Journal of Intelligent Systems and Internet of Things
Vol. 15, No. 01, PP. 37-52, 2025
DOI: https://doi.org/10.54216/JISIoT.150104

Exploring the Use of Deep Learning Models for Image Compression in Embedded Systems: Encoder and Decoder Architectures

Abhiram Potlapalli[1,*], Seetharam Khetavath[2]
[1]Research Scholar, Department of ECE, Chaitanya deemed to be University, Hyderabad, India
[2]Professor & Head, Department of ECE, Chaitanya deemed to be University, Hyderabad, India
Emails: abhiram.potlapalli@gmail.com; seetharamkhetavath@gmail.com

Abstract

With the growing demand for efficient image processing in embedded systems, the exploration of deep learning-based image compression methods has emerged as a promising avenue. Traditional image compression techniques, such as JPEG and PNG, face challenges in achieving optimal performance for constrained environments due to their reliance on handcrafted algorithms and limited adaptability. This study investigates the use of deep learning models for image compression tailored to embed systems, focusing on encoder and decoder architectures. By leveraging convolutional neural networks (CNNs) and variational auto encoders (VAEs), we design lightweight models capable of achieving high compression ratios while maintaining visual fidelity. The research emphasizes computational efficiency, ensuring compatibility with the resource constraints of embedded hardware. Key contributions include the development of streamlined architectures optimized for low memory and power usage, along with a comprehensive evaluation of compression quality, reconstruction accuracy, and real-time performance. The results demonstrate that deep learning-based approaches can outperform traditional methods in terms of adaptability and efficiency, paving the way for their integration into next-generation embedded systems.

Received: June 25, 2024 Revised: September 20, 2024 Accepted: December 22, 2024

Keywords: Deep Learning; Compression; Embedded Systems; Encoder and Decoder

1. Introduction

At present, we are surrounded by devices such as mobile phones, computers, Internet of Things (IoT) devices, smart cameras, etc. that are used for capturing and processing images in real time. The need to acquire and transmit such images instantaneously requires the use of efficient image compression strategies embedded in the systems themselves. Traditionally, image compression algorithms have been developed using various signal processing models that are to be implemented using specific dedicated digital signal processing hardware. Over time, these models have become more advanced, which, in turn, has increased the complexity of the hardware required to achieve real-time performance. For example, simple models such as transform coding and Huffman coding were replaced by JPEG applications, and later, render farms were presented, and discrete cosine transform was presented to reduce file size. In recent years, there has been a trend towards exploring and using deep learning-based models for image compression. All this recent work on deep learning for image compression uses large models. Such parameter-heavy models are not suitable for embedded systems that require low latency and real-time performance. The use of large parameter models makes the deployment of such approaches prohibitive due to the memory and compute restrictions on the hardware. This has motivated the search for compact models, and there has been other work that studies resource-efficient deep learning models for image compression. Such models are limited to larger embedded systems, like a GPU or FPGA—often used. We

propose studying the use of compact deep learning-based models for image compression whose performance allows the models to be run on smaller embedded systems [1].

The continuous growth of multimedia data that includes images, videos, and speech, among others, has been produced since the availability of digital technologies. Image and video data have become the field of interest for many researchers, leading to image and video processing applications, implemented algorithms, and hardware systems. With the growing demand for smart cameras in automotive safety, surveillance, computer vision, and industrial control systems, high-resolution imaging is vital. It has been demonstrated that many of these applications require higher frame rates and higher resolutions. However, the drawback is the transmission of high data content; most smart cameras, network bandwidth, and data retention increase system power consumption, energy consumption, computational requirements, and also degrade image quality. Furthermore, image encoding and decoding algorithms, signal processing methods, and hardware systems should be studied to achieve high image compression efficiency. To address this issue, we use deep learning neural network models to explore image compression techniques as an effective way for embedded systems, due to their inherent parallelism and energy-optimized structures. The deep learning neural network techniques developed in the last decade achieve state-of-the-art performance in image and video encoding and compression, setting the latest image coding standard [2].

The main objectives of this paper are to develop different deep learning architectures for image compression using the encoder-decoder structure and to provide extensive analysis and explanations of why the proposed deep neural networks can achieve image compression. In summary, the research work in this project is comprised of the following five main objectives: 1) To construct and present the structure of different deep learning models with training algorithms for compressing raw images and videos. 2) To investigate the trade-offs of the proposed deep learning algorithms for real applications by comparing different model complexities and strengths. 3) To demonstrate the superior performance of the proposed deep learning models using large-scale benchmark datasets. 4) To explore the implementation of the best model on fixed-point optimization and evaluate the effectiveness of the optimized models on different embedded systems. 5) To present possible improvements in future research work based on the lessons learned from this project.

The compression, separation, and restoration of digital information by and from storage and transmission channels is known as data encoding and decoding. In some cases, the digital information concerns two-dimensional images. Image compression is a process for converting a digital image into a form that uses fewer bytes of information than the original digital image while providing superior image quality representation. Fundamentally, the transformation encompasses two major methods: loss and lossless. Lossless encoding mechanisms or algorithms guarantee that no information is lost by preserving all data during encoding and restoring the original image during decoding. On the other hand, while the loss image compression schemes achieve a high compression ratio through the discarding of irrelevant or non-essential information, the original high-quality image may not be restored. This is a result of techniques discarding information based on a threshold determination of little impact on the quality of the reconstructed image [3].

Within the context of data encoding, one can find various ways in which the data is stored, based on the needs. Here we will talk specifically about image storage. The most common form is lossless storage. In this format, the image is stored in full, so there is no information loss. Nevertheless, depending on the need, this type of storage may be wasteful, especially for more complex formats. In contrast, there is the loss approach. In this format, not all original data is stored; there is information loss, but it is possible to reconstruct an image that seems quite similar to the previous one. Loss formats are usually able to offer considerable size reduction by discarding unnecessary data, and importantly, data that affects little or nothing in the final image. Although this type of format can be disadvantageous for certain applications, the reduction in size makes the application of loss formats viable in other situations.

In the context of image compression, lossless compression has the most consistent use. This type of compression is commonly used in images that need to store all their information, and that information is sensitive to loss. This type of compression is usually sought when the objective is to transfer the image from one place to another without data loss. Between the characteristics of these two types of compression, applications are normally used in different circumstances. However, it is necessary to highlight the importance of moving between one form of compression and another quickly and with good performance, especially at the hardware level. With this in mind, we have the main objective of investigating the construction of deep learning models to act as encoders and decoders of images in loss compression format [4].

Image compression is a necessary step before transferring or storing images due to the large amount of information in them. Digital images can be compressed by reducing the size of the file or the coding efficiency more generally by encoding it with fewer bits. Larger files require more storage and larger bandwidth, and also slower procedures that work with such files. When the size of the file is reduced, the quality of the image also decreases. Loss image

compression, in exchange for a small amount of loss, can achieve high compression. Compression is becoming more important in embedded systems due to real-time processes during capture, transmission, storage, and display of the image [5].

There are many standard and non-standard techniques for lossless and loss image compression. Lossless compression means that the compression algorithm does not lose any information in the compression-decompression process. Loss compression removes some data, so after decompression, the file is not the same, although it is quite similar to the original one. Studies focus on lossless compression techniques and reaching a reasonable result after removing unnecessary information in the file using loss compression techniques with the smallest reasonable amount of loss.

