the prediction of z_0 codes based on those assigned by the VQ encoding, which are otherwise not explicitly in the derivation of $\tilde{x}_0$.

The complete training objective used is

$$\begin{aligned} \mathcal{L}_{cat}(\theta) &= ||x_0 - \tilde{x}_0^\theta||_2^2 + ||\epsilon - \epsilon_z^\theta||_2^2 \\ \epsilon_z^\theta &= (a_t * z_t - \tilde{z}_0^\theta)/b_t, \end{aligned} \tag{5.24}$$

where

- ϵ_z^θ is the derivation for the predicted noise in latent space, obtained from the prediction for the uncorrupted image $\tilde{z}_0^\theta$ sampled from $\hat{z}_0$ using the Gumbel-softmax trick;
- a_t and b_t are time-dependent constants values, that are derived from the detailed definition of the diffusion process in [153], the details of which are omitted for simplicity;

- the θ superscript is added to indicate values that are a function of model parameters.

(A) argmax (B) $\gamma = 0.1$ (C) $\gamma = 1.0$ (D) $\gamma = 2.0$

FIGURE 5.9: Effects of distribution smoothing on $\hat{z}_0$.

In this configuration, satisfactory results are obtained without resorting to the refiner model. One interesting thing to study is the introduction of a temperature smoothing at inference time in the generative process (i.e, deriving z_{t-1} given z_t). Since z_{t-1} is obtained from the current estimate for $\tilde{z}_0$, a temperature factor γ can be introduced in the sampling of $\tilde{z}_0$ from $\hat{z}_0$

$$\tilde{z}_0 \sim softmax\left(\hat{z}_0/\gamma\right) \tag{5.25}$$

The effect of the introduction of γ can be qualitatively appreciated in Figure 5.9. In particular, for $\gamma \to 0$ the sampling operation corresponds to the *argmax* function and essentially defines the mode of the categorical distribution, while for larger values of γ, the distribution shifts towards a uniform distribution.

Chapter 6

Summary

To draw all-encompassing conclusions from such a broad and articulate work is contrary to the spirit in which this was conceived. Therefore, below, a summary is provided hereafter of the goals achieved in each of the domains that have been addressed in this thesis.

- The decision to develop MTL models for automotive applications can rightly be considered a winning strategy. The trade-off of performance on single tasks and inference costs can certainly be considered advantageous, particularly in the case of DMS where it has been experimentally observed that performance improvement over the single-task baseline is possible in some cases. One of the main limitations to the application of this paradigm has been confirmed to be the availability of datasets with multi-task labels suitable for the purpose.
- One of the main contributions of the work on DMS is identified in the use of pseudo-annotated data for the addition of tasks not present in the base dataset. Although this is a relatively simple method, with careful selection of the appropriate training strategy, the goal was fully achieved.
- The work on ADAS presents a problem of a more complex nature because of the important differences between the tasks that were solved with the multi-task model. The formulations chosen for the object and lane detection tasks yielded satisfactory performance even with a relatively simple optimization strategy. Inference benchmarks ultimately validated the goodness of the multi-task approach here as well.
- Drawing on traditional data compression techniques to apply them to the latest deep learning models has proven to be a valid, exciting and insightful line of research. This line of research is validated by the results obtained on two different problems.
- Approximating the attention matrix with DCT, while involving a major relaxation in the formulation, proved to be a simple and effective approach. The performance obtained on the NLP benchmark, when compared to the large savings in computation and memory obtained, put it ahead of the literature competitors analyzed.
- The chapter on generative models offers probably the most aggressive and cutting-edge contribution among those presented in this thesis. Using a non-deep compression scheme in this scenario is a first. The generative performance achieved, when analyzed in relation to the negligible computational

resources available, certainly represents a step in the desired direction. The observations and the approach identified for extending the generation process by introducing pseudo-ordinality are an original and rich contribution.

Bibliography

[1] C. Scribano, G. Franchini, M. Prato, and M. Bertogna, "Dct-former: Efficient self-attention with discrete cosine transform," *Journal of Scientific Computing*, vol. 94, no. 3, p. 67, 2023.

[2] C. Scribano, G. Franchini, I. S. Olmedo, and M. Bertogna, "Cerberus: Simple and effective all-in-one automotive perception model with multi task learning," *arXiv preprint arXiv:2210.00756*, 2022.

