POLYTECHNIC OF TURIN

Master's Degree in Computer Engineering Data Science

Master's Degree Thesis

Image Compression using Deep Neural Networks

Applying Deep Neural Networks Compression Methods on SIREN Architecture

Supervisors

Prof. Enrico MAGLI

Prof. Diego VALSESIA

Candidate

s253666 Francesco Maria CHIARLO

July 2021

Summary

The main focuse of the current thesis proposal is related to performing image compression via Implicit Neural Representations based deep neural network architectures, specifically employing SIREN architectures, adopting pruning and quantization techniques for constraining in the number of weights parameters and their numerical representation, respectively, while measuring and evaluating induced performance image quality metrics to be related to achieved image compression bit rate calculated from deep neural network overall footprint. The results obtained while running and collecting data related to the involved image quality metrices, such as Pnsr score and bit rate, have suggested us as well as provided us evidence of how critical and still heavy task represents the attempt of reaching image compression throughout common Neural Network Compression Techniques applied directly to Neural Network Models. Finally, due to the vast number of potential suitable hyper-parameter configurations, we have noticed that there are chance to reduce the gap we can measure, in terms of performance, between well-established image compression methods such as Jpeg and SIREN compressed models.

Acknowledgements

ACKNOWLEDGMENTS My Parents, my sister and my brother for having supported me, throughout these years of study at Polytechnic of Turin.

"Any A.I. smart enough to pass a Turin test is smart enough to know to fail it."
Ian McDonald

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the following work thesis we are going to investigate the potential behaviors that *SIREN* based instance models may attain, once they were already trained at list to reach over trained stage and constitute a suitable baseline architectures to be employed for image compression, improving reachable image compression degree further employing some kind of deep learning compression method.

For what concerns with our targets beared in mind and related to thesis most important goals, and since we decided to adopt *SIREN* based deep neural network scheme as the model's architecture, the main insights and information we take care of, referring to the research work within which *SIREN* like models have been present, we can state what follows.
As we can read from Stanford University supported research about *SIREN* Architecture paper's introduction [40], we know that implicitly defined, continuous, differentiable signal representations parameterized by neural networks, have emerged as a powerful paradigm, offering many possible benefits over conventional representations. In particular, they propose:

- to leverage periodic activation functions for implicit neural representations and demonstrate that these networks, dubbed sinusoidal representation networks or *SIREN*s, are ideally suited for representing complex natural signals and their derivatives.

- as well as, also analyze *SIREN* activation statistics to propose a principled initialization scheme and demonstrate the representation of images, wavefields, video, sound, and their derivatives.

However, as we can state, even from the very beginning, our work thesis instead will be focused only and principally on computer vision related tasks, in particular a computer vision problem which is not a common classification problem, but rather a problem within regression family, where we want measure how much well the *SIREN* architecture is able to represent and estimate pixel values based on input pixels coordinates, so that, our model's parameters along with picked and fixed hyper-params as well, will become a good approximation of an implicit representation of the input processed image we attempt to compress.

Furthermore, in the paper, researchers, since from the very beginning show what they are looking and searching for, in math terms, and starting by the mathematical formulation, the main equation that have lead their entire publication work, that is the so called *Implicit Neural Representation*. In fact they reported that are interested in a class of functions Φ such that satisfy equations of the form:

$$F(x, \Phi, \nabla_x \Phi, \nabla_x^2 \Phi, \dots) = 0, \Phi : x \to \Phi(x) \tag{1.1}$$

The just above transcribed equation and function definition, in the precise case of Φ function, should be interpreted as follows. This implicit problem formulation takes as input the spatial or spatio-temporal coordinates $x \in R^m$ and, optionally, derivatives of Φ with respect to these coordinates. So that, the goal of Stanford's researchers has been to learn a neural network that parameterizes Φ to map x to some quantity of interest while satisfying the constraint presented in Equation 1.1. In this manner, they say that Φ s implicitly defined by the relation defined by F and they refer to neural networks that parameterize such implicitly defined functions as *implicit neural representations*.

Therefore, as they show in their paper, a surprisingly wide variety of problems across scientific fields fall into this form, such as modeling many different types of discrete signals in image, video, and audio processing using a continuous and differentiable representation, learning 3D shape representations via signed distance functions [1, 34], and, more generally, solving boundary value problems, such as the Poisson,Helmholtz, or wave equations.

They also stated and carried out most of their reasoning around the fact that a continuous parameterization offers several benefits over alternatives, such as discrete *grid-based representations*. So, they report as an example that:

- due to the fact that Φ is defined on the continuous domain of x, it can be significantly more memory efficient than a discrete representation, allowing it to model fine detail that is not limited by the grid resolution but by the capacity of the underlying network architecture;
- furthermore, Being differentiable implies that gradients and higher-order derivatives can be computed analytically, for example using automatic differentiation, which again makes these models independent of *conventional grid resolutions*;
- finally, with well-behaved derivatives, implicit neural representations may offer anew toolbox for solving inverse problems, such as differential equations.

For these reasons, within their works and in particular inside section 2 they states that implicit neural representations have seen significant research interest over the last year. And citing what they mean they observe however that most of these recent representations build on ReLU-based multilayer perceptrons(MLPs). While promising, these architectures lack the capacity to represent fine details in the underlying signals, and they typically do not represent the derivatives of a target signal well. Thisis partly due to the fact that ReLU networks are piece-wise linear, their second derivative is zero everywhere, and they are thus incapable

of modeling information contained in higher-order derivatives of natural signals. While alternative activations, such as tanh or softplus, are capable of representing higher-order derivatives, we demonstrate that their derivatives are often not well behaved and also fail to represent fine details.

Lastly. they ended their introduction on *SIREN* based neural network models purposing how to address the several limitations they detected and we reported just few line above. In fact, they leverage MLPs with periodic activation functions for implicit neural representations. They have been able to demonstrate that this approach is not only capable of representing details in the signals better than ReLU-MLPs, or positional encoding strategies proposed in concurrent work [31],but that these properties also uniquely apply to the derivatives, which is critical for many applications, but that these properties also uniquely apply to the derivatives, which is critical for many application.

To conclude, the contributions of their seminal work include:

- A continuous implicit neural representation using periodic activation functions that fits complicated signals, such as natural images and 3D shapes, and their derivatives robustly.
- An initialization scheme for training these representations and validation that distributions of these representations can be learned using hypernetworks.
- Demonstration of applications in: image, video, and audio representation; 3D shape re-construction; solving first-order differential equations that aim at estimating a signal by supervising only with its gradients; and solving second-order differential equations

However, in our thesis work instead we focus mainly on computer vision related problems, applying *SIREN* based neural network architecture properly configured, choosing suitable hyper-parameters values for both number of hidden layers and features per layers respectively, so that we are able to train base models that are going to constitute our baselines in order to compare how much close compressed counterpart derived models can be employed as an appropriate implicit neural representation of input processed image to represented by network's parameters. We do not just halt comparing baseline models performance against compressed counterparts, but also adopt as comparing baseline performance we can gather when employing Jpeg algorithm for compressing and obtaining an artifact which aims at representing as close as possible the original input image compressed by Jpeg procedure itself.

Chapter 2

Related Works

We briefly report some seminal and interesting already published research studies and works both related to the problem and context of *periodic nonlinearities* as well as other research issues and reports which instead are referring to compression methods within the artificial intelligence area of investigation related to approaches and algorithm derived for reducing in terms of memory footpring already trained and finte-tuned models while maintaining as much as possible unaltered performance properties or scores.

2.1 Implicit Neural Representation and Periodic Nonlinearities based Solutions

As we can read from citations and references Stanford Research group, which lead the study of Siren based models and their use, have reported about previous already done works over Implicit Neural Representation area, within their related work section, we understand that only more recent work has demonstrated the potential of fully connected networks as continuous, memory-efficient implicit representations for shape parts [11, 12], objects [1, 15, 30, 34], or scenes [5, 39]. These representations are typically trained from some form of 3D data as either signed distance functions [1, 30, 34, 36] or occupancy networks [6, 29]. In addition to representing shape, some of these models have been extended to also encode object appearance [3,5,10,15,16],which can be trained using (multiview) 2D image data using neural rendering [17]. Temporally aware extensions [18] and variants that add part-level semantic segmentation [19] have also been proposed.

2.2 Deep Neural Networks Compressing Techniques

As we can already read from seminal works done by *Michael H. Zhu et a.l* whit in their research published paper known as *"To prune, or not to prune: exploring the efficacy of pruning for model compression"*[50] we know that there exist a wide range of reasons that lead to the widespread of several different categories of *Deep Neural Networks related Compressing Techniques*.
In fact, as *Michael H. Zhu et a.l* stated, over the past few years, deep neural networks have achieved state-of-the-art performance on several challenging tasks in the domains of computer vision, speech recognition, and natural language processing. Driven by increasing amounts of data and computational power, deep learning models have become bigger and deeper to better learn from data. While these models are typically deployed in a data center back-end, preserving user privacy and reducing user-perceived query times mandate the migration of the

intelligence offered by these deep neural networks towards edge computing devices. Deploying large, accurate deep learning models to resource-constrained computing environments such as mobile phones, smart cameras etc. for on-device inference poses a few key challenges.
In particular, they further observe that state-of-the-art deep learning models routinely have millions of parameters requiring $O(MB)$ storage, whereas on-device memory is limited. They also claim that, it is not uncommon for even a single model inference to invoke $O(10^9)$ memory accesses and arithmetic operations, all of which consume power and dissipate heat which may *drain the limited battery capacity and/or test the device's thermal limits.*

These challenges, describe just above, lead to early works in the 1990s, as still *Michael H. Zhu et a.l* reported in their works, where as an instance studies conducted by LeCun et al.(1990)[25] or Hassibi et al. (1993)[19] investigated and so performed pruning using a *second-order Taylor approximation* of the increase in the loss function of the network when a weight is set to zero. when speaking about another paper from LeCun et al.(1990)[25] which is referred to as ***Optimal Brain Damage*** the saliency for each parameter was computed using a diagonal Hessian approximation, and the low-saliency parameters were pruned from the network and the network was retrained. While, in ***Optimal Brain Surgeon*** (Hassibi et al., 1993)[19] he saliency for each parameter was computed using the inverse Hessian matrix, and the low-saliency weights are pruned and all other weights in the network are updated using the second-order information. As we can understood from those two seminal early works about Compressing based techniques, are mainly approaches that adopt extensively numerical approaches which may be affected by numerical issues if input data are not properly treated, which means that, if missing data or too correlated data are present within input data set, them may let training procedure done via one of the previously cited methods may degrade final performance or accuracy, due to numerical operation applied to Hessian matrix derived from set of learnable net's parameters.

However, *Michael H. Zhu et a.l* also made reference to more recent techniques still speaking about Compressing based methods. More precisely, they describe briefly works related to ***magnitude-based weight pruning methods***. In fact, they suggest that, such approaches ave become popular techniques for network pruning as in those research studies as driven by Han et al. (2015b,a)[18], See et al.(2016)[38], and finally Narang et al.(2017)[32]. Generally speaking we can say that *Magnitude-based weight pruning techniques* are computationally efficient, scaling to large networks and datasets.

It is also known that, as *Michael H. Zhu et a.l* briefly discussed, while pruning focuses on reducing the number of non-zero parameters, in principle, model pruning can be used in conjunction with other techniques to further reduce model size. So a number of other technique have been explored in conjunction with pruning-, as well as, numerical-based or hessian-based techniques to constitute mixed or hybrid approach to further push both compressing degree as well as accuracy or more in general accuracy of trained Deep Net Models. In fact, as *Michael H. Zhu et a.l* depicted in their own paper, we clearly understand that ***Quantization Techniques*** aim to reduce the number of bits required to represent each parameter from *32-bit floats* to *8 bits or fewer*. They recall some seminal works with pros and cons as reported just below:

- Different *quantization techniques* such as *fixed-point quantization* by Vanhoucke et al. (2011)[41] or *vector quantization* by Gong et al. (2014)[14] achieve different compression ratios and accuracies but also require different software or hardware to support inference at runtime.

- Pruning can be combined with quantization to achieve maximal compression as in Han et al.(2015a)[17].

We also discovered that, in addition to all the previously cited compressing techniques for deep nets, an emerging area of research resalts be low precision networks where the parameters together with activations are quantized to 4 bits or fewer integer-precision, as in the following array of seminal research papers: Courbariaux et al.(2015)[7], Lin et al.(2015)[27], Hubara et al.(2016)[21], Rastegari et al.(2016)[37], and lastly Zhu et al.(2016)[46].