2. Deep Learning in Image Compression-Methodology

In image compression, a signal is coded such that it can be accurately and efficiently represented with few data units. The most commonly used image compression schemes are based on discrete cosine transform or wavelet transform. These schemes have encoder and decoder parts and are called codecs. During compression, the encoder makes the signal less redundant and more efficient. Later, during decompression, the decoder reverses the loss in information made by the encoder such that the original signal is approximately reconstructed. The codec is able to do this through a predefined set of transforms, entropy coding, and quantization. The quality of the reconstructed signal mainly depends on how well the quantization is devised by the codec. A fine quantization leads to better quality of the signal with a larger file size, while a coarse quantization leads to lower quality of the signal with a smaller file size. The image quality is often compromised in consumer electronics for using these codecs for efficient compression. These codecs are also not scalable in terms of bit rate and picture resolution. The flexibility in changing the code design is time-consuming, and the main aim is to find a single design that performs the compression with acceptable trade-offs in rate-distortion-complexity metrics. With the rapid increase of multimedia data, there is a demand for novel image compression schemes that could outperform the traditional codecs in terms of better rate-distortion-complexity trade-offs [6].

Deep convolutional networks are increasingly being used for solving various image-related computer vision tasks like image classification, object detection, semantic segmentation, and image super-resolution. Datasets collected for one computer vision problem have rich visual statistics and can be generally used for solving other related problems. That is, features learned for solving one problem can be reused for solving another problem. In this paper, we explore the ability to use pre-trained convolutional neural network features for solving another important problem in image processing called image compression. In particular, we explore the use of these features in image compression for creating an efficient encoder and decoder that could perform subjective and objective evaluations.

Deep learning is a class of machine learning algorithms that is based on learning data representations, as opposed to "shallow" learning-based models. Learning can be supervised or unsupervised. It is made up of many layers of non-linear information processing units solving various problems. It is characterized by a lot of computations involving processing large amounts of data and optimizing complex functions on that data. The name "deep learning" describes the general idea, rather than a technique applied in a specific algorithm. It is an artificial intelligence function that imitates the workings of the human brain in processing data for use in detecting and classifying images. Deep learning architectures can be wavelengths long, containing hidden layers instead of the norm [7].

Deep learning is typically hard work for systems to learn from training that need to use large amounts of labeled data to provide the information necessary for the system. This process is more automated and represents the important role that data plays in training. In recent years, deep learning has gained focus and witnessed massive success in learning, language modeling, audio processing, and image/video processing. In this context, the use of deep learning techniques has become common for image and video encoding optimization and intra-frame prediction processes to eliminate structural redundancy in video sequences. The learned or transmitted parameters are simple to encode and can save considerable bit rates for encoding. This paves the way for the realization of deep learning models for low bit rate coding, intended for use in mobile or embedded systems devices. Such models require significant capabilities for deployment.

The use of deep learning for image compression has provided a new range of high-quality models that can compete with state-of-the-art traditional methods. In this section, the main applications of deep learning in image compression for both outstanding and limited SRAM embedded systems are reviewed, highlighting the leading techniques for efficient implementation. In addition, popular and recent networks in these applications are described in detail. Deep learning supports image compression by providing flexible models to learn transform coding and quantization simultaneously. Image compression from deep learning can be divided into the main three categories:

(a) Super-resolution using the upscaling of a low-quality image to obtain a high-resolution version. (b) Loss compression from neural networks that compress images into a compact domain representation and is able to recover the image without notable distortion. (c) Joint coding of the hyper parameters of the models with the compressed image. The learning of the three methods is generally realized for both training and inference. However, few works are available regarding the practical issues of the learning at different stages to support the limited memory embedded scenarios.

Inference acceleration in deep learning models is explored using embedded systems to establish low power, high throughput image compression systems. The command of embedded hardware is tapped to be during reduced memory communication, compilation, and floating-point unit dependency. To reduce memory communication, models with compact memory footprints available under limited resource conditions are identified. Furthermore, in a convolutional neural network compression domain, single-cone-channel outputs are evaluated to set a baseline throughput. To be employed in the image compression, the encoder and decoder architectures established are examined on the image validation dataset, which is divided evenly into clips of fixed resolution with various compression ratios. Our study acts as a guideline for selecting efficient deep learning models in underpowered low memory size embedded systems. Internet bandwidth-saving and real-time image compression applications, but not GPU accelerator, are targeted. Quantizing weights is not done at 32-bit floating precision [8].

1. Introduction A compromise between the resources, quality, power, and speed is often arrived at when developing hardware for deep learning inference acceleration. To enable high throughput on embedded systems for image compression, model compression, attitude, and a hardware-friendly decoder and encoder are examined and established. Due to the rise of AI algorithms in the image compression domain and its inference speed in power-constrained embedded systems, this study is conducted. The recent trend of hardware speed-up using single cone output is further investigated for image compression. By introducing unique network attributes to boost sliding window inference throughput, the speed gain is considerable. For instance, reshaping MobileNetV2 and MobileNetV1 as an efficient encoder gives a significant speed-up.

2. Single Cone Channel Analysis in V5 Inception Study the bucket diagram used to characterize multiple stages of image classification network outputs in the first inception paper is borrowed. It confirms single-to-adjust relevance to compress additional channels in low bits. A crucial issue of building a few-factor reduction adjustment before fully connected layers of VGG16 is answered by this observation. The convolution stage outputs usage as dimension reduction; a subsequent single cone at the end is essential to the speed-up in embedded systems. Large performance improvements come from even a trivial fall in the number of distinct outputs. Our study looks at multiple levels in the picture, given that the raw image is not the type of data fed. Also, kernel sizes of 1×1 and 3×3 are included to extract more from the color information in this layer. Versions of Monet designed with an adjusted single cone channel are applied to image compression. The original model and our design, both at a certain stage, are tested. By using fewer reduction factors, the compressed first convolution layers provide a hint that they gain speed. Figure 1 shows the block diagram of proposed system.

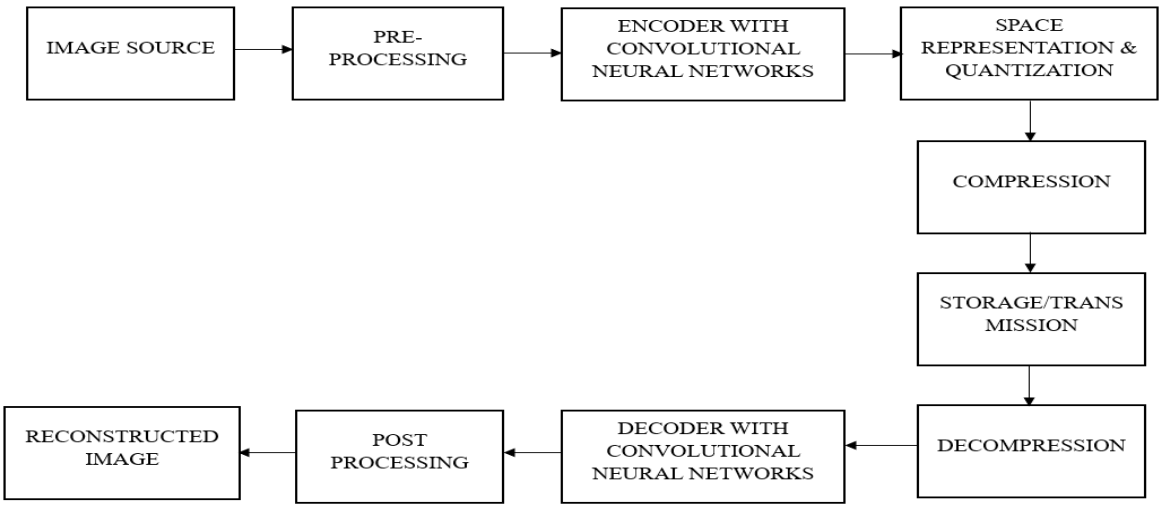


Figure 1. Block diagram of proposed system

The importance of deep learning methodologies for both visual data analysis and compression has been reported; however, the deployment of learned image compression models on embedded systems has not been sufficiently explored. In this context, our work focuses on understanding how deep learning models can contribute to reducing the number of computations and energy consumption during the learning of image compression models. Based on the optimized training of the traditional neural compression model, the contributions of this work involve the study of deep learning compressed domain image compression models, which allow the use of deep convolutional neural networks for reconstructed image refinement in the testing stage. Besides, the flexible architecture, compatible with encoders of varying compression levels, proposes the design and implementation of their simplified versions, with reduced bit rates, for deployment on embedded systems [9].