[3] C. Scribano, D. Pezzi, G. Franchini, and M. Prato, "Denoising diffusion models on model-based latent space," *Algorithms*, vol. 16, no. 11, p. 501, 2023.

[4] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile gpus," *arXiv preprint arXiv:1907.06724*, 2019.

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[6] D. Cristinacce and T. Cootes, "Automatic feature localisation with constrained local models," *Pattern Recognition*, vol. 41, no. 10, pp. 3054–3067, 2008.

[7] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1867–1874.

[8] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[9] X. Dong, Y. Yan, W. Ouyang, and Y. Yang, "Style aggregated network for facial landmark detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 379–388.

[10] W. Wu, C. Qian, S. Yang, Q. Wang, Y. Cai, and Q. Zhou, "Look at boundary: A boundary-aware face alignment algorithm," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2129–2138.

[11] X. Wang, L. Bo, and L. Fuxin, "Adaptive wing loss for robust face alignment via heatmap regression," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6971–6981.

[12] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3476–3483.

[13] X. Guo, S. Li, J. Yu, *et al.*, "Pfld: A practical facial landmark detector," *arXiv preprint arXiv:1902.10859*, 2019.

[14] M. Kopaczka, J. Schock, and D. Merhof, "Super-realtime facial landmark detection and shape fitting by deep regression of shape model parameters," *arXiv preprint arXiv:1902.03459*, 2019.

[15] R. Liu, J. Lehman, P. Molino, *et al.*, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.

[16] Z. Zhou, H. Li, H. Liu, N. Wang, G. Yu, and R. Ji, "Star loss: Reducing semantic ambiguity in facial landmark detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 475–15 484.

[17] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, "Wing loss for robust facial landmark localisation with convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2235–2245.

[18] Z. Xu, B. Li, Y. Yuan, and M. Geng, "Anchorface: An anchor-based facial landmark detector across large poses," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 3092–3100.

[19] A. P. Fard and M. H. Mahoor, "Acr loss: Adaptive coordinate-based regression loss for face alignment," in *2022 26th International Conference on Pattern Recognition (ICPR)*, IEEE, 2022, pp. 1807–1814.

[20] X. Dong, S.-I. Yu, X. Weng, S.-E. Wei, Y. Yang, and Y. Sheikh, "Supervision-by-registration: An unsupervised approach to improve the precision of facial landmark detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and vision computing*, vol. 47, pp. 3–18, 2016.

[22] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 146–155.

[23] W. Wu, C. Qian, S. Yang, Q. Wang, Y. Cai, and Q. Zhou, "Look at boundary: A boundary-aware face alignment algorithm," in *CVPR*, 2018.

[24] Y. Liu, H. Shi, H. Shen, Y. Si, X. Wang, and T. Mei, "A new dataset and boundary-attention semantic segmentation for face parsing.," in *AAAI*, 2020, pp. 11 637–11 644.

[25] N. Ruiz, E. Chong, and J. M. Rehg, "Fine-grained head pose estimation without keypoints," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 2074–2083.

[26] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang, "Fsa-net: Learning fine-grained structure aggregation for head pose estimation from a single image," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1087–1096.

[27] V. Albiero, X. Chen, X. Yin, G. Pang, and T. Hassner, "Img2pose: Face alignment and detection via 6dof, face pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7617–7627.

[28] M. Roth and D. M. Gavrila, "Dd-pose - a large-scale driver head pose benchmark," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 927–934. DOI: 10.1109/IVS.2019.8814103.

[29] W. W. Wierwille and L. A. Ellsworth, "Evaluation of driver drowsiness by trained raters," *Accident Analysis & Prevention*, vol. 26, no. 5, pp. 571–581, 1994.

[30] S. Junaedi and H. Akbar, "Driver drowsiness detection based on face feature and perclos," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1090, 2018, p. 012 037.

[31] B. Reddy, Y.-H. Kim, S. Yun, C. Seo, and J. Jang, "Real-time eye blink detection using facial landmarks," *IEEE CVPRW*, 2017.

[32] G. Pan, L. Sun, Z. Wu, and S. Lao, "Eyeblink-based anti-spoofing in face recognition from a generic webcamera," in *2007 IEEE 11th international conference on computer vision*, IEEE, 2007, pp. 1–8.