Chapter 3

Experiments

3.1 Distiller Framework

As we can read from related paper[51], ***Distiller*** is nothing but *an open-source Python package for neural network compression research.* In fact, *Network Compression* can reduce the footprint of a neural network, increase its inference speed and save energy. Distiller provides a *PyTorch*[35] environment for prototyping and analyzing compression algorithms, such as sparsity-inducing methods and low precision arithmetic. Distiller contains:

- A framework for integrating pruning, regularization and quantization algorithms.
- A set of tools for analyzing and evaluating compression performance.
- Example implementations of state-of-the-art compression algorithms.

Where, the main motivations that lead Distiller's developers to build up directly from the ground up to the top a Compressing Compliant and Supporting Framework for enabling Deep Nets Compressing Techniques are the following. As Distiller's developers describe, large models are also memory-intensive with millions of parameters. Moving around all of the data required to compute inference results consumes energy, which is a problem on a mobile device as well as in a server environment. Data center server-racks are limited by their power-envelope and their *ToC (total cost of ownership)* is correlated to their power consumption and

thermal characteristics. In the mobile device environment, we are obviously always aware of the implications of power consumption on the device battery. Inference performance in the data center is often measured using a *KPI (key performance indicator)* which folds latency and power considerations: inferences per second, per Watt (inferences/sec/watt). Moreover, Distiller's developers also claim that the storage and transfer of large neural networks is also a challenge in mobile device environments, because of limitations on application sizes and long application download times. They believe, in fact, that sparse neural networks hold the promise of speed, small size, and energy efficiency. For these reasons, they wish to compress the network as much as possible, to reduce the amount of bandwidth and compute required.

Other interesting motivations that Distiller's programmers cited about their own deep nets compression framework, and so, that further characterize such a compressing framework are related to some features that so trained and treated deep models will acquire, which can be represented by the fact that those resulting architecture would be smaller, faster, as well as more energy efficient. In fact, speaking about final neural net size, Distiller Framework's designers noticed that *Sparse NN model representations* can be compressed by taking advantage of the fact that the tensor elements are dominated by zeros. . The compression format, if any, is very HW and SW specific, and the optimal format may be different per tensor (an obvious example: largely dense tensors should not be compressed). The compute hardware needs to support the compressions formats, for representation compression to be meaningful. Compression representation decisions might interact with algorithms such as the use of tiles for memory accesses. Data such as a parameter tensor is read/written from/to main system memory compressed, but the computation can be dense or sparse. In dense compute we use dense operators, so the compressed data eventually needs to be decompressed into its full, dense size. The best we can do is bring the compressed representation as close as possible to the compute engine. Sparse compute, on the other hand, operates on the sparse representation which never requires decompression (we therefore distinguish between sparse representation and compressed representation). This is not a simple matter to implement in HW, and often means lower utilization of the vectorized compute engines. Therefore, there is a third class of representations, which take advantage of specific hardware characteristics. For example, for a vectorized compute engine we can remove an entire zero-weights vector and skip its computation (this uses structured pruning or regularization).
While speaking about faster deep compressed models, we can report what Distiller's developers intent for saving more computing time when smaller and compressed

models are trained to be later employed for inference tasks once those models have been deployed in production. In fact, tanks to Distiller framework we know as well as from the following observation that many of the layers in modern neural-networks are bandwidth-bound, which means that the execution latency is dominated by the available bandwidth. We can take advantage of compressed model's schema to improve performance ant computational time. So, due to the fact that the hardware spends more time bringing data close to the compute engines, than actually performing the computations, where Fully-connected layers, RNNs and LSTMs are some examples of bandwidth-dominated operations, Reducing the bandwidth required by these layers, will immediately speed them up. Furthermore, some pruning algorithms prune entire kernels, filters and even layers from the network without adversely impacting the final accuracy. Depending on the hardware implementation, these methods can be leveraged to skip computations, thus reducing latency and power.
Lastly, analyzing the energy issue Distiller's team attempt to save energy and built a framework that would be more energy efficient. They in particular underline that: because we pay two orders-of-magnitude more energy to access off-chip memory (e.g. DDR) compared to on-chip memory (e.g. SRAM or cache), many hardware designs employ a multi-layered cache hierarchy.Fitting the parameters and activations of a network in these on-chip caches can make a big difference on the required bandwidth, the total inference latency, and off course reduce power consumption. And of course, if we used a sparse or compressed representation, then we are reducing the data throughput and therefore the energy consumption.

3.2 Siren Setting

As we can understand from the preivous section, we established to employ IntelLabs Distiller Framework to carry out our experiments related to adopting and applying different Deep Nets Compression Techniques upon several different initial or intermediate trained models by means of Siren-like Architecture as the backbone of our principal Deep Neural Network. However, as we also have learnt from Siren paper the wide range of usage for which such a model architecture was designed for, we rather focused our mind ans payed our attention on a particular form of employment of such a Deep Architecture. We still use it within Computer Vision Field, but, yer we decided to adopt siren net models not for Classification Tasks but instead for Regression one problems. In particular we attempt to learn models, in more precise words, to learn a set parameters from which the architecture is made that should represent somehow an implicit representation of the original information that we decided to process.
In order to be more explicit, we adopt Siren architecture to investigate its own

properties and capabilities even when some compressing technique have been adopted to reduce original Network's size either getting rid of non-salient weights, or even whole units, or representing weights values with reduce precision, passing from full- or complete-precision representation, a.k.a floating-point, to reduced- or integer-precision, such as 8-bit precision or even lower, as well as a bit higher whole-precision.

So, we briefly recall the main characteristics of Siren architectures that are essential and fundamental in our experiment context, with respect to the overall properties that might fully describe such architecture but that should result too many for our explicit purposes. In fact, Siren architecture, as we adopted to solve our regression problems within Computer Vision Context, is nothing but a Deep Neural Network Model belonging to the family of *Fully-Connected Architectures*, which are also referred to as *Dense Deep Neural Networks*. So such a model is characterized by the fact that between two intermediate layers we can notice that each output unit, from the preceding layer, is in relation with each input unit from the subsequent layer so we have a dense interconnection supported by set of weights that are involved when for each input unit the network has to calculate the resulting activation value to be propagate ahead until reaching the output layer which is in charge of either providing output scores. In our precise setting final output values from the Siren architecture will represent pixel intensity obtained from processing a pair of input coordinates from the grid matrix of the overall coordinates by which input image is compound since. This means that the main task of our fully connected based Siren network is represented by processing a set of ordered pairs made from x and y coordinates for which we aim at estimate the relative pixel intensity or value so that by means of the learnable weights, biases values that all together resemble model's parameters, which represent nothing but an indirect or more precisely implicit representation of the processed input data or information, which is represented by an Input Image, where in the most simple setting such input image would be nothing but a grayscale centered cropped image down to a square size such as 256x256 pixels. Summarizing what sayed just above, we adopted a Fully connected based architecture in order to process a grid matrix of coordinates related to a centered cropped down grayscale image's pixels in order to estimate the as more precise as possible pixel level or intensity by means of learnable parameters represented by weights and biases that all together make up the internal Network architecture.

Once we have specified how we decided to employ or for which precise task we have established to use Siren Network, describing shortly which kind of internal architecture Siren models lay on, that is a Fully connected Structure adopted for Computer Vision related task, which result to be regression problems for estimating

pixels intensities or levels from a grid of ordered coordinates so that learnable models' parameters should be an implicit derived representation of the image itself a train time we attempt to represent, we spent few words around brief description of which kind of activation function, e.i. non-linearity employed by the network along each layer for computing output values to be propagated from layer to layer until the end of the architecture itself, have been chosen by the Siren's designer to let such a model be unique within its genre or family of related architectures. In fact, reading Siren's papers we can read about the fact that the current adopted non-linearity or non-linear function is nothing but a trigonometric based function such as sine math function, which enables accurate representations of natural signals, such as images, audio, and video in a deep learning framework, however we narrow down our focus just to implicit image representations. The advantage of using such trigonometric non-linearity that is nothing but a periodic activation is that this non-linear function is able to fits complicated signals, such as natural images in our cases and to just mention few scenarios, and even reveals to be capable of representing details in the signals better than *ReLU-MLPs*, or positional encoding strategies proposed in concurrent work[31].

Another interesting feature that Siren based architectures issue when describing and speaking about their own properties is that, differently from many others already purposed deep architectures that we can read about in several papers already published in literature, such a Deep Model does not adopt a standard or widely recognized weight and biases initialization schema such as that known as *Xavier Deep Nets initialization Schema*, or more shortly just *Xavier Initialization*, as we can read about within related research paper to such an interesting parameters initialization method in [13]. In fact reading about Xavier Initialization, or also referred to as Glorot Initialization, we clearly understand that it is an initialization scheme for neural networks. Biases are initialized be 0 and the weights W_{ij} at each layer are initialized as:

$$W_{ij} \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}] \tag{3.1}$$

where U is a uniform distribution and n is the size of the previous layer (number of columns in W).

In Siren paper description[40], instead we learnt about the fact that as the statement of the Siren's initialization scheme declare, ropose to draw weights according to a uniform distribution:

$$W \sim U[-\sqrt{\frac{6}{fan_{in}}}, \sqrt{\frac{6}{fan_{in}}}] \tag{3.2}$$

where, they aim that this leads to the input of each sine activation being Gauss-Normal distributed, and the output of each sine activation approximately arcsine-distributed with a standard deviation of 0.5, providing even overview of the proof that we accept greatly. These observations proven by Siren developers within their own published papers allow us to also state that since such output weights and parameters distributions have been theoretically as well as empirically found out, these precise final probability distributions ensure us that when we are going to adopt some kind of quantizing technique for compressing siren architectures some properties from statistical quantization theory still hold. In other words even when probability data distribution for model's quantized parameters do not follow a uniform like distribution but rather a Gauss-Normal distribution we can still recovery original non-quantized parameters with a marginal quantizing error, and such statement is even more acceptable when Gauss-Normal parameters distribution reaches zero as much faster as possible though such a probability distribution is not bandwidth-limited and support compact, as stated within works written by *Bernard Widrow and István Kollár* in their textbook known as Quantization Noise[45]

3.2.1 Input Dataset

After having briefly exposed the properties and characteristics of both the framework or tool and programming language, by means of which we will collect data related to trials we are going to perform, as well as the setting and context within which Siren Architecture will be employed, we go ahead describing which kind of data set we will adopt for attempting to collect results coming from different trials we should carry out. As we already mentioned in the previous section about Siren scenario or context where we are going to use, our input dataset should be represented by nothing but an Input Image, possibly Grayscale image that once selected will be adapted to both time and available resources constraints, so that we decided to centering crop whatever image we decided to adopt down to 256x256 pixels, in order to reduce the overall amount of data in a compatible way with time and hardware resource limited constraints.