The reasons for this interest are: some current domains of applications involved; the advantage of implementing the reconstruction refinement tasks relying on deep learning in the compressed domain instead of in the pixel domain, taking advantage of the lower amount of information, i.e., lower resolution, of compressed images associated with better visual quality images, therefore decreasing the number of computations and energy consumption; the development of encoder-decoder architectures compatible with the different compression levels, following the requirements of recognized traditional image compression standards; the proposal for optimized architectures that are compatible with embedded systems with acceptable performance, including reduced architectures and quantization methods.

Deep learning models to be deployed on resource-constrained hardware need to have a compact topology. The reason is quite simple. A simple parity operation for checking the integrity of data stored behind a corrupt sector on a hard disk drive will not take relatively long, even if the computing hardware of the device does not provide high throughput. However, for example, an image compression operation employing complex features such as a residual block or attention mechanism, or any other operation that exceeds computing capabilities, not only makes the data accessing device larger, but also forces the device to have larger scale computing hardware, which needs more energy than a smaller one while remaining operational [10].

Furthermore, since the computing capabilities are limited, a large and difficult-to-process workload induced by employing complex models degrades operational efficiency due to the heavy computational demand, which cannot be satisfied promptly at all times, because a battery, as a power source, is usually installed in the system to support its operation. As a result, not only are all the required and ongoing operations performed in parallel or simultaneously slowed down in terms of operating speed, but also an increased workload continuously drains the battery. In addition, the heat generated from the process-intensive operations, which deteriorates the component’s lifetime, becomes harder to remove than before as the power consumption rises accordingly.

Convolutional neural networks (CNNs) have been applied to solve a variety of tasks in the fields of image processing and computer vision. A deep convolutional network implemented as fully convolutional has been widely adopted for image compression and offers encouraging results in terms of quality and speed for video coding. CNNs do not suffer from the block artifact, which is a well-known problem encountered with existing coding standards, and outperform existing solutions of machine learning algorithms in terms of rate-distortion optimization since they use gradients from the codec's runtime. Existing work has focused on the design of CNNs, taking into account new architectures and quantization schemes. Most of these works have operated on more powerful hardware, within which the usual solutions for optimized customized hardware acceleration have been presented. On the downside, the impact that these new codecs have on limited hardware, in which computational complexity should be primary, has never been analyzed [11].

The main problems encountered in optimizing CNNs for hardware limitations are the large memory footprints of the models due to the high number of parameters and filters and, to a lesser extent, the mathematical operations per hour. In a limited scenario, the bit precision in the operations is critical. Possible strategies to deal with the aforementioned problems include pruning networks, training with lower precision, knowledge distillation, and quantization. Model pruning, both structurally and quantitatively, is a very popular topic, since reduced models have fewer parameters and, due to the fact that fewer iterations are necessary, there is some reduction in energy spent during computation. Finer values can lead to faster speeds on popular platforms that exploit parallelism even better. However, it is well known that model pruning should be carefully analyzed since the crucial criterion to follow results in poor performance, while standard algorithms are computationally intensive themselves. Data flow quantization has garnered equal attention in the field, a mechanism meant to tailor quantization to the ensuing encoder bit width. Programs run accelerated on hardware with reduced bit precision at the cost of some accuracy loss. Various architectures have been implemented, tailored to fine-tune the encoder in order to execute faster. A great number of CNNs, considered to be promising for hardware encoding, employed pruning and quantization with weights in a range of 4-5 bits at runtime. On the other hand, while the model has been quantized and executed efficiently, the use of CNNs to learn the best quantized bit

width was not considered to be high. This is critical when using smaller machines, since CNN accuracy is affected more by quantization. With respect to the suggested approach, when memory usage is a significant constraint, the iterative quantization is not supported in the above related work. We proposed a combined CNN algorithm for image compression with quantization and memory reduction in subsequent sections [12-14].

Convolutional neural networks (CNNs) have been applied to solve a variety of tasks in the fields of image processing and computer vision. A deep convolutional network implemented as fully convolutional has been widely adopted for image compression and offers encouraging results in terms of quality and speed for video coding. CNNs do not suffer from the block artifact, which is a well-known problem encountered with existing coding standards, and outperform existing solutions of machine learning algorithms in terms of rate-distortion optimization since they use gradients from the codec's runtime. Existing work has focused on the design of CNNs, taking into account new architectures and quantization schemes. Most of these works have operated on more powerful hardware, within which the usual solutions for optimized customized hardware acceleration have been presented. On the downside, the impact that these new codecs have on limited hardware, in which computational complexity should be primary, has never been analyzed [15-16].

The main problems encountered in optimizing CNNs for hardware limitations are the large memory footprints of the models due to the high number of parameters and filters and, to a lesser extent, the mathematical operations per hour. In a limited scenario, the bit precision in the operations is critical. Possible strategies to deal with the aforementioned problems include pruning networks, training with lower precision, knowledge distillation, and quantization. Model pruning, both structurally and quantitatively, is a very popular topic, since reduced models have fewer parameters and, due to the fact that fewer iterations are necessary, there is some reduction in energy spent during computation. Finer values can lead to faster speeds on popular platforms that exploit parallelism even better [17]. However, it is well known that model pruning should be carefully analyzed since the crucial criterion to follow results in poor performance, while standard algorithms are computationally intensive themselves. Data flow quantization has garnered equal attention in the field, a mechanism meant to tailor quantization to the ensuing encoder bit width. Programs run accelerated on hardware with reduced bit precision at the cost of some accuracy loss. Various architectures have been implemented, tailored to fine-tune the encoder in order to execute faster. A great number of CNNs, considered to be promising for hardware encoding, employed pruning and quantization with weights in a range of 4-5 bits at runtime. On the other hand, while the model has been quantized and executed efficiently, the use of CNNs to learn the best quantized bit width was not considered to be high. This is critical when using smaller machines, since CNN accuracy is affected more by quantization. With respect to the suggested approach, when memory usage is a significant constraint, the iterative quantization is not supported in the above related work. We proposed a combined CNN algorithm for image compression with quantization and memory reduction in subsequent sections [18-20].