[33] F. Song, X. Tan, X. Liu, and S. Chen, "Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients," *Pattern Recognition*, vol. 47, no. 9, pp. 2825–2838, 2014.

[34] R. Fusek, "Pupil localization using geodesic distance," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11241 LNCS, pp. 433–444, 2018. DOI: 10.1007/978-3-030-03801-4_38.

[35] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, "Yawdd: A yawning detection dataset," in *Proceedings of the 5th ACM multimedia systems conference*, 2014, pp. 24–28.

[36] R. Ghoddoosian, M. Galib, and V. Athitsos, "A realistic dataset and baseline temporal model for early drowsiness detection," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.

[37] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.

[38] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3614–3633, 2021.

[39] L. Celona, L. Mammana, S. Bianco, and R. Schettini, "A multi-task cnn framework for driver face monitoring," in *2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*, IEEE, 2018, pp. 1–4.

[40] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[42] C.-H. Weng, Y.-H. Lai, and S.-H. Lai, "Driver drowsiness detection via a hierarchical temporal deep belief network," in *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III 13*, Springer, 2017, pp. 117–133.

[43] W. Kim, W.-S. Jung, and H. K. Choi, "Lightweight driver monitoring system based on multi-task mobilenets," *Sensors*, vol. 19, no. 14, p. 3200, 2019.

[44] D. Yang, X. Li, X. Dai, *et al.*, "All in one network for driver attention monitoring," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 2258–2262.

[45] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[47] Z.-H. Feng, J. Kittler, M. Awais, and X.-J. Wu, "Rectified wing loss for efficient and robust facial landmark localisation with convolutional neural networks," *International Journal of Computer Vision*, vol. 128, no. 8, pp. 2126–2145, 2020.

[48] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[49] P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi, *et al.*, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 285–21 296, 2020.

[50] B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson, "There are many consistent explanations of unlabeled data: Why you should average," *arXiv preprint arXiv:1806.05594*, 2018.

[51] U. Ozbulak, *Pytorch cnn visualizations*, https://github.com/utkuozbulak/pytorch-cnn-visualizations, 2019.

[52] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[53] H. Wang, Z. Wang, M. Du, *et al.*, "Score-cam: Score-weighted visual explanations for convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 24–25.

[54] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[55] 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)*, Ieee, vol. 1, 2005, pp. 886–893.

[56] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[57] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[58] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

[59] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[60] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[61] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[62] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[63] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[64] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[65] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[66] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[67] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[68] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: An instance segmentation approach," in *2018 IEEE intelligent vehicles symposium (IV)*, IEEE, 2018, pp. 286–291.

[69] X. Li, J. Li, X. Hu, and J. Yang, "Line-cnn: End-to-end traffic line detection with line proposal unit," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 248–258, 2019.

[70] Z. Chen, Q. Liu, and C. Lian, "Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection," in *2019 IEEE intelligent vehicles symposium (IV)*, IEEE, 2019, pp. 2563–2568.

[71] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Keep your eyes on the lane: Real-time attention-guided lane detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 294–302.

[72] L. Liu, X. Chen, S. Zhu, and P. Tan, "Condlanenet: A top-to-down lane detection framework based on conditional convolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3773–3782.

[73] T. Zheng, Y. Huang, Y. Liu, *et al.*, "Clrnet: Cross layer refinement network for lane detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 898–907.

[74] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz, "Key points estimation and point instance segmentation approach for lane detection," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[75] Z. Qu, H. Jin, Y. Zhou, Z. Yang, and W. Zhang, "Focus on local: Detecting lane marker from bottom up via key point," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 122–14 130.

[76] J. Wang, Y. Ma, S. Huang, *et al.*, "A keypoint-based global association network for lane detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1392–1401.

[77] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE intelligent vehicles symposium (IV)*, IEEE, 2018, pp. 1013–1020.

[78] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[80] Y. Qian, J. M. Dolan, and M. Yang, "Dlt-net: Joint detection of drivable areas, lane lines, and traffic objects," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4670–4679, 2019.

[81] F. Yu, H. Chen, X. Wang, *et al.*, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[82] D. Wu, M. Liao, W. Zhang, and X. Wang, "Yolop: You only look once for panoptic driving perception," *arXiv preprint arXiv:2108.11250*, 2021.

[83] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[84] C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang, and J. Yuan, "Yolopv2: Better, faster, stronger for panoptic driving perception," *arXiv preprint arXiv:2208.11434*, 2022.