We decided to adopt Cameramen target input image, as also Siren's developers have already done for their own trials and experiments, which has been reported just above in its whole size:

However, as we have already anticipated just in the sections above, we decided not to adopt for our trials the entire image, but rather to crop from its center down to 256x256 pixels, so reducing both in width and length the original untreated image, which result in a smaller image as follows:

Figure 3.1: Camera 512x512 whole image

Figure 3.2: Cropped Camera 256x256 target image

Image Feature	Value
name	Camera
shape	(256, 256)
size_byte	65536
image_band	(L,)

Table 3.1: Cropped Image Main characteristics

The reduction both in width and length lead us to obtain starting from the original a new smaller image that is four times smaller in size, leading to a 75% of memory usage reduction to store and process such input image. However, we also have add a look to how the original pixel values distribution have changed with respect to the pixel values distribution we will obtain after having cropped from its center the original image, to understand and let arise some questions about how training will be affected from such a choice. In fact looking at the two distinct pixel values distribution that are reported within the same chart for comparison reasons as follows:

We can immediately notice that even if the pixel values distribution related to cropped image has been subject to an important reduction nearly 75% with respect to the entire original distribution from the original non-cropped cameramen image, the distribution from cropped cameramen images is somehow still similar to the original one, even if some consideration still shoudl be done referring to some sub-intervals we can identify from the whole range of possible values that a single pixel can assume from 0 up to 255, since we are dealing with a grayscale

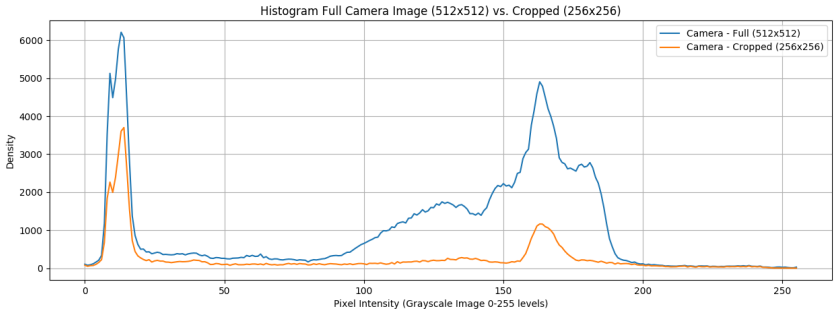


Figure 3.3: Pixel Values Distributions for both Original and Centered Cropped Cameramen image

image with pixel values that are within the just mentioned above span from 0 up to 255, since each pixel is encoded as 8bit unsigned integer. In fact as we can seen from the picture above where a blue line color has been assigned to original pixels distribution while, orange one for cropped image pixels distribution, while the highest picks has been still the highest even if resized because of cropping procedure applied to original image, where these picks can be located respectively within $[0; 50]$ and $[150; 250]$ sub-ranges, we cannot state the same things for secondary and so lower picks that instead have seen their distribution to resize dramatically down toward number of occurrences that resembles those pixel values frequencies much lower than 1000 samples. In particular those secondary picks that have seen their frequencies to reduce in considerable manner lay within pixel values sub-ranges as $[100; 150]$, and $[150; 200]$ respectively. This observation leads us to raise a question whether the experiments we will carry out in order to collect results and statistics about compressed Siren based networks when employed for learning an implicit representation from input image can be translated to full size image in those case when full and cropped images shown not so much different pixel values input distribution. In the context of this thesis work we do not investigate the potential answer to the question just stated above, but focus on retrieving results that should represent our initial baseline for further subsequent improvements or other investigations.

3.2.2 First Trials done with Siren Based Architecture

Before starting explicitly with adopting different compression techniques centered or focused only and mainly around pruning like and quantizing aware compressing methods, we employ Siren Architecture to run a considerable wide number of attempts and trials in order to collect basis results that allow us to constitute the so called control group, that is, a set of samples represented from already trained

and more precisely overly over-trained Siren based architectures which have been learnt and estimated as models without using any kind of deep nets compression techniques. We do not limit ourselves to just run and carry out training with plain Siren architecture but we even produce results employing a well established compression algorithm or technique for representing input target image to compressed to final image characterized from a lower amount of data for their own representation, such as Jpeg standard.

Among the other reasons for which we spent some time training plain Siren Architectures we have the fact that we want at least to provide a fist measure of comparison of this precise kind of Deep Model when compared against another technique which is not focused around the employment of any source of Deep Neural Network Architecture, as well as, to obtain a basis data sets of samples that may be used for later comparisons against those examples that we will obtain by running experiments and trials that will be focused instead on compressing somehow original and untreated plain Siren architecture to reduce such models both in size and memory consumption while still not corrupting or wasting selected model's performance metrics with respect to original non-compressed model's performance scores.

Considered Performance Metrics for Allowing Comparisons

Still before showing most insightful summarizing graphics about our early attempts related to representing processed cameramen image both via Jpeg compressing tool and plain siren models, we should decide and then pick the best perforamnce metrics that result to be suitable and adequate for comparing differences that naturally arise when distinct techniques are adopted. Within the view provided by Computer Vision field we know that there exist several diverse metrics and scores that one can choose from in order to estimate performance from the algorithm we try to adapt for the purposes we attempt to reach. To cite few in such a field and related to image processing, we recall: *Means Squared Error (MSE)*[42], *Peak-Signal-to-Noise-Ratio (PSNR)*[8], *Structural Similarity Index(SSIM)*[33, 44], and many others as well.

In particular, speaking about those few just above cited image quality measures related to processed input items such our target images, we can recall few thing for each of them to just point out or highlight the main reasons for them to be requested for our purposes. More precisely, and starting out from MSE quality measure, we claim that rather to directly using it for representing any kind of

graphics, we exploit such metric to compute more interesting PSNR quality score for the reasons we explain in the following. However referring such metric to just Siren architectures, we claim that such function, represented by MSE score function will be employed mainly as loss function or cost function at training time for let models to learn the as right as possible parameter estimates that allow the models to correctly infer pixel levels once them are fed via order grid matrix of picture coordinates. Such a target function which is involved into the optimizing problem by which Siren Training phase is characterized is part of the Optimizer Scheduler identified, then fixed and adopted for the rest of our experiments represented by Adam Optimizer (Diederik P. Kingma, 2015)[24]. That precise optimization algorithm was used by us within our trials due to the fact that yet Siren developers fixed such learning procedure as their reference optimizer. Furthermore, reading Adam's paper we can clearly understand the numerous advantages we can gain from it. In fact, It represents:

- an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.
- a method that is is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters.
- The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients.

Once explained the ways we consider in our thesis, that is represented by Adam Optimizer algorithm that embedded such quality measure in particular when referring to regression problems as it is the case in our scenario, we go ahead looking at how result really important and useful PSNR quality score. We start by saying that generally speaking, PSNR is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. Where, PSNR is usually expressed in terms of the logarithmic decibel scale, due to the fact that many signals have a very wide dynamic range, i.e. ratio between the largest and smallest possible values of a changeable quantity. The need of using a metrics like Psnr depends on the fact that usually image enhancement or improving the visual quality of a digital image can be subjective. Saying that one method provides a better quality image could vary from person to person. For this reason, it is necessary to establish quantitative/empirical measures to compare the effects of image enhancement algorithms on image quality. Using the same set of tests images, different image enhancement algorithms can be compared systematically to identify whether a particular algorithm produces

better results.
Mathematically speaking we discuss the PSNR implementation we adopt recalling that we assume to deal with with a standard 2D array of data or matrix, related to Grayscale Images, as well as we should guarantee that dimensions of the correct image matrix and the dimensions of the compressed or learn image matrix must be identical. Under those assumption we represent PSNR by:

$$PSNR = 20 \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right), \tag{3.3}$$

where the MSE is:

$$MSE = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} |f(i,j) - g(i,j)|^2 \tag{3.4}$$

where, normally, f epresents the matrix data of our original image, g represents the matrix data of our estimated image in question, m represents the numbers of rows of pixels of the images and i represents the index of that row, as well as, n represents the number of columns of pixels of the image and j represents the index of that column. Lastly MAX_f is the maximum signal value that exists in our original *"known to be good"* image. The proposal suggested by PSNR is that the higher the PSNR, the better degraded image has been reconstructed to match the original image and the better the reconstructive algorithm. This would occur because we wish to minimize the MSE between images with respect the maximum signal value of the image. We should notice that when trying to compute the MSE between two identical images, the value will be zero and hence the PSNR will be undefined (division by zero). The main limitation of this metric is that it relies strictly on numeric comparison and does not actually take into account any level of biological factors of the human vision system such as the structural similarity index. (SSIM). FInally, in a more general settings that includes colour images, the MSE is taken over all pixels values of each individual channel and is averaged with the number of colour channels. Another option may be to simply perform the PSNR over a converted luminance or grayscale channel as the eye is generally four times more susceptible to luminance changes as opposed to changes in chrominance. Final remarks we want to add about PSNR are that, This precise metric will represent together with Bit-per-pixel measure the two most important measurements that we aim at representing as an instance via scatter plots where we want to show and illustrate at several distinct working points how well jpeg, plain siren models, as well as compressed siren version learn nets are going, comparing the resulting pairs of coordinates made from PSNR, Bit-Per-Pixels values along y-axis and x-axis respectively.

Another image quality measure that we known we may desire to employ for comparing different results we are going to collect and perform, via compression techniques for obtaining compression version of plain fully precision Siren based Deep Nets might be represented from Structural Similarity Index Measure, more shortly SSIM. As we can understand reading about such an image quality score, it is nothing but a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. So, SSIM in mosto of the cases is used for measuring the similarity between two images, and it is even considered to be a *full reference metric*, which means that e measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as reference. So, from such a brief description we clearly suggest that it is reasonable a meaningful metric to be employed for our experiments, however, we prefer to employ more prominently PSNR score together with Bit-Per-Pixel measure, as owr reference performance metrics for later discussion and analyses. In particular, the rationale behind this choice is motivated by the fact that SSIM values, as we had observed and appreciated after empirical experiment were undergone, seem to share mostly the same results at least comparing different SSIM quality measures related to different trials. In fact we need to go further several decimal very often digits before finding out some signs of dissimilarity of models performance to understand which was going better than the others for our tasks, that is attempting to learn a set more or less wide of parameters that constitute a whole Deep Nets which aims at representing implicitly a target image by means of a grid of ordered pixels cartesian coordinates.

However, we still compute such interesting performance metric and store it at least for future investigations or further analyses that might be carried out after this very first analyses that we are going to compute within the space of this precise thesys work. In fact we reprot shortly a math explanation of such image quality score. As we have already sayd about SSIM, it is a *perception-based model* that considers image degradation or as in owr case image processing as *perceived change in structural information*. It also incorporates or is made from important *perceptual phenomena*, including both *luminance masking* and *contrast masking terms*. I differs from other image quality measures such as MSE and PSNR meanly because those latter scores are focused on estimating *absolute errors* as their mean to convey useful insights about how well a given procedure or processing technique affect a certain input target piece of information as an instance image. Instead, structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close, These dependencies carry important information about the structure of the objects in the visual scene. Laslty, luminance masking is a phenomenon whereby image distortions or more generally image processing tend to be less visible in bright regions, while contrast masking is a phenomenon whereby

distortions become less visible where there is significant activity or "texture" in the image.

Mathematically speaking, SSIM algorithm is given as follows, assuming that this index is calculated on various windows of an image, and the measure between two windows x and y y of common size NxN is[43]:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_1)} \tag{3.5}$$

with:

- μ_x the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy}^2 the covariance of x and y;
- $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator, in other words, for numerical reason issues;
- L the dynamic range of the pixel-values, typically $2^{\#bits\ per\ pixel} - 1$
- $k_1 = 0.01$ and $k_2 = 0.03$ by default, that means them have been empirically derived.

Other math properties that have been studied in [4] state and tell us that SSIM satisfies the non-negativity, identity of indiscernibles, and symmetry properties, but not the triangle inequality, and thus is not a distance function. However, under certain conditions, SSIM may be converted to a normalized root MSE measure, which is a distance function. The square of such a function is not convex, but is locally convex and quasiconvex, making SSIM a feasible target for optimization. However we adopted Adam optimizer equipped with MSE loss function as reference target function to be opyimized in order to learn the best parameters for implicit image representation. Finally we report that SSIM is still employed in a number of different applications, such as *Image Compression* as described in [42], *Image Restoration* as depicted in [42], *Pattern Recognition* as illutstrated in [10]. Due to its popularity, SSIM is often compared to other metrics, including more simple metrics such as MSE and PSNR, and other perceptual image and video quality metrics. SSIM has been repeatedly shown to significantly outperform MSE and its derivates in accuracy, including research by its own authors and others. Since in a

paper[20], an analytical link between PSNR and SSIM was identified, so it might result to be intriguing to investigate whether such correlation might happen even when Siren Based Architectures whether compressed or not show same patterns, but because of the small amount of time as well as because of hardware constraints, we left for further extension to this initial thesis work.

Exploring Jpeg and Plain Siren Psnr vs Bpp Scatter Plot

Once we have ended with explanations as well as clarification about our main choice that lead the overall experiment process carried out for collecting several kind of results both from plain applications of Siren like architectures for creating a so called *control group* which will be maded also from cases produced by jpeg standard compression, as well as applying deep nets compression methods for reducing both in size as well as precision as an instance by means of quantizing techniques, we have begun our investigation from analysing which meaningful insights we can gain just comparing plain Siren architectures trained throughout a mostly wide range of distinct hyper-params initial configurations against Jpeg compressing algorithm.

However, for seek of completeness, we roughly say some main properties of Jpeg algorithm just to introduce it as well, with in the context of our experiments. JPEG or JPG is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography. The degree of compression can be adjusted, allowing a selectable trade-off between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality. Since its introduction in 1992, JPEG has been the most widely used image compression standard in the world, and the most widely used digital image format, with several billion JPEG images produced every day. The basis for JPEG is the discrete cosine transform (DCT), a lossy image compression technique that was first proposed by Nasir Ahmed in 1972. So, JPEG uses a lossy form of compression based on the discrete cosine transform (DCT), and this mathematical operation converts each frame/field of the video source from the spatial (2D) domain into the frequency domain (a.k.a. transform domain). A perceptual model based loosely on the human psychovisual system discards high-frequency information, i.e. sharp transitions in intensity, and color hue. In the transform domain, the process of reducing information is called quantization. In simpler terms, quantization is a method for optimally reducing a large number scale (with different occurrences of each number) into a smaller one, and the transform-domain is a convenient representation of the image because the high-frequency coefficients, which contribute less to the overall picture than other coefficients, are characteristically small-values with high compressibility. The quantized coefficients are then sequenced and losslessly packed into the output bitstream.