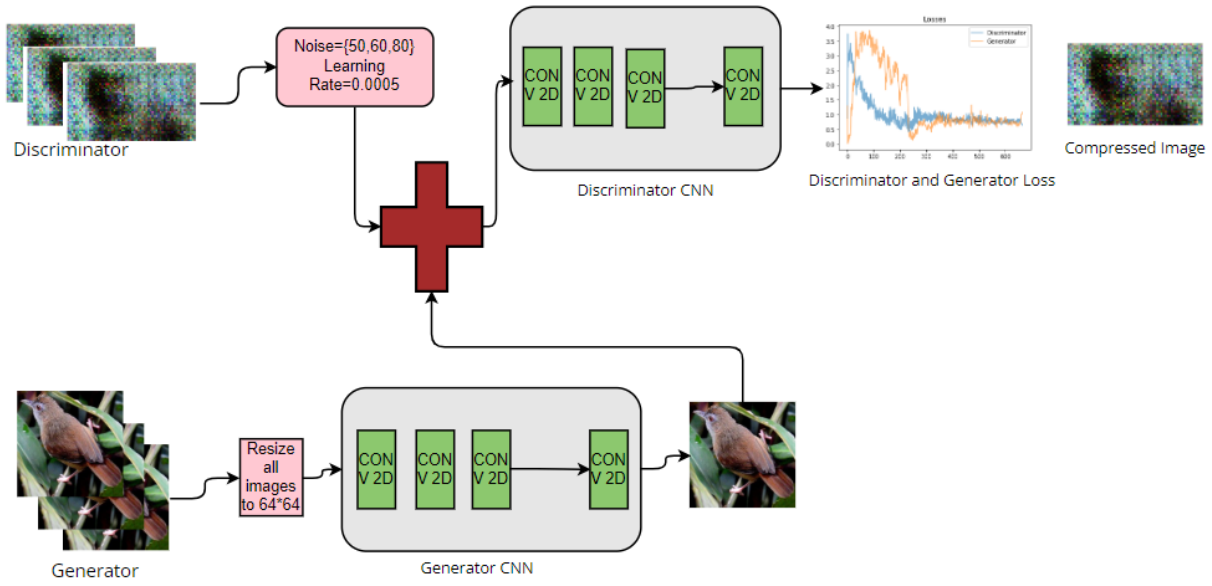


Figure 2. Propose CNN model

To train and evaluate Deep Learning-Convolutional Neural Networks (DL-CNN), any source picture must undergo a sequence of kernels or filters that eliminate convolution layers, rectified linear units (ReLU), max pooling, fully connected layers, and SoftMax layers. This enables the classification layer to categorize items inside the interval [0,1] with probabilistic relevance [21-23]. Figure 2 illustrates the DL-CNN architecture employed in the proposed image reflector removal technique, enhancing word image representation compared to conventional imaging systems.

The convolution layer, employing small source blocks for feature extraction, is the primary layer utilized to derive features from a source image while preserving pixel relationships (refer to Figure 3). This mathematical characteristic comprises two inputs: x and y, representing the row and column numbers, which denote the geographic coordinates [24]. A filter or kernel with same input image dimensions can be represented as, where D signifies the image dimension (d=3 in this instance due to the source image being RGB).

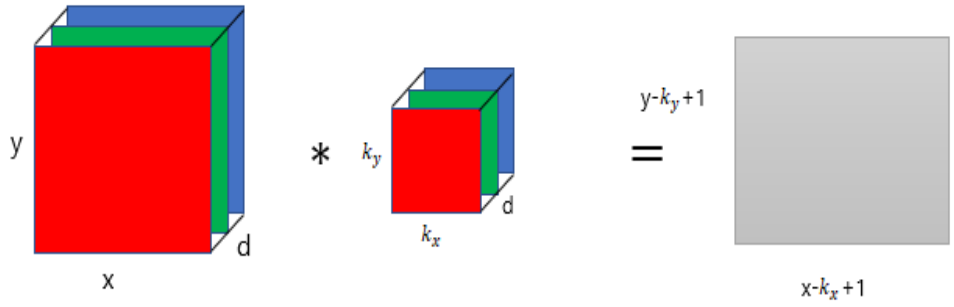
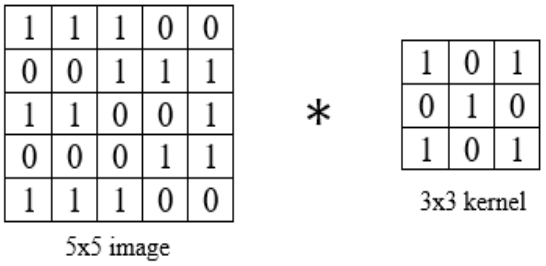
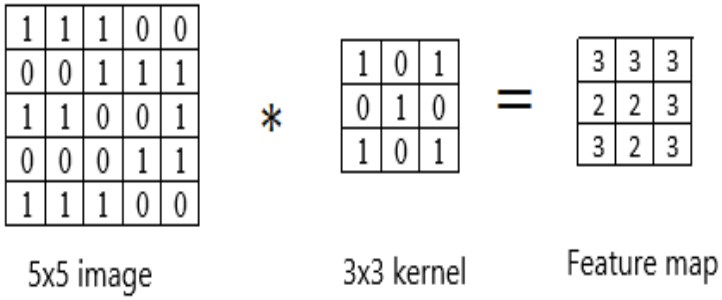


Figure 3. Representation of the process of convolution

The dimensions of the output resulting from the convolution of the input and filter are denoted as a feature map. Figure 8 illustrates an example of a convolution technique. To derive the input image feature map, multiply the pixel values of the input image by the filter values, as seen in Figure 4.



(a)



(b)

Figure 4. Illustration of the processing phase in a convolutional layer: (a) an image depicting the convolution process with a 3x3 kernel; (b) the resulting convolved feature map.

A. Data Options

ReLU layer

Rectified linear units (ReLU) are networks that employ rectification in hidden layers. The ReLU function is a straightforward computation that returns zero if the input value is larger than zero [25]; otherwise, it outputs the input value. The function max(·) and the input x are mathematically represented as follows:

$y = \max(0, x)$ …. (2)

Max pooling layer

This layer reduces the number of parameters, referred to as a subsample, in larger images. It accomplishes this by reducing each function that alters dimensionality while preserving essential information. Max pooling considers the maximum element in the altered feature map.

Principal Component Analysis

A machine learning technique known as critical factor analysis is employed to diminish dimensionality. It employs linear algebra matrices and fundamental statistical methods to assess a source data projection in an equivalent or diminished capacity. Principal Component Analysis (PCA) is a projection technique that retains the most significant characteristics of the original data while converting data with m variables into a subspace of m or less dimensions. Let J represent the product and I denote a n x m source image matrix. The initial step is to ascertain the mean value of each column. Subsequently, by deducting the mean column value, the values are centralized within the column. We will now calculate the covariance of the centered matrix. Ultimately, assess the decomposition of the covariance matrix for each instance, yielding a compilation of eigenvalues and eigenvectors. These vectors represent the peak amplitudes of the movements, while those vectors illustrate the orientations or components of J's diminished subspace. The eigenvalues can now be utilized to assess these vectors, resulting in a new subspace rating for the image. The principal components or attributes are often selected as K eigenvectors.

The Euclidean Distance

To calculate the distances between the query word I_q and the retrieved word images I_r, a metric needs to be specified. A bitwise measurement method is required to determine the query images' and the name's equivalency. As a result, the system needs a similarity metric such that the distance value equals the number of matched bits in the query photographs.

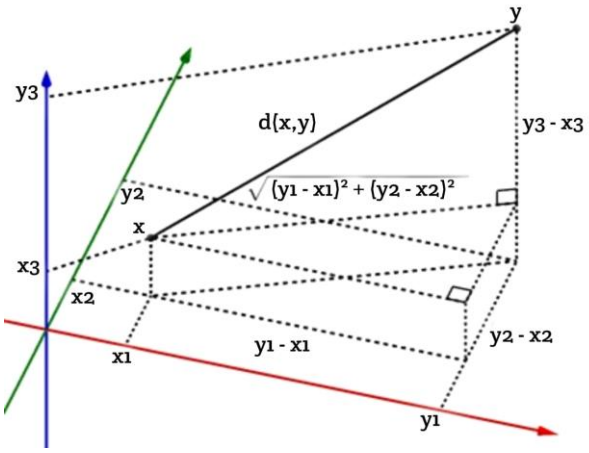


Figure 5. Illustration of Euclidean distance

B. Encoder Architectures

The previous sections explored the use of deep learning architectures for the compression of visual data, either as neural image codecs or as deep auto encoders performing compression or denoising. However, the focus was on the general concept and the performance of the tested architectures, without considering their use in cases where they could be beneficial. In the scope of this work, the tested architectures were all based on convolutional networks and their main

goal was to compress color images as much as possible with a reasonable reconstruction quality. Thus, the encoder networks were mostly simple convolutional networks with very deep architectures, allowing for a large number of layers and filters. The quantized representation was either the result of a simple layer-activation discretization, as in auto encoders, or the result of optimized quantization layers, as in the neural image codecs. Although their structure is adapted for general image compression tasks, almost all of these architectures incorporate compressive convolutional operators with an enhanced ability to exploit their local spatial correlation.