[85] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475.

[86] D. Vu, B. Ngo, and H. Phan, "Hybridnets: End-to-end perception network," *arXiv preprint arXiv:2203.09035*, 2022.

[87] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[88] J. Wang, Q. Wu, and N. Zhang, "You only look at once for real-time and generic multi-task," *arXiv preprint arXiv:2310.01641*, 2023.

[89] Q.-H. Che, D.-P. Nguyen, M.-Q. Pham, and D.-K. Lam, "Twinlitenet: An efficient and lightweight model for driveable area and lane segmentation in self-driving cars," in *2023 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, IEEE, 2023, pp. 1–6.

[90] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 552–568.

[91] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[92] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[93] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[94] J. Fu, J. Liu, H. Tian, *et al.*, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3146–3154.

[95] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[96] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[97] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2403–2412.

[98] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578.

[99] X. Li, W. Wang, L. Wu, *et al.*, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 002–21 012, 2020.

[100] D. Zhou, J. Fang, X. Song, *et al.*, "Iou loss for 2d/3d object detection," in *2019 International Conference on 3D Vision (3DV)*, IEEE, 2019, pp. 85–94.

[101] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 12 993–13 000.

[102] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[103] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[104] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: https://aclanthology.org/P11-1015.

[105] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[106] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[107] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[108] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[109] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in neural information processing systems*, vol. 2, 1989.

[110] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[111] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[112] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," *arXiv preprint arXiv:1705.00108*, 2017.

[113] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[114] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[115] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[116] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2020.

[117] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[118] F. N. Iandola, A. E. Shaw, R. Krishna, and K. W. Keutzer, "Squeezebert: What can computer vision teach nlp about efficient neural networks?" *arXiv preprint arXiv:2006.11316*, 2020.

[119] Y. Wu, M. Schuster, Z. Chen, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[120] Y. Zhu, R. Kiros, R. Zemel, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[121] P. Izsak, M. Berchansky, and O. Levy, "How to train bert with an academic budget," *arXiv preprint arXiv:2104.07705*, 2021.

[122] A. M. Raid, W. M. Khedr, M. A. El-dosuky, and W. Ahmed, "Jpeg image compression using discrete cosine transform a survey," *arXiv preprint arXiv:1405.6147*, 2014.

[123] X. Shao and S. G. Johnson, "Type-II/III DCT/DST algorithms with reduced number of arithmetic operations," *Signal Processing*, vol. 88, no. 6, pp. 1553–1564, 2008.

[124] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[125] K. M. Choromanski, V. Likhosherstov, D. Dohan, *et al.*, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021, p. 636.

[126] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020, p. 1838.

[127] Y. Xiong, Z. Zeng, R. Chakraborty, *et al.*, "Nyströmformer: A Nyström-based algorithm for approximating self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 14 138–14 148.

[128] J. Lu, J. Yao, J. Zhang, *et al.*, "Soft: Softmax-free transformer with linear complexity," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 297–21 309, 2021.

[129] H. Ren, H. Dai, Z. Dai, *et al.*, "Combiner: Full attention transformer with sparse computation cost," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 470–22 482, 2021.

[130] T. Nguyen, V. Suliafu, S. Osher, L. Chen, and B. Wang, "Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 449–29 463, 2021.

[131] C. Wu, F. Wu, T. Qi, Y. Huang, and X. Xie, "Fastformer: Additive attention can be all you need," *arXiv preprint arXiv:2108.09084*, 2021.

[132] S. Jaszczur, A. Chowdhery, A. Mohiuddin, *et al.*, "Sparse is enough in scaling transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9895–9907, 2021.

[133] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[134] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention for transformer models," in *International Conference on Machine Learning*, 2021, pp. 10 183–10 192.

[135] C. Zhu, W. Ping, C. Xiao, *et al.*, "Long-short transformer: Efficient transformers for language and vision," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 723–17 736, 2021.

[136] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré, "Scatterbrain: Unifying sparse and low-rank attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 413–17 426, 2021.

[137] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 261–24 272, 2021.

[138] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "Fnet: Mixing tokens with fourier transforms," *arXiv preprint arXiv:2105.03824*, 2021.

[139] W. You, S. Sun, and M. Iyyer, "Hard-coded gaussian attention for neural machine translation," *arXiv preprint arXiv:2005.00742*, 2020.