Once we have done with briefly describing and presenting some characteristics related to Jpeg Image Compressing Standard also involved as reference algorithm against which comparing both plain as well as compressed version of Siren based Deep Net Architectures which are aiming at implicitly representing target image by means of which we attempt to learn some sets of parameters that indeed should as much as possible represent an implicit so indirectly the overall image, that can be obtained by processing a grid of ordered cartesian coordinate systems referring to pixels x, and y pairs along axes, we are ready to go ahead presenting the very first graphics related to measurements for Psnr and Bpp scores, that we have plotted via Scatter Graph to start our analyses with a couple of consideration relative to how well plain siren models are going with respect to Jpeg Standard if any improvement or overcoming is match.

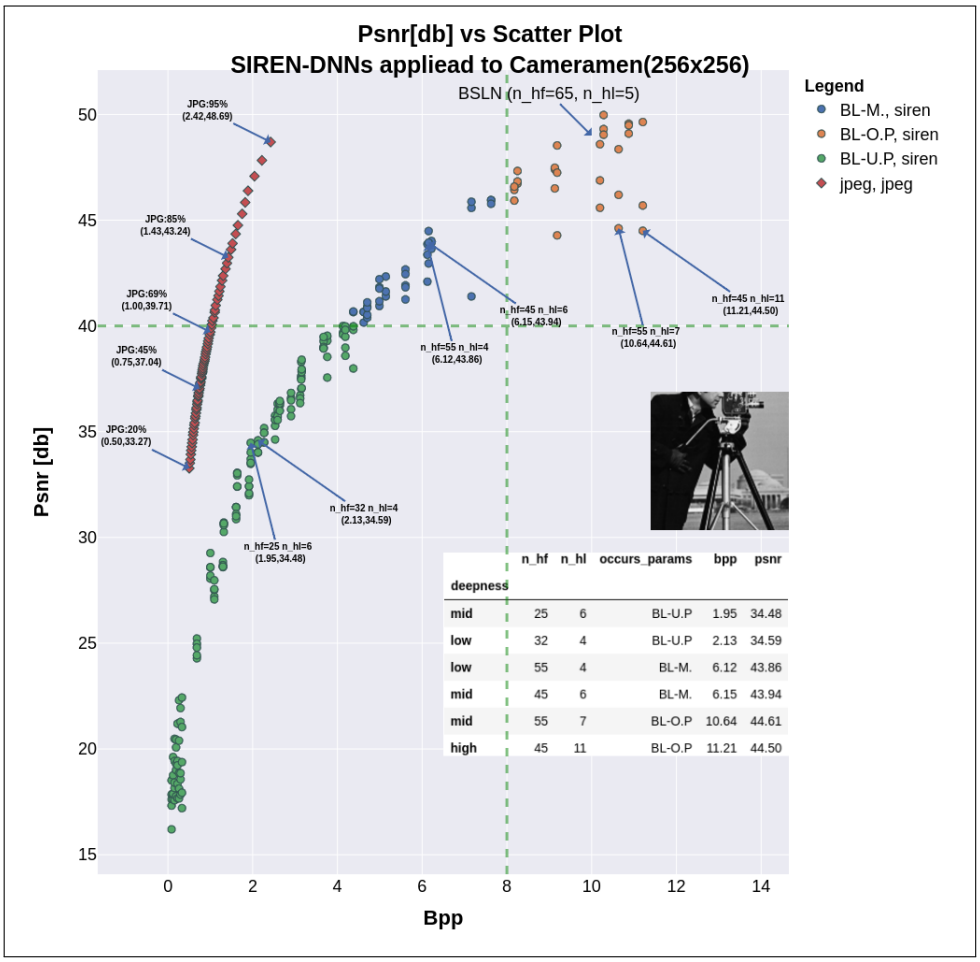


Figure 3.4: Jpeg & Plain Trained Siren Networks: Scatter Plot Psnr[db] vs BPP.

As we can observe from the picture illustrated just below in 3.4, related to values of Pnsr score, expressed in db scale, shown against Bpp measure for each trials we have carried out for siren trained architecture, as well as for each compressed image we could obtain from applying Jpeg algorithm, we focused on treating Cameramen image cropped from its center down to 256x256 of width and length respectively. From the graph itself, we notice that two distinct crowd of points, which resembles to curve drawn within a 2-dimensional plane, where Jpeg data points and Siren tested configurations do not overlap at all, and more precisely Jpeg cluster of points seems to dominate over the whole set of Siren plain architectures, suggesting us that at least without any further processing steps applied to Siren architectures for learning implicit representations of target image by means of weights and other net's parameters, Siren Deep Nets do not compete with Jpeg Algorithm, if we lead our considerations and reasoning comparing trade-off between Psnr values and number of pixels necessary to represent compressed image without ruining the quality of data that is the image itself. To support this initial investigation, we reported and show for each clusters of points, that resemble distinct curves with in Psnr VS Bpp Scatter Plot, some interesting points related to jpeg as well as siren plain trials, where for the latter points we decided even to distinguish them further into at least three separate sub groups depending on the amount of parameters by which siren net are made from. Speaking about Jpeg algorithm related data points, we have decided to try several different attempts that aim at representing in a compressed form target cameramen image to be processed. We definitely toggle or let vary and change a particular parameter that allow obtaining distinct compression degree that eventually show different quality expressed in terms of Psnr score as well as a corresponding Bpp score constituting a precise functioning point within Psnr vs Bpp scatter plot. More precisely, we let quality parameter to vary between somewhat wide range of possible values expressed in percentage terms, that is from 20% up to 95% of quality. So, it results clear that many unusual and normally unemployed quality compression configuration are yet reported, and the main rationale behind such data examples releases on the fact that even such points may results useful for later comparison we results we should obtain from running both plain and compressed trained siren deep models. However for letting the graph being more communicative we explicitly reported some meaningful data pairs made from (psnr, bpp) values for about Jpeg applied algorithm, as also reported directly in the table below, which also holds some additional pieces of information:

As we can notice looking simultaneously toward both Jpeg reported short table 3.2 as well as those data points related to Jpeg cluster, we can notice that nearly

quality(%)	size(byte)	psnr	bpp
20	4133.0	33.27	0.50
45	6115.0	37.04	0.75
68	8083.0	39.59	0.99
85	11679.0	43.24	1.43
95	19863.0	48.69	2.42

Table 3.2: Jpeg Cameramen selected Functioning Data Points

68% of quality assured parameter tuned for jpeg algorithm we obtain data examples that more or less gain a unit value for Bpp score while Psnr image quality measure revolves around 39.59*db* while too far in the down direction we can reach values for Pnsr and Bpp that both decrease, as an instance when we set a compression quality equals to 45% we obtain $\sim 37.04db$ corresponding to nearly 0.75 bpp, which means that while we lose $\sim 3db$ for psnr score, we require 25% bpp for compressing at that precise degree cameramen image, however degradation and reduction become more significant when we even decided to adopt a poor 20% compressing quality which lead to few 33.27*db* at a cost of losing half of data information with respect to a compression quality fixed at 68% or near that value. Conversely, when we are referring to compressing qualities higher than 68%, where the latter as one can quickly understand becomes to represent our reference compressing quality rate for that precise cameramen image, we let Jpeg algorithm to take into account more information, so both Pnsr and Bpp values increases, but at different rate, in other words, with different steepness with respect to the trend with which both Image Quality Measures we can see are growing when compressing quality values are set before reaching 68% compressing rate. In fact in the latter case the steepness is more evident, while moving toward higher compressing quality rate such as 85% we notice that such a steepness is reduced and fade away the more the compressing quality tunable parameter increases toward 95%, which is our higher tested compressing quality for cameramen image. As instances for compressing attempts via jpeg tool, for quality compressing higher than 68%, we reported at least explicitly two data points referring to 85% and 95% of quality rates, respectively. Where, the former allows reaching 43.24*db* at 1.43bpp, which means that a significant increase in number of bits to be employed for each pixels we can notice a marginal increasing in psnr score, differently from the latter case where instead we can appreciate that even if a higher value for bpp than those we have measured for 68% jpeg case, we have gained an important 10db leading to 48.68db for psnr at such high quality rate.

Once we have done with explanations about most relevant features shown from cluster of data points related to applying Jpeg Algorithm throughout a wide span of distinct and increasing compressing quality rates from 20% up to 95%, where nearly 68% we have reported we can appreciate the limit before which bpp will be lower than unity where psnr quality measure ranges from 33 to 39 db, contrary, above which we will experience higher than unity bpp scores that corresponds to psnr quality metric to vary from higher than 39 db until 48 db, we move on presenting salient properties and characteristics of applying plain siren architectures while several distinct model's configurations have been tested, that allow us to identify the potential candidate to be selected for further deep nets investigations carried out by means of deep models compressing techniques.
As we have already mentioned in many rows before, we decided to split or divide the whole cluster of siren related (psnr,bpp) pairs, measured once training procedure was over for each single attempt, into three distinct sub groups depending on the number of parameters from which each net is made and from other additional constraints that we have established and we will discuss in the following. However, before illustrating the constraints we refer above, we reported some information about how different values related to models hyper-parameters lead our tests and how they have been selected.
In fact we decided to let number of hidden layers from which the hidden stack of layers would be made from to vary from 2 up to 4 consecutive layers in the case of models categorized as being low depth, while whit a number of hidden consecutive layers from 5 up to 9 we identify mid depth models, and finally when number of hidden consecutive layers will be chose from 10 up to 12 we identify highly depth models. Such a characterization is not the single one, but we have also described our tested models on the basis of number of parameters per layers mixing such information with both jpeg 68% quality rate that corresponds to unity gain in terms of bpp as well as a vertical axis departing from 8bpp x-axis value which signs the threshold we can report because of the rate we can obtain from un-compressed original cameramen image which is a 256x256 width, height image.
All together, such observations made up our constraints that let us break whole plain siren related data points clusters down into three separate groups of under-, mid- and over-parameterized models, that further allow us to identify special candidate for later employing deep nets compressing methods to gather results for consequently comparisons in terms of performance. For explaining how different configurations lay out within Pnsr vs Bpp graphics we also reported a summarizing table just below, related to some data examples we decided to report for comparing both against jpeg reported values as well as for later investigations when also data points arriving from having trained plain siren models via deep nets compressing techniques.

deepness	size(byte)	n_hf	n_hl	occurs_params	psnr	bpp
mid	16004.0	25	6	BL-U.P	34.48	1.95
low	17412.0	32	4	BL-U.P	34.59	2.13
low	50164.0	55	4	BL-M.	43.86	6.12
mid	50404.0	45	6	BL-M.	43.94	6.15
high	91804.0	45	11	BL-O.P	44.50	11.21
mid	87124.0	55	7	BL-O.P	44.61	10.64