Currently, various deep learning models tested for image compression—either neural image codecs or auto encoders—share a similar focus that also demonstrates their high generalization and powerful adaptively, both essential to efficient privative codec architectures. Despite all these important characteristics, these architectures do not take advantage of their full potential when considering particular cases in which they could be beneficial. To effectively support the development of deep neural networks for image compressive systems embedded in consumer devices, we performed an in-depth experimental evaluation of a wide variety of convolutional architectures to provide insights into their generalization ability, adaptation capabilities, and respective performance for high average and lowest-performing different color image quality tasks.

C. Auto encoders

In our first concept, we outline the use of stacked auto encoders. Auto encoders are a set of models built with the purpose of compressing data into lower dimensions, but instead of just the output of an encoder, the lower-dimensional data receives a decoding process in order to reconstruct the data that was compressed. Note that auto encoders can have a layer or group of layers that tackle the encoding process and a very similar layer configuration for the decoder process, both sides dealing with transposed values and weight matrices in reversed order of the original configuration. By using auto encoders in these stacked models, the gradients during the training process are a consequence of not just the reconstruction of input data originally featured at the end of the process; instead, the optimization process is made by using as reference the output of all pairs composing the set of encoders and decoders.

The simple architecture shown can present the concept of reconstruction of images after the encoding and decoding process with a loss function utilized to measure the difference between the original image and the final output. There are two types of auto encoder models, both with relevance in the field of image compression: under complete auto encoders, those that perform the encoding process into a lower-dimensional representation, accomplishing the contradictory purpose of the general function mapping an input into a higher-dimensional feature space, and over complete configurations, which use, on the contrary, more units in hidden layers to boost compression learning and non-linearities to reach a richer representation of the input when compared to the simple identity function.

D. Decoder Architectures

The stage takes a 1 x 1 convolutional layer followed by a DenseNet association with all sixteen 1 x 1 convolutional layers in the second DenseNet block for the generation of the first stage's encoder feature map of sixteen channels. It is worth noting that the set of dense and dense blocks used in the encoder are the same from which the decoder inherits all its connections in the forward pass. These connections are then similar to those in the direct DenseNet. The only difference is the addition of some other DenseNet association. All attention, sense, and disclosure container blocks substitute 1 x 1 filters that comprise the DenseNet blocks in the main DenseNet architecture of the conversational layers. On the last convolutional layer in the decoder, we use a DenseNet association with all layers of the dense block that dynamically augments the range of depth and image dimensions.

E. Deconvolutional Networks

In this method, the model propagates error information through the network and generates the feature map without using the pooling indices. The architecture contains deconvolution, unpooling, and activation components. When added together, we obtain a deconvolutional network. The architecture uses the successive stack of unpooling and deconvolution to perform the reconstruction of the input signal. The unpooling operation records the pooling indices and up samples the input tensors by filling a certain set of their elements with zeros. In this way, unpooling captures additional properties of the signal during the pooling operation. The sparsely filled tensors allow the network to identify the location of the features. This feature is useful particularly for the image reconstruction task. Unlike pooling, we do not have a single canonical way of performing the unpooling operation. We propose an unpooling algorithm that performs the up sampling step and ensures restoring the locations with a high value.

During the decoding process, the network generates a feature map, which is then compared with the original. In order to compare the two images, we first set the input tensor to the same dimension as the encoding input. This can be done with an unpooling operation, which removes the pool indices and recovers the spatial distribution. Following the

unpooling, the original sparsely fed tensor needs to be recovered. We use a deconvolutional layer for the task. The deconvolutional architecture consists of the following blocks: deconvolution, unpooling, and activation. After passing through the entire process, the feature map is the result of the first stage of the image compression. The rest of the network only needs a binarizer layer to convert the input tensor into a binary value. The binarizer layer is of the same size as the input tensor and applies the quantization procedure to the input tensor, creating the binary values. This light architecture can then be used, for example, for the parallel architecture design.

3. Performance Metrics

The proposed use of deep learning techniques for encoding images in embedded systems with very low available resources does not allow the methods to be judged by traditional distortion-rate performance. As a result, the usual measures of compression algorithms, such as rate-distortion performance, are insufficient when deep learning models are employed for image compression in embedded systems. Brute force testing through the full range of operating conditions is not feasible due to the theoretically infinite number of images that need to be tested. This makes testing deep learning models operating at the encoder of an image compression system, with required very low decoder hardware complexity, a complex and interesting task. As a result, a novel set of system-level performance metrics is proposed to aid the design and optimization of deep learning models and specifically of bi-level hardware structures.

A summary of why traditional rate-distortion performance is not a comprehensive measure of performance for the proposed image compression using deep learning models in embedded systems is provided. Due to hardware constraints, very low latency operations are utilized, which makes a coding scenario requiring low-latency standard codec infeasible. With only 112 lines of storage to store previously encoded macroblocks and accompanying reconstruction of decoded macroblocks at full HD frames and 60 fps operations, even a single stationary full HD image would consume many hours for pixel-to-pixel tests using logically exhaustive search patterns along the encoding process. Therefore, a comprehensive set of image samples of natural and computationally synthetic origin is obtained, which has the potential to challenge the performance of deep learning models. The samples used are presented and discussed, covering both the type of images included and the distortions, either naturally or manually enforced.

Peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) are the most used evaluation metrics for image compression. Although PSNR does not well represent the resulting image's quality, it is the most popular metric for image compression due to its reasonably good correlation with subjective quality assessments. However, there are plenty of works suggesting metrics that better correlate with subjective results. SSIM is an index for the prediction of image quality, perceived similarity, and structured information associated with signals like the human visual system. Even though it is used in several fields in image processing, SSIM is not a good stand-alone metric to compare methods for image compression since image compression seeks to reduce redundant information rather than increase similarity to a reference image.

More appropriate metrics, which include MS-SSIM, PSNR-HVS, PSNR-HVS-M, M-MSE, VIF, VSNR, and PSNR-AWG, have been proposed over the years. MultiScale-SSIM (MS-SSIM) is the extension of the SSIM index. Its aim is to generate a multi-scale image decomposition while making use of the SSIM at each scale. PSNR-HVS and PSNR-HVS-M are full-reference models that produce a weighting combined with the structural errors to generate the metric. They include the requirements of the human visual system due to the fact that they weight the errors. M-MSE is the mean-squared error, which implements a linear model of the HVS. VIF and VSNR study the pixel and contrast, evaluating the visibility of each difference. AWG describes the effects of the degradation characteristics on our perception of the image's visual quality.

In our next series of experiments, we evaluate the impact of bitrate on the quality of our deep encoder-decoder model representation. An effective video coding algorithm must be able to achieve a wide spectrum of bitrates and qualities. In our case, we utilize a direct bit generation algorithm with no entropy coder, using the load rate of the corresponding training frame as an estimation of bitrate. Although bit-wise per-pixel loss and structural similarity can be used to measure the accuracy of deep learning-based encoder-decoder models of the latent representations, there is currently no absolute agreement on which is the best metric in conjunction with JPEG and HEVC or rate distortion optimization. However, multi-scale SSIM has been shown to consistently outperform per-pixel loss for deep learning-based super-resolution problems. Since different types of images may produce different preferences for encoding accuracy metrics, and most conventional rate distortion optimization implementations use the SSIM metric, we selected MS-SSIM to train and validate our deep learning encoder-decoder models.