[140] B. Jacob, S. Kligys, B. Chen, *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[141] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, 2015, pp. 1135–1143.

[142] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[143] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from jpeg," *Advances in Neural Information Processing Systems*, vol. 31, pp. 3933–3944, 2018.

[144] A. Dziedzic, J. Paparrizos, S. Krishnan, A. Elmore, and M. Franklin, "Band-limited training and inference for convolutional neural networks," in *International Conference on Machine Learning*, 2019, pp. 1745–1754.

[145] B. Rajesh, M. Javed, S. Srivastava, *et al.*, "Dct-compcnn: A novel image classification network using jpeg compressed dct coefficients," in *2019 IEEE Conference on Information and Communication Technology*, 2019, pp. 1–6.

[146] K. Xu, M. Qin, F. Sun, Y. Wang, Y.-K. Chen, and F. Ren, "Learning in the frequency domain," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1740–1749.

[147] S. F. dos Santos, N. Sebe, and J. Almeida, "The good, the bad, and the ugly: Neural networks straight from jpeg," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 1896–1900.

[148] J. Makhoul, "A fast cosine transform in one and two dimensions," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 1, pp. 27–34, 1980.

[149] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.

[150] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[151] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[152] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3505–3506.

[153] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[154] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[155] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[156] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[157] G. Parmar, R. Zhang, and J.-Y. Zhu, "On aliased resizing and surprising subtleties in gan evaluation," in *CVPR*, 2022.

[158] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[159] A. Kuznetsova, H. Rom, N. Alldrin, *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[160] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[161] M. Goldberg, P. Boucher, and S. Shlien, "Image compression using adaptive vector quantization," *IEEE Transactions on Communications*, vol. 34, no. 2, pp. 180–187, 1986. DOI: 10.1109/TCOM.1986.1096503.

[162] N. Nasrabadi and R. King, "Image coding using vector quantization: A review," *IEEE Transactions on Communications*, vol. 36, no. 8, pp. 957–971, 1988. DOI: 10.1109/26.3776.

[163] I. Garg, S. S. Chowdhury, and K. Roy, "Dct-snn: Using dct to distribute spatial information over time for low-latency spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 4671–4680.

[164] R. Gray, "Vector quantization," *IEEE Assp Magazine*, vol. 1, no. 2, pp. 4–29, 1984.

[165] R. Gray and D. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, 1998. DOI: 10.1109/18.720541.

[166] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[167] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 744–755, 2014. DOI: 10.1109/TPAMI.2013.240.

[168] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[169] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 931–938.

[170] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11 259–11 273, 2010, ISSN: 1424-8220. DOI: 10.3390/s101211259. [Online]. Available: https://www.mdpi.com/1424-8220/10/12/11259.

[171] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. Van Den Berg, "Structured denoising diffusion models in discrete state-spaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 981–17 993, 2021.

[172] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, "Argmax flows and multinomial diffusion: Learning categorical distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 454–12 465, 2021.

[173] A. Campbell, J. Benton, V. De Bortoli, T. Rainforth, G. Deligiannidis, and A. Doucet, "A continuous time framework for discrete denoising models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 266–28 279, 2022.

[174] A. Vahdat, K. Kreis, and J. Kautz, "Score-based generative modeling in latent space. 2021," *arXiv preprint arXiv:2106.05931*, 2021.

[175] Z. Tang, S. Gu, J. Bao, D. Chen, and F. Wen, "Improved vector quantized diffusion models," *arXiv preprint arXiv:2205.16007*, 2022.

[176] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20 150 202, 2016. DOI: 10.1098/rsta.2015.0202.

[177] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[178] L. Zhang, X. Wu, A. Buades, and X. Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *Journal of Electronic imaging*, vol. 20, no. 2, pp. 023 016–023 016, 2011.

[179] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[180] L. Roberts, "Picture coding using pseudo-random noise," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 145–154, 1962.

[181] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[182] T. Hang, S. Gu, C. Li, *et al.*, "Efficient diffusion training via min-snr weighting strategy," *arXiv preprint arXiv:2303.09556*, 2023.

[183] T. Chen, R. Zhang, and G. Hinton, "Analog bits: Generating discrete data using diffusion models with self-conditioning," *arXiv preprint arXiv:2208.04202*, 2022.