Table 3.3: Plain Siren Deep Net Cameramen selected Functioning Data Points

As one can notice, both from Table 3.3, and looking at the curve that somehow the overall set of data points from Plain Siren cluster approximates, we can notice that such a curve within Psnr vs Bpp cartesian coordinate system also show us that diverse steepness and so increasing trends are characterizing the data points when different combinations of number of hidden layers and hidden features are employed. In fact when we are dealing with those hidden layers, features combinations that we can spot among low-parameterized models the curve's steepness is more evident, while the degree of steepness becomes smoother and lower as we advance passing first through middle-parameterized and finally toward over-parameterized models. However the Plain Siren cluster of data points as well as the imaginable fitting curve that ft as much close as possible to these data is yet dominated and so does not overlap with other curve we can instead fit toward jpeg data points, where the later it is steeper and that fact indicates as jpeg algorithm is itself stil a more competitive algorithm for image processing than siren architectures alone, in other words without any further processing such as employing deep nets related compressing methods.
Looking more closely to the points we decided to report explicitly and directly both on the graph as well as within the Plain Siren Data Table, we can say what follows. For those point that have been characterized as instances of low-parameterized examples, the rationale depends on the fact that such instances expose a combination of hidden layers and hidden features that are not enough to be better than jpeg data points corresponding to a 68% of guaranteed quality, and moreover when compared so against lower jpge guaranteed tested qualities the plain siren data examples are not competitive in terms of numebr of bits required for representing image pixels. In fact as we can see for reported examples where different hidden layers and hidden features combinations that are however similar in terms of number of final overall employed parameters that lead to somewhat similar psnr and bpp

scores, as in the case of $(n_{hf} = 32, n_{hl} = 4)$ and $(n_{hf} = 25, n_{hl} = 6)$, where n_{hf} stands for number of hidden features per layer, while n_{hl} is referring to number of hidden layers employed by that precise siren based architecture, we can understand that both cases are even worst than employing a 45% compressing quality for jpeg, while their bpp is even much more higher than 85% jpeg compressing quality case. While, looking at data points laying within the sub group of middle-parameterized instances, we can learn that as from reported cases, referring to $(n_{hf} = 45, n_{hl} = 6)$ and $(n_{hf} = 55, n_{hl} = 4)$ which are again example of diverse initial hyper-params configurations that correspond to somewhat similar psns and bpp values, that even such points though owning psnr scores similar to that of jpeg points that are nearly 85% of compressing quality we got bpp score for such siren configurations that is also more than nearly three times higher than an even high jpeg compressing quality as 95%. However, such points are so classified because even if shown both psnr and bpp score that are higher than 68% reference jpeg compressing quality, them are not yet too parameterized in the sense that their bpp score is still lower than the number of bits per pixels necessary for describing un-processed cameramen 256x256 image. We should, however, note that those plain siren configurations are not captivating choices since do not seems competitive over jpeg algorithm still not really employable for being subject to further processing via deep nets compressing algorithms at least in our initial analyses. So them will be reported as reference against which comparing later compressed plain siren models.
Lastly, looking at over-parameterized data points, them belong to such a category or class due to the fact that own a so high number of parameters due to their combination made up from number of selected hidden layers and hidden features that both made them be suitable for being subject to deep nets compressing methods as well as we can notice that some among them are at least competitive over a jpeg 95% compressing quality, allowing us to attempt to embark one or more of such plain trained siren models to state whether or not deep nets compressing method allow for reducing model's size without impacting heavily against psnr scores, reducing it by an important extent. For such data examples we reported up to three cases whith in our preliminary Psnr vs Bpp graph. Those points corresponds to $(n_{hf} = 55, n_{hl} = 7$ and $(n_{hf} = 44, n_{hl} = 11)$ that corresponds, as earlier already motivated, to distinct models configurations that both show a final overall number of parameters, with similar resulting scores relative to psnr and bpp measurements. However, such reported point do not show performance that are higher than data points we can identify looking at middle-parameterized trials, even if we can spot over-parameterized examples that own a psnr score higher than those from middle-parameterized istances. However the rationale behind showing those low quality over-parameterized examples is due to the fact that the more the higher are both $(n_{hf}$ n_{hl} the more the number of epochs, or steps we require to the model to converge to solutions that shown low variance, otherwise we end up with

examples that are characterized by high variance leading to examples with poor performance in terms of psnr score, while approximately similar bpp score.
Finally, arriving at the third point we decide to mention among those that are related to over-parameterized plain trained siren models, we illustrated which was the baseline plain siren model selected as our reference to be further processed by means of deep nets compressing approaches. In particular such a point, which is represented as red start with in our Introductory Pnsr vs Bpp Scatter plot, is nothing but a siren model characterized from a low deepness and somewhat intermediate number of hidden parameters for each layer that allow us to select an over-parameterized model that is not too much far a part from reference vertical line at 8bpp of x-coordinates, so that such model owns a 10.28 bpp value of information ration between overall bits and number of pixels from which the original image is made, as well as allowed us to train a basis siren models that is somewhat competitive to jpeg 95% compressing quality, even if comparing to instead 68% jpeg compressing quality we ca notice that our selected baseline model needs 1o times more bits that lead to plus 10 db units in terms of psnr score, as reported also in the following table where we included most salient information about our selected baseline architecture:

deepness	size(byte)	n_hf	n_hl	occurs_params	psnr	bpp
low	84228.0	64	5	BL-O.P	49.97	10.28

Table 3.4: Plain Siren Deep Net Cameramen Baseline Choice

3.2.3 Automed Gradual Pruning Deep Nets Compressing Technique

Once we have effectively ended the introductory description which refers to the illustrated Pnsr vs Bpp scatter plots where both (psnr,bpp) score pairs have been discussed, investigated and motivated both for cases where jpeg algorithm was adopted for compressing original cameramen 256x256 image, as well as, for other cases where instead several different combinations of hyper-params as n_{hf} and n_{hl} for plain siren architectures have been tested training such networks to reach over-trained, over-fitting final states, we have reached an abundant control group from which starting to develop further investigation of selected baseline candidate that will be further processed by means of identified deep nets compressing technique motivated both from their usefulness, as well as IntelLab Distiller framework

support. To be more precise, we decided to begin our study from adopting firstly pruning based compressing technique, then moving over and also applying some kind of quantizing compressing technique to further reduce base line selected model's size as well as attempting to preserve and do not ruin or spoil reached baseline performance, in terms of psnr score.

Theory behind Automated Gradual Pruning Based Compressing Technique

Speaking broadly, with Pruning Techniques applied to Deep Neural Nets Architecture Topologies we meant that the technique, involved during training phase, aim at reducing the number of effective and salient deep nets's parameters, that are weights as well as biases if any, so that the resulting network models get rid of unneccessary weights and/or biases since those nets' parameters do not impact significantly at inference time and during training phase do not impact or contribute widely to improve the learning step. While, from theoretical standpoint the minimum that we should report about Automated Gradual Pruning compressing technique for deep neural networks, is represented from what follows in the current section, before moving ahead presenting and overviewing meaningful results we have collected by appling such technique against our over-trained as well as over-sized fixed baseline model. Automated Gradual Pruning approach[50], more shortly AGP, seeks to prune the network's connections during training. In particular they report that for every layer chosen to be pruned, we add a *binary mask variable* which is of the same size and shape as the layer's weight tensor and determines which of the weights participate in the forward execution of the graph. However, such binary mask variable is part of training and in IntelLab Distiller framework it is involved in the training graph where there are sort the weights in each layer by their absolute values and mask to zero the smallest magnitude weights until some desired sparsity level denoted generally speaking as s is reached. We also has to notice that the back-propagated gradients flow through the binary masks, and the weights that were masked in the forward execution do not get updated in the back-propagation step. In fact, Michael H. Zhu et al(2017)[50] were able to introduce a new automated gradual pruning algorithm in which the sparsity is increased from an *initial sparsity value* s_i, usually set starting from zero, to a *final sparsity value* s_f over a span of n *pruning steps*, starting at *training step* t_0 and with *pruning frequency* Δ_t, as illustrated by the following math formula, which governs the pruning algorithm that so becomes also part of training phase:

$$s_t = s_f + (s_i - s_f(1 - \frac{t - t_0}{n\Delta t})^3), t \in \{t_0, t_0 + \Delta t, \ldots, t_0 + \Delta n\} \tag{3.6}$$

From the math equation 3.6, we notice that the *binary weight masks* necessarily involved in the pruning process will be updated every Δ_t as the network is trained to gradually increase the sparsity of the network while allowing the network training steps to recover from any pruning induced loss in accuracy. An interesting feature, that also have been introduced whit in AGP algorithm implementation provided by IntelLab Distiller framework, is that one that enable the model to achieve the target sparsity s_f, the weight masks are no longer updated. This property has the advantage over other well-known and established pruning techniques such as some Magnitude based pruning approaches, where several attempts should be done before knowing the amount of time through which let model to be pruned to carry out several steps and epochs to fulfill the compression task as well as many different derived threshold as an instance from statistics collected ahead of training time for weight values distributions per layers, since we are freed from the necessary or mandatory need of knowing statistics distributions or properties we have to collect from per layer weights set reducing the amount of time required before identifying promising hyper-params setting the lead the pruning algorithm when it takes action at training time.
Another important feature about such pruning technique, proven by Michael H. Zhu et al(2017) within their paper [50] is that they attempt to prune the network rapidly in the initial phase when the redundant connections are abundant and gradually reduce the number of weights being pruned each time as there are fewer and fewer weights remaining in the network. Furthermore, the pruning method described here does not depend on any specific property of the network or the constituent layers, and can be extended directly to a wide-range of neural network architectures.
Again, Michael H. Zhu et al(2017), within their paper [50], also provided at a high level description some useful heuristics for selecting and picking values related to hyper-params that characterize AGP compressing behavior at training time to derive a compression of the trained models. As an instance, when they spoke about pruning frequency Δ_t they suggest to pick or select values between 100 and 1000 training steps, in order to let models to be pruned to recovery greatly from brain damage, as well as they noticed that pruning frequencies chosen in such a way had a negligible impact on the final model quality. Other interesting heuristics are taking into account relationship between pruning scheduler, that will lead pruning process, and learning scheduler that instead is in charge of leading the optimizer. In particular they highlight the focus on the relationship we should carefully tune between learning rate and pruning frequency need for reaching a desired pruning level. In fact, Optimizers typically decay the learning rate during training, and we have observed that pruning in the presence of an exceedingly small learning rate makes it difficult for the subsequent training steps to recover from the loss inaccuracy caused by forcing the weights to zero. At the same time, pruning with

too high of a learning rate may mean pruning weights when the weights have not yet converged to a good solution, so it is important to choose the pruning schedule closely with the learning rate schedule.

AGP pruning: determining most adequate Hyper-parameters

Before showing the insights and generally speaking the results we have obtained from the choices we have taken in order to accomplish a training phase characterized from having compressed by means of AGP pruning based technique our selected plain siren like baseline model, with $n_{hf} = 64$ and $n_{hl} = 5$ which corresponds to an instance that features over-parameterized example, we dedicate some time to exposing the choices partly guided by several heuristics purposed in literature about how we can compressing a deep nets by means of pruning based approaches such as AGP pruning.
In fact, as mentioned in [50], relative to AGP pruning method itself, we follow partly what Michael H. Zhu et al(2017) suggested relative to how identifying and then selecting best values for updating frequency Δ_t, which is a hyper-parameter in charge of partly guiding pruning algorithm represented by the overall procedure by which AGP is made, because such hyper-parameters when correctly set allows for properly recovery performance that normally get worse and so are somehow wasted and ruined by a kind of damage that happens when some neurons, e.i. units in each layers, have been removed by any kind of pruning algorithm. In particular, They suggest to let Δ_t vary between 100 and 1000 step.
However, due to resource and time constraints, in the context of our thesis, we have initially followed a *Random Search Approach* for selecting only later more promising subsets of plausible hyper-parameters.Then, from the briefly tested and seen trial results from Random Search Approach, we decided instead to let Δ_t vary with in the fixed set of values represented from $\{50, 100, 200\}$.
In fact, we noticed that selecting too lower update frequency factor, we do not let siren like pruned models to recovery from damage and this means that finally those trained and pruned architectures shown to be less competitive both against plain siren models due to ruined performance psnr scores. On the other hand, if we instead let update frequency factor tunable hyper-param to be set with values higher than 500 or 1000 we observe that those choices required too much time for reaching desired pruning rates and so they should represent unfeasible hyper-params choices, that we decided to discard a priori.