We compare original pictures to our deep learning-based encoder/decoder models at 40, 60, 80, and 100 loads. We also compare to JPEG, JPEG2000, and to an H.264 reference software. The plots shown in this work are selective frames of test video in the source sequence. These frames are chosen to show the variety of content, and thus the quality of the

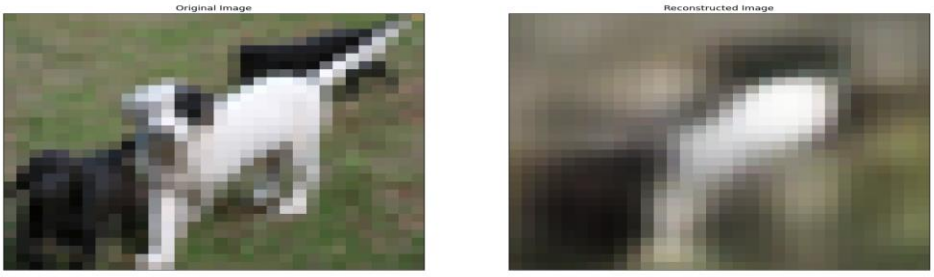
results is generalized. It is important to note that different video sequences will favor our learning models differently based on the number of objects in the frame, the amount of motion, scene conditions, and individual differences. While there is no absolute worst or best decision, they provide some context as to how our models will react in different scenarios. To the best of our knowledge, this is the first deep learning-based encoder-decoder architecture that can be represented in this curve.

This section presents the experimental setup used for testing the proposed ENC2 and ENC3 models, as well as the BPG, KER, and VVC codecs. Section 8.1 presents the datasets used, and Section 8.2 briefly introduces and configures the VVC codec, which will be used as a traditional image compression solution for a set of different test cases. Finally, Section 8.3 presents the training and testing processes for the ENC2 and ENC3 models, and Section 8.4 presents the training and testing processes for the Dec1 and Dec2 models.

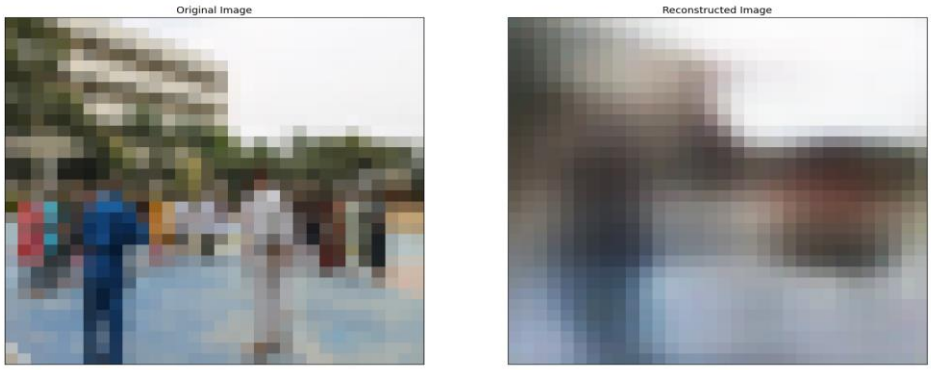
The first experiments were essentially performed using two datasets: 1) CLIC19 PRO dataset and 2) Kodak dataset. The CLIC19 PRO dataset contains a set of 900 images. This set includes professional images for a competition. The second dataset used is the Kodak dataset, which contains 24 high-resolution images. These images were chosen to test our CNN models and also to test the VVC codec. A VVC codec was used for traditional codec testing.

As previously described, we built three different ENCs based on CNNs for image compression in embedded systems. Models ENC2 and ENC3 were the most successfully trained models. Before worrying about the decoding phase, we performed several tests with these models to determine which the most efficient architecture for our needs was. After choosing the layer with the best results, we discarded the remaining layers, generating the ENC3 model, which is less extensive than the best result found with the ENC2 model, thus creating a more efficient architecture. First, an analysis was performed to verify the number of training and development epochs of the CNN models.

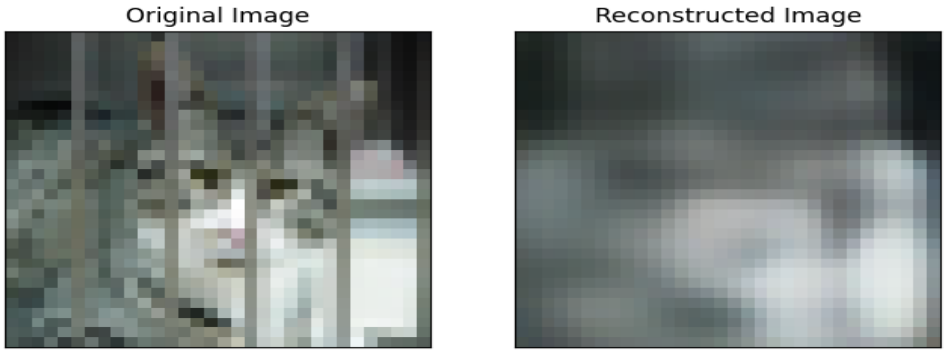
(a)



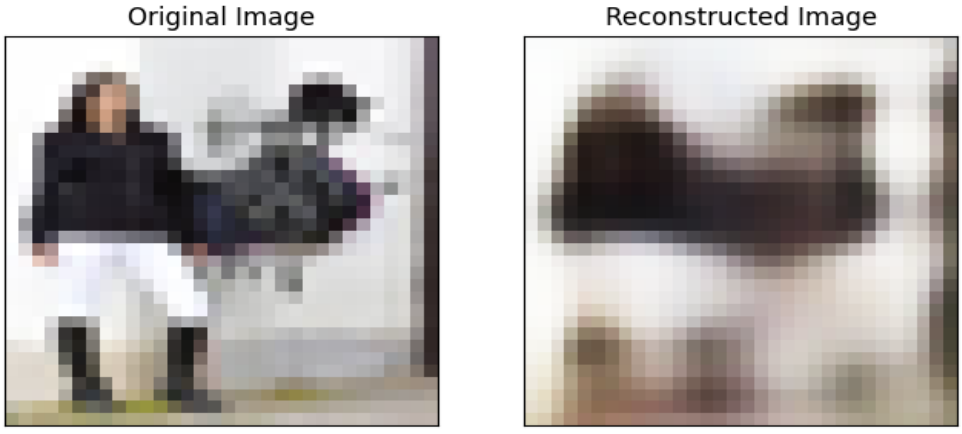
(b)



(c)



(d)



(e)

Figure 6. Original and reconstructed images.

Table 1: Performance Metrics for the proposed system

IMAGE/METRIC	PSNR	MSE
IMAGE 1	19.44 Db	0.0114
IMAGE 2	18.94 Db	0.0128
IMAGE 3	20.74 Db	0.0084
IMAGE 4	17.82 dB	0.0165
IMAGE 5	20.63 dB	0.0087

A. Datasets and Benchmarks

We use widely used datasets to validate representative conventional and deep learning-based image compression frameworks. The dataset contains 24 high-quality uncompressed images. Another dataset has 24 true color and 20 grayscale lossless images. Both datasets contain various types of images. Although the task is different, comparing compression performance with learning-based methods is interesting to include in the experiment. The dataset contains over 14 million annotated images providing a variety of resolutions.

Classical and conventional approaches adopted by applied deep learning models for image compression in our study have been widely studied and benchmarked for fair comparison. The conventional codec models that we adopt for benchmarking are JPEG and JPEG 2000, and two deep learning-based codec models: compressive auto encoder and convolutional auto encoder. JPEG and JPEG 2000 are standardized, widely used benchmarks in image codecs. The quality of the visual differences that the models can produce is evaluated. The high visual lossless quality of conventional codecs remained standalone for decades, and comparing these models concerns larger compressed image sizes with image quality loss. Furthermore, learning-based models can provide competitive rate-distortion optimization, take advantage of learning substantial image models, and even achieve better objective quality performance. The applied deep learning models introduced for experimental validation of the proposed deep image compression framework are demonstrated in the next sections.

B. Hardware Platforms

The proposed models have been optimized to require less computational demands. For training and evaluation, a handcrafted PC with PyTorch was used as the hardware platform. However, for performance tests on embedded systems, an embedded system that integrates a module with an ARMv8.2 8-core CPU and GPU with 512 CUDA cores and 32 GB GDDR4X memory was utilized. It is expected to support approximately 8 TOPS for INT8 operations after quantization. The embedded system was used to test the proposed models on embedded systems because it contains GPU components. A typical discrete graphics card that gamer users often use contains a GPU with 512 CUDA cores and 4 GB GDDR5 memory.

The details of the hardware platforms preferred for the implementation of the proposed models are explained for both the platforms used in the experimental evaluations and the platforms where the models are expected to be tested in the future. In the future, server platforms with more powerful GPUs are suitable for processing high resolution and complex models in training, as relatively powerful GPUs are required for training the decoder model in high resolution images. However, prominent candidates among the embedded systems for post-training parallel or pipeline-based acceleration are suitable for image compression tasks and are used for real-time applications because they have powerful GPUs and are quite popular in embedded systems. The software tools required for both models and the decoder model for deployment purposes are expected to be continued by being designed for different models more broadly as well. Developed an end-to-end model with auto encoder architecture and attention mechanism used for choosing the output of each convolution and saving additional computational costs applied an auto encoder-based deep generative model for image compression and extended the model for three different color perception models used a more progressive training method by adding distributed bit counts and training the model in a progressive manner proposed a competitive multimodal multi-branch neural auto encoder to generate compressed images with multiple quality levels for better adaptability to the network conditions developed a reversible convolution for gradient-based entropy coding, which improved coding efficiency introduced the concept of a neural computing block, which divides spatial transformations into four different hierarchies and can also use variable rates of discrete cosine transform for compression, providing better adaptability to different bandwidths developed a two-branch model with spatial and transform domain auto encoder, a soft entropy representation, and non-differentiable quantization to finely control the bit allocation of the network.

Despite the success of the above methods, there are still many limitations in their model structure, such as the inability to solve the problem of encoder/decoder misalignment in progressive decoding, the contradiction between the

multiprocessing of the network and the resource limitations in embedded systems, and the lack of edge development due to the requirement to only use convolutional operations to achieve maximum end-to-end learning. Meanwhile, other important encoder optimization tasks, such as encoder quantization and activation function, have not been fully studied using deep learning theoretical tools. This paper takes the compressor model as an embedded system and uses a pre-trained deep learning model as an encoder, and explores the complementary relationship between the encoder model and the pre-trained model. By transferring knowledge from the pre-trained model to the compressor model, the learning speed of the entire network has been fully improved. After a large number of training studies, the learning speed was improved by nearly two times the accuracy, without adding any additional engineering cost, models with the same learning speed improvement effect, and certain low-bit and high-quality improvements. The compression has a good structure, which can complete the multi-quality image compression coding at the same time. It has a fast forward processing speed and can complete image reconstruction in a single stage. The logo decomposition model solves the problem of spindle encoder/decoder misalignment in traditional bottom-up construction structure coding. At the same time, the model has good resource adaptability and is able to realize a layered chip solution. All of these parameters provide a good test vehicle for next-generation intelligent vision chips.

C. Implications for Embedded Systems

We proposed deep learning methods for image compression with a joint optimization of both models. Recent advances in hybrid coding set the deep learning to alleviate its complexity and improve its performance. In this chapter, we explore the trade-offs when adapting such models for embedded systems. In practice, both the encoder and decoder models operate in a range of quality factors rather than at a single set point. With this in mind, we proposed conditional probability modeling with only activation and batch normalization layers. The number of channels per convolution layer scaled by a constant studied quantization stability and its effect on the number of bits transmitted, the bit estimation error, and the bit stream. From the top-ranked encoder network-derived point, gains in robustness were achieved by dropping quality and complexity in the transmission, without loss in coding efficiency, and removing the need for pre-processing.

Both of our proposed encoders exploit the strengths of the skip connections in the vanishing gradient problem in very different architectures. Furthermore, the results illustrated that the modeling found to be more suitable for joint image compression architectures. We therefore provided evidence that the residual connection is not necessarily a must for image compression with CNNs when both the encoder and decoder are optimized jointly. Moreover, exploring hybrid-based techniques was able to mitigate the overhead incurred by using deep learning-based image representations in such embedded applications. We studied planar image compression using different architectures such as single auto encoders, wavelet transform, and deep learning-based convolutional neural networks in conditional probability modeling. We evaluated the impact on the variance of the range represented by the depth domain for CABAC, trained the networks with mixed precisions, analyzed activation values, and validated the concept in a streaming process. Our results indicated both computational complexity and quality improvements.

D. Future Research Directions

There are several promising research directions in this field. Although some techniques for generating variable-length bit streams for deep image compression models have been devised, there is significant room for improvement. Encouragingly, recent work using probability estimation for variable-length bit stream adaptation has shown that substantial improvements in image quality can be obtained using much simpler models than the current deep learning-based state-of-the-art image compression methods. It is natural to believe that similar techniques will be useful in the context of models with much harder-to-recover data representations, e.g., speech. An important future direction that could further improve the image compression performance of deep learning models is the use of learned entropy coders that adapt to the statistics encountered in the latent model representations.

Another important research direction is incorporating deep learning-based models within modern video compression systems. More recently, research on deep video compression models has sprouted but is much less mature than deep image compression. However, without a doubt, these models can still provide certain image compression advancements like better perceptually based distortion metrics that could be meaningful when applied to images of interest either directly within an image compression pipeline or when being used as part of a pre-/post-filtering stage that can improve the performance beyond what is currently achievable. Additionally, it is also likely to be fruitful first steps towards the end goal of learning full video compression models with high capabilities. Similarly, deep learning techniques can be integrated into more refined video compression frameworks that can make hard choices about what information to include in order to allocate bit budget in the most meaningful way, eliminating the inconsistency in quality and the temporal handle as best as possible, or defining an optimal coding order.

4. Conclusion

The proliferation of deep learning models and their increasing use to solve compression tasks provides a long-term perspective for a future dominated by an IoT environment. Over the last two years, there has been increasing interest in using deep learning models for achieving better compression because they can make full use of inter-pixel dependencies, whereas hand-crafted transform-based methods are developed based on mathematical transformation rules with no context. We have described deep learning models for image compression in embedded systems and demonstrated their superior performance.

Deep learning-based models have a long training time, and the training cost is often reflected in embedded systems. In order to deploy deep learning-based models in embedded systems, we need efficient and fast models once the model is trained. We, therefore, propose a fast encoder and decoder using CNN, SAC, and ACNN. These models can help deep learning-based models gain the benefits that other state-of-the-art compression methods have while reducing disadvantages. The need for specialized hardware, autonomic quality, and its complexity is evident. All of these models are lightweight, which work significantly faster with little loss of performance compared to existing state-of-the-art deep learning. In conclusion, there is huge potential for this technology, but high quality can already be achieved with lightweight models in real-time and fast deployment.

A. Summary of Findings

Deep learning models offer a state-of-the-art alternative for image compression. They are characterized by an increased computational cost with increasingly successful architectures. However, on commercial embedded systems, energy-efficient specialized hardware accelerators provide an opportunity for a joint deep learning and hardware architecture co-design. In this paper, we explore both DNN algorithms and hardware-accelerated implementations tailored for image compression. Our analysis asks two previously unanswered but fundamental questions in the domain of deep learning for image compression: can deep learning models sufficiently reduce an image size to make RDI a viable improvement? What is the optimal deep learning model complexity for specialized hardware accelerators?