Once we have established the update frequency factor related to when or how frequently applying or let AGP pruning routine that will take place along with plain training procedure so that AGP method can prune away a number of not salient weights as well as units in order to reach the wanted amount of selected pruning

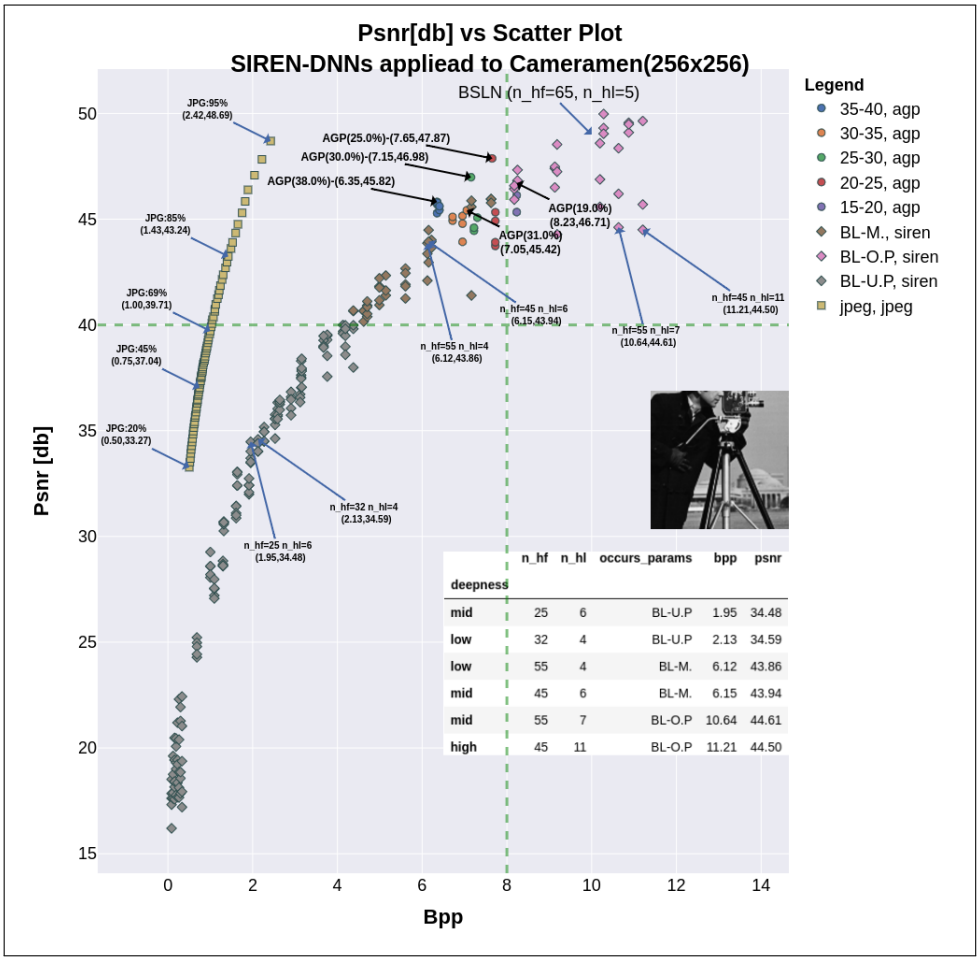
rate, we go further giving reasons for how distinct pruning rate have been derived, how generally speaking the layers from which models are made have been treated or pruned individually. We, first, recall that in literature normally the choice of pruning rate have been suggested to be performed by means of a particular ahead of pruning time technique named *Sensitivity Analysis*. Such a pruning rate selection or deriving approach is motivated by the fact that we should treat each layer, especially hidden stack of several layer, in different manner, without necessarily adoping the same pruning degree for each intermediate layer beacues, these hidden layers do not have the same tolerance to equal pruning degree. In other words, *sensitivity pruning analysis* aims at identifying the most suitable pruning amount for each layer depending on some layer's information. In particular pruning level should be dependent from layer location within net architecture stack, as well as from how salient it results when only that precise layer alone is subject to pruning without touching other remaining layers. This means that, *sensitivity analysis* for being carried out requires both a given amount of time before a selected pruning technique actually can be performed, as well as, needs to test an array of several different pruning rates to establish the most suitable pruning amount for each layer. From literature, however we know that, roughly speaking a shared heuristics within deep learning community related to compressing methods is that , the deeper the layer the greater the amount of pruning we can decide to apply. This is mainly justified by the fact that we do not want information, that flow from input data through different intermediate layers while being processed, can be heavily truncated - because of reduced number of units within early layers that implicitly re-project input data with in lower dimensional representational spaces - since from the very beginning because that will produce a loss of information that will affect the final prediction and heavily compromise model performance. On the other hand, more aggressive pruning amounts have been proven to be more tolerant by pruned deep net models so lead to less wasted performance with respect to original non-pruned corresponding models because of the efficiency with which the pruning method at deeper layers are able to remove non-salient neurons, e.i. layer's units, that do not allow for appreciable improvement for predicting out values from the model.

Having provided reasons for applying *sensitivity analysis* before proceeding with training baseline model for pruning and so compressing it, we follow suggestions provided by such a preliminary analysis method. We have done also other considerations before illustrating the various degree of pruning levels selected according with layers characteristics partly derived from *sensitivity analysis*, in fact we consider due to the particular architecture we fixed for siren like plain and baseline models, to just prune hidden layers and more precisely to prune only weights matrices

for each hidden layers, leaving untouched both biases arrays, in fact, each hidden layers as well as the very first - also referred to the input embedding layer from pixels domain to implicit intermediate multidimensional spaces to get intermediate representations functional for our predicting task - as well as the last whole layers. The reason is meanly due to the fact that both input and output layers as well as intermediate biases arrays for each intermediate and hidden layers do not account to huge amount of models' parameters when compared to instead the overall amount of weights that hidden layers account for. That choice has also another important result, allowing first and last layer to take into account all the units from which they are made from, allow nets to preserve as much input information as possible for the former layer, while to get the most precision when speaking about the latter, avoiding the final pruned models to ruing or impact importantly in a worst manner to performance we instead want to preserve as much as possible when comparing pruned models to their corresponding original baseline models.
Arrived at this point, we illustrate how we decide to prune individually hidden layers on the basis of *sensitivity analysis* and other additional considerations. In particular, the hidden layer right after the input embedding layer was pruned sampling a pruning rate that ranges within 10% up to 20%. Instead, for middle hidden layers except the one right before the output layer, we adopted or have sampled a pruning rate that ranges within 25% up to 40%. Finally, we selected or have sampled a pruning rate for the hidden layer right before the output layer that whose value ranges within 45% up to 55%.

AGP pruning: Results

With all the observations we have done for justifying how our deep nets, derived from selected baseline plain siren architecture, compressed following a pruning like compressing algorithm such as AGP pruning based procedure we have obtained the subsequent results as shown within the graphics provided below:



deepness	n_hf	n_hl	occurs_params	bpp	psnr
mid	25	6	BL-U.P	1.95	34.48
low	32	4	BL-U.P	2.13	34.59
low	55	4	BL-M.	6.12	43.86
mid	45	6	BL-M.	6.15	43.94
mid	55	7	BL-O.P	10.64	44.61
high	45	11	BL-O.P	11.21	44.50

Figure 3.5: Jpeg & Plain and Pruned Siren Networks: Scatter Plot Psnr[db] vs BPP.

Looking at the graph we have reported just above in picture 3.5, and more precisely focusing our attention to AGP pruning data examples, we can notice how we have pruned our baseline model, as we have divided our pruned models into five distinct groups created grouping data on the bases to which interval their pruning rate fall into. Where those pruning intervals are equally sized, spanning by 5 percent units, from 15% up to 40 % so that we have formed up to four distinct groups.

Starting our discussion from those pruned siren like models that fallen within 15%-20% pruning overall amount interval, clearly, we can state that those pruning levels are not enough to cross and overcome the vertical threshold line located at 8bpp on the x-axis which represents a constraints that we want our pruning

model to pass if we want to migrate pruned models into Pnsr vs Bpp coordinate system regions, where those models result to be no more over-parameterized models. However, such poorly pruned siren based instances show to be better than some over-parameterized plain siren based models, such as plain siren models with $n_{hf} = 45$ and $n_{hl} = 8$ which corresponds to a mid deep architecture with a psnr value equals to $\sim 45.92db$ and a bpp score eqauls to ~ 8.11, but at the same time those pruned models went somehow worst than other plain siren cases such as plain siren models with $n_{hf} = 64$ and $n_{hl} = 4$ which corresponds to a shallow architecture with a psnr value equals to $\sim 47.0db$ and a bpp score equals to ~ 8.25, as also we have reported within the summarizing table:

baselines	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
(chosen model*)	(64, 5)	84228	49	10
(other plain model)	(45, 8)	62384	46.59	8.17
(other plain model)	(64, 4)	58954	47.33	8.25
agp (prune intervals)	**prune_rate(%)**	**size(byte)**	**psnr(db)**	**bpp**
15-20	19.0	67420	46.71	8.23
20-25	25.0	62688	47.87	7.65
25-30	30.0	58592	46.98	7.15
30-35	31.0	57780	45.42	7.05
35-40	38.0	52040	45.82	6.35

Table 3.5: Best Siren AGP pruned cases

When we are looking at those interval pruning rates represented by 20-25, 25-30, and 30-35 percent intervals, we noticed that for each sub interval there are more examples that are slightly better than plain siren comparable, in terms of number of parameters, models, but yet there are more cases that instead are somewhat worst than some plain siren architectures. In fact, as an instance our pruned models have been better than plain siren models such as $(n_{hf} = 45, n_h l = 6)$ and $(n_{hf} = 55, n_{hl} = 4)$, with respectively $(psnr = 43.94db, bpp = 6.15)$, $(psnr = 42.43db, bpp = 6.12)$ performance, where the former corresponds to a mid depth arch, while the latter to a low depth arch, while both being mid parameterized models. Conversely, our pruned models, that lay with in one of the previous three pruning intervals, were not able to improve over some other plain siren models such as $(n_{hf} = 45, n_h l = 7)$ and $(n_{hf} = 55, n_{hl} = 5)$ with comparable number of parameters such as $(psnr = 45.87db, bpp = 7.16)$, $(psnr = 45.95db, bpp = 7.63)$ performance, where the former corresponds to a mid depth arch, while the latter to a low depth arch while both being mid parameterized models. In summary, we also report our results, where along with previously stated plain baseline siren models we include also best pruned siren based models we have been able to train for reaching desired pruning rate levels:

baselines (unbeaten)	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
siren(chosen model*)	(64, 5)	84228	49	10
(low-depth, mid-params)	(55, 5)	62484	45.95	7.63
(mid-depth, mid-params)	(45, 7)	58684	45.87	7.16
agp (prune intervals)	**prune_rate(%)**	**size(byte)**	**psnr(db)**	**bpp**
20-25	24.0	63304	45.32	7.72
25-30	28.0	59832	45.07	7.30
30-35	31.0	57780	45.42	7.05
baselines (beaten)	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
(mid-depth, mid-params)	(45, 6)	50404	44.48	6.15
(low-depth, mid-params)	(55, 4)	50164	43.86	6.12

Table 3.6: Best Siren AGP pruned cases. Where low-,mid-,high depth stand for lower than 4 hidden layers, between 5 up to 9 hidden layers, and greater than 9 hidden layers, respectively. Instead low-params stands for baseline models with not enough parameters to overcome psnr score as jpeg compression image with unit bit rate,high-params stands for baseline models with bit-rate greater than input image bit rate, and mid-params stands for baseline models in between.

However, if we focus our attention instead on just those AGP pruned models which correspond to best results we have found when training baseline siren instance involving AGP procedure as our compressing deep nets technique, we notice that at least there are few pruned examples that were able to overcome greatly in term of performance mid-parameterized plain siren models, as well as, have been able to overpass some over-parameterized plain siren instances, as reported within below summarizing table:

unbeaten baselines	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
siren(chosen model*)	(64, 5)	84228	49.0	10
(mid-depth, over-params)	(55, 7)	87124	48.5	10.63
(high-depth, mid-params)	(45, 9)	75244	48.53	9.18
agp (prune intervals)	**prune_rate(%)**	**size(byte)**	**psnr(db)**	**bpp**
25-30	30.0	58592	46.98	7.15
beaten baselines	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
(mid-depth, mid-params)	(45, 10)	83524	46.88	10.19
(mid-depth, mid-params)	(55, 7)	87124	46.19	10.63

Table 3.7: Best Siren AGP pruned cases: Showing which baseline architecture have been overcomed in terms of Psnr score. Where low-,mid-,high depth stand for lower than 4 hidden layers, between 5 up to 9 hidden layers, and greater than 9 hidden layers, respectively. Instead low-params stands for baseline models with not enough parameters to overcome psnr score as jpeg compression image with unit bit rate,high-params stands for baseline models with bit-rate greater than input image bit rate, and mid-params stands for baseline models in between.

Since we can notice that two entries among plain siren architectures reported within the table illustrated just above, where those entries corresponds to the same initial hyper-param configuration that leads to distinct results depending on the initial random weights and biases initialization, that observation again suggest us as it is more difficult learning low variance estimates of siren based models as them begin to become more and more wider and deeper. In other words we can still notice when comparing our pruned models against over-parameterized plain siren models, the latter show a wider variance than AGP pruned instances, where among those compressed deep nets examples we can still identify even competitive results in terms of performance represented by Psnr image quality measure that are better than un-compressed siren counterparts.

The remaining prune interval that we have to discuss is represented by 35-40 percent pruning span. This range corresponds to that with the highest amount of pruning rate that we have attempted to apply and reach when compressing our deep Siren baseline net. The interesting things we can reported about trials we have carried out setting a pruning degree that lays within 35-40 percent pruning span are the following. Those cases show to be less subject to a wider variance with respect to lower selected pruning rate with which carrying out AGP compression. Moreover, we have also several cases belonging to 35-40 percent pruning span that overcome other pruned instances with lower pruning rate. However, best siren architectures compressed by means of a pruning rate higher than 35% show a psnr score that is much lower than corresponding psnr score for other siren compressed case by means of lower pruning rate. But, due to the fact that these last samples from found within 35-40 percent pruning span have experienced a lower variance we decided to pick for the downstream quantizing step, to further compress intermediate pruned siren architectures, best instance identified within this precise group.

Finally, as we compare jpeg data points represented also within Pnsr vs Bpp scatter plot where also we have shown pruned siren instances along with plain siren architectures samples, we can understand roughly speaking what follows. If we consider the overall set of pruned siren architectures by means of AGP pruning method as a unique group of data examples along with the performance we have obtained for psnr score when a further training phase has been conducted for accomplishing AGP compressing technique, we can end up saying that on average our compressed architectures are comparable against 85% up to 90% quality jpeg results but withsignificantly higher rates, in the order of a 78% inrease.

Up to this point, that is, having both shown main features related to both jpeg and plain deep siren architectures where the latter have been trained to reach over-trained state and which do not have been further processed, and then having

also added the discussion and insights about AGP compressing most relevant theory aspects as well as strategy adopted for identifying best or most suitable sets of values related to AGP pruning algorithm hyper-params so that such compressing can be applied and finally the results collected from recorded models' performance, we can state what follows. Before moving on with next step, we have gain some interesting results from having just pruned away non-salient weights or even units from hidden stack of siren architecture's layers to compress nets without reducing yet numeric precision as we will do in the remaining sections. In particular, by testing several different overall pruning degree, which can however be divided into pruning rate ranges into which them are then grouped, we have seen that we were able to reduce baseline siren model's size still obtaining processed architecture that can even overcome other siren plain models with more or less same number of overall net's learnable parameters for implicitly representing target image we aim at learning, more specifically such overcome models have been found among both mid- and over-parameterized plain siren architecture as depicted in previous sections. However, on average our pruned models, via AGP technique, shown a nearly $\sim 10\%$ reduction relative to Psnr score, where we moved down from $\sim 49db$ for a baseline fixed siren model for our trials corresponding to a low depth examples with $(n_{hf} = 64, n_{hl} = 5)$ which leads to a $\sim 10bpp$ model, which is competitive with a 95% quality jpeg instance but requiring $\sim 75\%$ more memory storage, for reaching similar performance. While looking at the difference between jpeg image compressed psnr image quality scores against compressed by means AGP process compressed siren architectures we can claim that on average our siren compressed models are similar referring to psnr score them have obtained to those jpeg compressed images that are characterized from 85% up to nearly 95% quality, even if agp instances are still on average nearly $\sim 78 - 80\%$ wider in terms of memory footprint.
From the results we have reached so far, by means of Autometed-Gradual Pruning Approach, we can end up saying that there exist some configuratios we can adopt by setting specific pruning rate for each of the hidden layers, from which pruned shallow SIREN picked architecture was made, that are able to overpass other reference shallow SIREN configurations that are similar in terms of achieved bit rate but different somewhat in terms of Psnr score. In particular, keeping more weights while pruning we can increase significantly Psnr score, without spoiling too much the Psrn image quality score, but the differnt pruned models lead to higher intra-variance amongst models with similar pruning rate. On the other and, deeper pruning rate adopted for pruning basic SIREN architecture leads to higher compression but even Psnr values get low quicklier but with models experiencing lower intra-variance when calcualted among pruned models with not so far pruning level.

We, finally, state that for the remaining analyses we have to report about quant-aware training technique we will consider in particular to distinct cases taken from agp pruned siren models, which correspond to a low pruned model and a higher pruned model, more precisely to a 25% and 38% pruned models, where those choices depend on the fact that we want to observe how quantizing methods can improve, if any improvement will be indeed observed, models performance or just to observe which kind of performance we will get. We report below best results found for each pruning rate intervals, as well as we indicated which have become our pruned models that will be subject to quant aware training subsequent phase, which have been marked by means of asterisk symbol:

baseline	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
fixed siren baseline	(64, 5)	84228	49	10
agp (prune intervals)	**prune_rate(%)**	**size(byte)**	**psnr(db)**	**bpp**
15-20	19.0	67420	46.71	8.23
20-25	**25.0***	62688	47.87	7.65
25-30	30.0	58592	46.98	7.15
30-35	31.0	57780	45.42	7.05
35-40	**38.0***	52040	45.82	6.35

Table 3.8: Best Siren AGP pruned cases: Selected Models with asterisk

3.2.4 Weight quantization and Quatization-aware training

Quantization refers to the process of reducing the number of bits that represent a number. In the context of deep learning, the predominant numerical format used for research and for deployment has so far been 32-bit floating point. However, the desire for reduced bandwidth and compute requirements of deep learning models has driven research into using lower-precision numerical formats. It has been extensively demonstrated that weights and activations can be 8-bit integers without incurring significant loss in accuracy. The use of even lower bit-widths, such as 4/2/1-bits, is an active field of research that has also shown great progress. In fact, as we can imagine the more obvious benefit from quantization is significantly reduced bandwidth and storage. For instance, using INT8(8-bit integer representation for both weights, biases as well as even activations) for weights and activations consumes 4x less overall bandwidth compared to FP32(floating point full precision). As already mentioned, we mainly Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference work [23] by Benoit Jacob et.al(2018), which attempt to propose a quantization scheme along with a co-designed training procedure allowing inference to be carried out using integer-only arithmetic while preserving an end-to-end model accuracy that is close to floating-point inference, as we will shortly describe for seek of completeness.

Having done with the earlier section within which we have dedicated most of our discussion about how can we employ a kind of pruning based deep nets compression algorithm, such Automated Gradual Pruning approach whose major properties can be read within Michael Zhu et al (2017)[50] research paper, we move ahead we another section where we focus our attention to a kind of Quant-Aware Training compression algorithm to further reduce in size our already pruned models via AGP procedure, as well as attempting to keep as much as possible unaltered our quantized models performance with respect to pruned counterparts. However, we do not limit our strength for comparing quantized models over pruned ones, but even compare the results we derive from applying one such quantizing algorithm against jpeg and our baseline reference from which we have begun our experiment which consists, as we have already said many times, to understanding which might be the implications of adopting different deep nets compressing approaches applied on siren like models in terms of our selected image quality measures that assess how much well those compressing algorithm can be. There are several reasons for adopting a kind of quantizing compression algorithm for further compressing in size huge deep nets. As we can read from Xiandong Zhao et al. (2020) [47], while the past few years witnessed the success of DNNs on cloud and server-end computers, neural networks have been recently pushed to embedded and mobile areas to enable edge intelligence. For these scenarios, the power provision and

computational strength on the edge computing devices are limited. As a result, it is essential to have more efficient network architectures and less expensive inference overhead, however within the context of our work thesis we mainly focus on the fact that we rather attempt to learn and memorize a implict image representation with the value of the models' weights that we will try to quantize, so that the more compressed. There is increasing attention from the research community to study the compression of modern deep neural networks that are typically over-parameterized and computationally costly, for such reasons we focuses on quantization which have demonstrated to be not only a method to reduce the memory footprint as in traditional work, but also a mandatory step to make the network deployable on integer hardware. Having said that just above, we proceed providing a brief introduction and explanation about the quantizing method we have finally chose for our later experiments - where we will deal with quantizating aware approach for further reduce memory footprint of our compressed both via agp pruning and then quatized models - which is represented from *Qantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*, a seminal work done by Benoit Jaco et al. (2018)[23].

3.2.5 Range Linear Quantization: Theoretical Aspects

Quantization refers to the process of reducing the number of bits that represent a number. In the context of deep learning, the predominant numerical format used for research and for deployment has so far been 32-bit floating point. However, the desire for reduced bandwidth and compute requirements of deep learning models has driven research into using lower-precision numerical formats. It has been extensively demonstrated that weights and activations can be 8-bit integers without incurring significant loss in accuracy, for models addressing classification problems. The use of even lower bit-widths, such as 4/2/1-bits, is an active field of research that has also shown great progress. In fact, as we can imagine the more obvious benefit from quantization is significantly reduced bandwidth and storage. For instance, using INT8(8-bit integer representation for both weights, biases as well as even activations) for weights and activations consumes 4x less overall bandwidth compared to FP32(floating point single-precision). As already mentioned, we mainly follow Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference work [23] by Benoit Jacob et.al(2018), which attempt to propose a quantization scheme along with a co-designed training procedure allowing inference to be carried out using integer-only arithmetic while preserving an end-to-end model accuracy that is close to floating-point inference, as we will shortly describe for seek of completeness.

In particular, they provide a quantization scheme that quantizes both weights and activations as 8-bit integers,and just a few parameters - such as bias vector - as 32-bit integers. They try to provide a quantized inference framework that is efficiently implementable on integer-arithmetic-only hardware such as the Qualcomm Hexagon, however in our thesis research we do not focus onto such low level and hardware compliant issue, and so prefer to postpone such interesting argument for future extension to our primal analyses on collecting meaningful statistics about siren deep nets main performance metrics values when trained adopting any kind of compressing algorithm for deep nets such as pruning, or quantizing methods. Lastly, they applied their frameworks to efficient classification and detection system that showed ignificant improvements in the latency-vs-accuracy trade offs for state-of-the-art architectures.

In this paragraph, we are going to rapidly describe Benoit Jacob et.al(2018) general quantizing scheme, which deals with a correspondence between the bit-representation of values them have denoted with q which stands for quantized value, and their interpretation as mathematical real numbers instead denoted by r for real value. Their quantization scheme is implemented using integer-only arithmetic during inference and floating-point arithmetic during training, where them have attempted for both implementations maintaining a high degree of correspondence with each other. Roughly speaking their quantization scheme reduced to be an *affine transformation or mapping* of interger q to real numbers r of the form

$$r = S\dot{(}q - Z) \tag{3.7}$$

for some constants S and Z, which are their *quantizing parameters*. Benoit Jacob et.al have been able to treat also by means quantization such requirement, in fact, activations can be quantized at points where they would be during inference, e.g. after the activation function is applied to a convolutional or fully connected layer's output,or after a bypass connection adds or concatenates the out-puts of several layers together such as in ResNets. However, we recall, since from the very beginning of our technical overview of such quantization method, that we aim at just focusing on quantization applied to only models's weghts rather than taken into account also biases, in order to simplify our treatment of such quantization procedure. Moreover, We specify that, their quantization scheme uses a single set of quantization parameters for all values within each activations array and within each weights array; separate arrays use separate quantization parameters, in that manner each layer can be quantized differently from other previous and later layers along the pile or stack of many layers from which our deep nets are normally constituted. We have to specify how are more precisely defined both S and Z *quantization parameters*. Starting from S, we can read from Benoit Jacob et.al(2018) paper that such parameter stands for "scale" and is an arbitrary positive

real number. It is typically represented in software as a floating-point quantity, like the real values r. While the latter that is Z, such constant represents "zero-point" and is of the same typeas quantized values q, and is in fact the quantized value q corresponding to the real value 0. This allows us to automatically meet the requirement that the real value $r = 0$ be exactly representable by a quantized value.