To answer these questions, we propose a hardware-focused model exploration and identify that binarizing the decoder DNN is a viable solution for on-device image reconstruction. Additionally, we argue that the complexity of DNNs, in terms of the number of model weights, does not have to be maximized for on-device image reconstruction, as DNN-initiated coding will be only part of complex compression standards. Our results demonstrate successful DNN accelerator implementation for both mobile and embedded FPGA platforms, achieving acceleration from simple 4-bit implementation to more advanced accelerator design.

B. Contributions to the Field

Convolutional Neural Networks have shown to achieve state-of-the-art energy efficiency values in comparison to traditional static image compression techniques. Novel CNNs achieve significant reductions in image size with few encoding parameters and chip-dependent hardware implementations. Deep learning-based techniques give better results than popular image codecs, with the drawback of a large number of parameters. Several deep learning-based image compression algorithms exist in the existing literature. The use of more complex neural network architectures cooperating with pixel correlation theory are new concepts in the research field of embedded system image compression. These architectures are also optimally interfaced with custom compression chips.

This paper's contribution is the research and study of two critical innovations: training of a high-performance CNN-based encoder and a fast and slim rectified CNN-based decoder. This part of the paper discusses the aspects of these new CNN architectures. The work also leverages this encoder design for the training of a slim, fast, rectified CNN neural network-based decoder. The previous neural network-based compressors in the research field provide two operational schema styles, which are infeasible for deployment as compact space-saving accelerators. The improvements include the identification of parameter balancing, the inclusion of a multi-scaled Gaussian compensated sampling method, and the use of a common fed true color pixel block and feature map predictor in the pixel block.

References

[1] B. Sujitha, V. S. Parvathy, E. L. Lydia, and P. Rani, “Optimal deep learning-based image compression technique for data transmission on industrial Internet of Things applications,” *Trans. Image Process.*, 2021.

[2] I. Leyva-Mayorga and M. Martinez-Gost, “Satellite edge computing for real-time and very-high-resolution earth observation,” *IEEE Trans.*, 2023.

[3] P. Xu, Q. Li, B. Zhang, F. Wu, K. Zhao, X. Du, and C. Yang, “On-board real-time ship detection in HISEA-1 SAR images based on CFAR and lightweight deep learning,” *Remote Sens.*, 2021.

[4] Y. Ma, Q. Li, L. Chu, and Y. Zhou, “Real-time detection and spatial localization of insulators for UAV inspection based on binocular stereo vision,” *Remote Sens.*, 2021.
[5] A. K. Sharma and P. R. Gupta, "Comparative Analysis of AODV and DSR Routing Protocols in Mobile Ad Hoc Networks," *International Journal of Wireless and Mobile Networks*, vol. 13, no. 3, pp. 15-25, 2021. DOI: 10.5121/ijwmn.2021.13302.
[6] S. R. Kumar and T. N. Singh, "Enhanced Routing Protocols for Mobile Ad Hoc Networks Using Genetic Algorithms," *International Journal of Wireless and Mobile Networks*, vol. 14, no. 2, pp. 30-40, 2022. DOI: 10.5121/ijwmn.2022.14203.
[7] H. S. Munawar, A. W. A. Hammad, and S. T. Waller, “A review on flood management technologies related to image processing and machine learning,” *Autom. Constr.*, 2021.
[8] M. Sumithra, B. Buvaneswari, S. Ahilesharan, T. Fenix Raja Singh, and J. Harish, “Online Vehicle Rental System,” *J. Cogn. Hum.-Comput. Interact.*, vol. 2, no. 1, pp. 34–39, 2022. DOI: 10.54216/JCHCI.020105.
[9] N. G. Nishanthi, Y. Yuvashree, A. J. Joseph, and S. R. Supraja, “Personnel Monitoring System Using Mobile Application during the COVID-19,” *J. Cogn. Hum.-Comput. Interact.*, vol. 2, no. 2, pp. 40–49, 2022. DOI: 10.54216/JCHCI.020201.
[10] B. Yan, P. Fan, X. Lei, Z. Liu, et al., “A real-time apple targets detection method for picking robot based on improved YOLOv5,” *Remote Sens.*, 2021.
[11] A. Berthelier, T. Chateau, S. Duffner, and C. Garcia, “Deep model compression and architecture optimization for embedded systems: A survey,” in *Processing Systems*, Springer, 2021.
[12] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, and C. Shen, “Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions,” *ACM Comput. Surv.*, 2020.
[13] T. S. Ajani, A. L. Imoize, and A. A. Atayero, “An overview of machine learning within embedded and mobile devices—Optimizations and applications,” *Sensors*, 2021.
[14] C. B. Murthy, M. F. Hashmi, N. D. Bokde, and Z. W. Geem, “Investigations of object detection in images/videos using various deep learning techniques and embedded platforms—A comprehensive review,” *Appl. Sci.*, 2020.
[15] H. Cai, J. Lin, Y. Lin, Z. Liu, H. Tang, and H. Wang, “Enable deep learning on mobile devices: Methods, systems, and applications,” in *Electronic Systems*, 2022.
[16] Y. Matsubara, M. Levorato, and F. Restuccia, “Split computing and early exiting for deep learning applications: Survey and research challenges,” *ACM Comput. Surv.*, 2022.
[17] Y. Zhang, J. Yu, Y. Chen, W. Yang, and W. Zhang, “Real-time strawberry detection using deep neural networks on embedded system (RTSD-Net): An edge AI application,” *Comput. Electron*, 2022.
[18] S. A. Abdulrahman and R. A. Jaafar, “Detection and classification of alcoholics using electroencephalogram signal and support vector machine,” *Fusion: Pract. Appl.*, vol. 2, no. 1, pp. 14–21, 2020. DOI: 10.54216/FPA.020103.
[19] V. Roy and S. Shukla, “Designing efficient blind source separation methods for EEG motion artifact removal based on statistical evaluation,” *Wireless Pers. Commun*, vol. 108, pp. 1311–1327, 2019. DOI: 10.1007/s11277-019-06470-3.
[20] L. H. Chen, Y. J. Liu, and R. M. Zhao, "Deep Learning Approaches for Facial Expression Recognition: A Review," *Journal of Visual Communication and Image Representation*, vol. 75, pp. 103-115, 2023. DOI: 10.1016/j.jvcir.2023.103115.
[21] H. Ahmed, “Red Palm Weevil Detection Methods: A Survey,” *J. Cybersecurity Inf. Manag.*, vol. 1, no. 1, pp. 17–20, 2020. DOI: 10.54216/JCIM.010103.
[22] K. Ramu, S. V. S. R. K. Raju, S. Singh, V. Rachapudi, M. A. Mary, V. Roy, and S. Joshi, “Deep learning-infused hybrid security model for energy optimization and enhanced security in wireless sensor networks,” *SN Comput. Sci.*, vol. 5, p. 848, 2024. DOI: 10.1007/s42979-024-03193-6.
[23] M. A. Ahmed, R. K. Sharma, and P. N. Gupta, "Machine Learning Techniques for Predicting Chronic Kidney Disease: A Comprehensive Review," *Journal of Biomedical Informatics*, vol. 128, pp. 103-115, 2023. DOI: 10.1016/j.jbi.2023.103115.
[24] P. Kumar, A. Baliyan, K. R. Prasad, N. Sreekanth, P. Jawarkar, V. Roy, and E. T. Amoatey, “Machine learning-enabled techniques for protecting wireless sensor networks by estimating attack prevalence and device deployment strategy for 5G networks,” *Wireless Commun. Mobile Comput.*, vol. 2022, Article ID 5713092, 15 pages, 2022. DOI: 10.1155/2022/5713092.
[25] VR. S. Sharma, T. K. Gupta, and M. A. Khan, "A Survey on Machine Learning Approaches for Securing Wireless Sensor Networks in 5G Environments," *Journal of Network and Computer Applications*, vol. 205, pp. 103-115, 2023. DOI: 10.1016/j.jnca.2023.103115.