Benoit Jacob et.al(2018), while describing their quantization scheme they have provided for their quantizing experiments, have also shown and described the main reasons for training with simulated quantization and at the same time have briefly depicted the main reason that suggest why post quantization training represents sometimes a uneffective fine tune choices under certain conditions. So that, by their analyses, we should not adopt post-train quantization but rather to follow a quant-aware strategy if we want to exploit some kind quantization processing of the models we aim at compressing for reducing memory foot-print, and keep as much as possible performance untouched or nearly shallow models's original performance. In fact as they have found, just appling post train quantization to resulting trained models works sufficiently well for large models with considerable representational capacity, but leads to significant accuracy drops for small models. They have explained some common failure reasons for simply applying to our models post-training quantization, as large differences in ranges of weights for different output channels, or, *outlier weight values* that make all remaining weights less precise after quantization. Having saying that, we can now report the high level behavior of Benoit Jacob et.al(2018) quantization algorithm, which consists in simulating quantization effects in the forward pass of training. They, however, still made backpropagation to work as we are used to and all weights and biases are stored in floating point, but the forward propagation pass simulates quantized inference as it will happen at inference time, by implementing in floating-point arithmetic the rounding behavior of the quantization scheme. Finally, Benoit Jacob et.al stated that, weights are quantized before they are convolved with the input, furthermore, if batch normalization - as its use have been explored into [22] - is used for the layer, the batch normalization parameters are folded into the weights before quantization. However such additional features is not meaningful for our experiments since SIRENs do not use either convolution or batch normalization operations. Furthermore, if quantizing even net's activations concerns us for performance or constraints issue.
Mathematically speaking, Benoit Jacob et.al's quantization scheme can be formalized for each layer as follows. Quantization is parameterized by the number of quantization levels and clamping range, and is performed by applying pointwise the quantization function q defined as follows:

$$
\begin{aligned}
clamp(r; a, b) &:= min(max(x, a), b) \\
s(a, b, n) &:= \frac{b-a}{n-1} \\
q(r; a, b, n) &:= \left\lfloor \frac{clamp(r; a, b) - a}{s(a, b, n)} \right\rceil s(a, b, n) + a
\end{aligned}
\tag{3.8}
$$

where, we recall that r is a real-valued number to be quantized, instead $[a; b]$ is the quantization range, and n is the number of quantization levels, finally $\lfloor \cdot \rceil$ denotes operation corresponding to rounding to the nearest integer.

However, having made explicit the math formulae underling the quantization process for translating a full-precision real number to a reduce-precision whole value, Benoit Jacob et.al have also suggested how to correctly work weights and activations out, in fact, quantization ranges are treated differently for weight quantization vs. activation quantization. On one side, that is, for weights, the basic idea is simply to set $a := minw$,$b := maxw$. They apply a minor tweak to this so that the weights, once quantized as int8 values, only range in $[-127,127]$ and never take the value -128. On the other, that is, for activations, ranges depend on the inputs to the network. To estimate the ranges, them collect $[a; b]$ ranges seen on activations during training and then aggregatethem via exponential moving averages (EMA) with the smoothing parameter being close to 1 so that observed ranges are smoothed across thousands of training steps. This allows the network to enter a more stable state where activation quantization ranges do not exclude a significant fraction of values, however, for our experiments activation do not have been quantized or treated so that them can be represented in terms of lower precision values, and so, we left such further trick we may apply to compress to a wider degree the target AGP pruned models for future extension to this current initial analyses related to compressing siren based models for collecting statistics and performance showing how well such processed models behave with respect to original plain siren models when addressing the task of implicitly representing target input image.

Range Linear Quantization: Hyper-params Choice

Earlier for hyper-parameters choice when speaking about learning strategy pursued for training to prune selected baseline plain siren arch - represented from a plain siren example with fixed setting equal to $(n_{hf} = 64, n_{hl} = 5)$ which is a low depth but over-parameterized instance that lead to $(psnr(db) =\sim 49db, bpp =\sim 12)$. In particular, We then describe how was pruned by means of a specific compressing algorithm we have identified in Automated Gradual Pruning approach from Michael

Zhu et al(2018) paper. Also here, that is quantization step, where we are going to treat our already agp-pruned siren models via Range Linear Quantization - which is a quant-aware deep nets compressing technique. In fact, we have also to understand what would be best hyper-params setting for adequately learning quanted versions of our pruned siren instances to further reduce memory footprint as well as maintaining as much high as possible psnr score or at least not to ruin or impact heavily on psnr image quality measure.

Nevertheless, we have to follow different considerations and take into account distinct observation to guide the derivation and choice of suitable hyper-parameters to be tested quantized pruned siren architectures. We known, in fact, that when dealing with quantizing algorithm we cannot rely on some kind of preliminary analysis we can conduct as Sensitivity Pruning Tool, commonly applied for pruning based approaches for identifying the most suitable pruning rate for each involved layer within network structure. So, we should follow other heuristics or suggestions provided by literature that mainly leverage empirical experiments results obtained from other research works or studies to determine the right amount of quantizing level we desire to reach across several attempts we will carry out. However, as in pruning based methods, also for most quantizing approaches we have to establish the most adapt values for update frequency, that is, the amount of steps/epochs we have to wait and let pass before selected compression algorithm will take place to improve or update some statistics moving toward the goal of reducing processed models' memory footprint while preserving or attempting to slightly ruining those models' image performance quality measures. We have noticed some differences, however, when one comes to decide how to tune or pick frequency update value for allowing a kind of deep net compressing algorithm in different contexts. In fact, to be more concrete, we have learned that on a side, that is pruning method, we should adopt frequency update hyper-parameters values that are enough large to let pruning algorithm to made pruned nets while training is ongoing to recover from brain- or unit-damages which means that we attempt to let pruned models to recovery as much as possible through the intermediate reasonably abundant amount of epochs before net's architecture undergoes further neural units pruning. On the other side, that is quantization algorithms, instead we are suggested to sample our frequency update most suitable values from a low range of positive integers, in order to let quant procedure to follow as much closer as possible the learnable weights, biases as well as activation values distribution to be able to properly model the quant application that is in charge of mapping or matching floating-point parameters or units' output activations to some whole-precision n-bits levels that can maintain as much as possible unaltered final models' inference performance or as much as possible those output scores to original un-quanted models, before the latter have been trained by means of our selected quant-aware

training approach. That choice and suggestion was mainly due to the observations - that we will better explain, but that we briefly anticipate here - which deal with the fact that models quantized to a given bit-level for reduced-precision representation of their learnable parameters seem to better maintain original models performance otherwise too heavily corrupted when wider update frequencies span are preferred. Summarizing, we have selected update frequency values that resemble the following set of hyper-params values we have tested {2,5,10,25} with less attempts when we have tried 10, 25 update frequencies since yet from such values performance seem to degrade importantly.

Having suggested the strategy we follow for adopting or hypothesizing most eligible update frequency hyper-param values that are in charge to determine the amount of middle steps/epochs before quant-functions take place for updating internal quant-parameters necessary to allow quant scheme to correctly work, we have focused our attention to another relevant hyper-param that is learning rate. In fact, as we have previously already done when speaking about AGP pruning, also here we have to understand which is the relation of such hyper-param to other more closely related and proper quant-setting from which roughly speaking range linear quantization is characterized. Furthermore, while from the content we can read about AGP algorithm within its paper description we learned that pruning methods generally don't need important variations or fine-tuning about learning rate such that one can stick it to the value as it had reached when plain architecture was earlier trained, in the more general scenario where even a scheduling decaying learning rate was adopted together with optim-strategy to lead the procedure of learning parameters values, we instead have understood that when we further train deep nets archs via quantizing method we should follow a different way. In particular, it is conversely suggested to ulterior slightly reduce learning rate, and the main rationale underling such option is due to the fact that if we adopt a wider learning rate, quant-aware work-flow operate somewhat worst than adopting or preferring lower learning rate choices, which lead to broader performance spoils impacting dramatically as an instance to Psnr image quality measures. So, also for learning rate we can sum that the adopted learning rate we decided to tests end up to rely within the following subset of possible learning rate values, $1e-4$,$7.5-e5$,$5.5-e5$. So, as we can seen, we still stick with unchanged learning rate value that is $1e-4$ but further attempt to run some trainings via much lower learning rates that have nearly a lower order of magnitude in difference. We can briefly anticipate that we have collected better results when lower in magnitude learning rate was fixed together with a higher update frequency for training our already compressed agp-siren instances via quant-aware range linear quantization method.

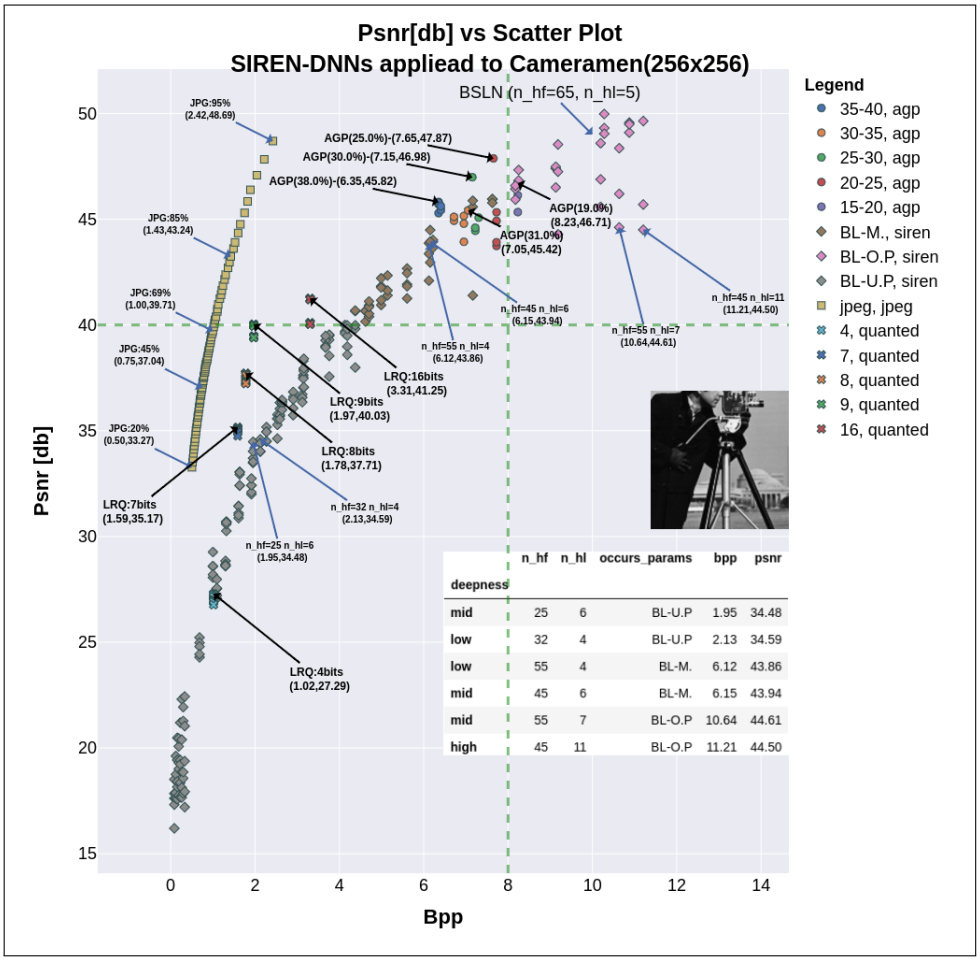
Once having finished with justifications and main motivation behind our choices - partly due to already studied research literature suggestions and heuristics provided for correctly guiding deep nets training - about such important and central hyper-parameters as update frequency and learning rate, where the former deals with more directly and closer than *learning rate* to quantizing algorithm settings, whereas the latter is a common and cross hyper-params that is involved into learning procedure generally adopted when fitting deep nets toward target cost functions to obtain resulting models able to correctly inferring and predicting output values, we made a further step toward discussing another major hyper-params we find ourselves to necessarily treat since differently from the other two already cited hyper-params, that one is much closer to quantizing algorithm essence, still roughly speaking. In fact, here we are going to tell how major choice behind bits-depth we have experimented with when attempting to quantizing our agp-pruned instance, for collecting and gathering results to be employed later for comparing gained statistics and performance scores against already trained plain, agp-pruned siren base architectures as well as jpeg cameramen compressed images we can create by running jpeg algorithm choosing diverse percentage qualities. We decided to test different options for bits depth, where such terms stands to genrally speking the number of level of bits or equivalently the integer precision desired, we can set for transforming, in other words, quantizing complete- or full-precision survived floating points already pruned via agp strategy siren based models' weights values which are the arch's parameters upon which we focus our strength and attention toward further reducing memory footprint while having still care about preserving or slightly spoiling performance image quality measures. In fact, as suggested in research literature, we firstly established which layers as well as parameters within each selected layer to quantize. In particular, due to original as well as pruned structure owned by siren models arrived at current phase of our experiments - that is, quantization stage - we opted to just reduce precision and so to represent in whole-precision way those parameters values that correspond to weights net's params, leaving untouched and intact biases vectors - considered not really important within the context of our primal analyses subject in future to further extensions and so left for future work's extensions - as well as activations for same reasons we have provided to biases vectors. Furthermore, we end up and complete the claim and hypotheses we supply for justifying our rationale underling selection of subset of networks parameters to be quantized, saying that we converge our strength and attention to merely handle weight parameters belonging to hidden stack of layers that siren architectures are commonly made from, so, avoiding quantizing both input layer, also referred to embedding layer from pixel's coordinate pairs to very first latent space further processed by remaining downstream network, as well as, output layer which instead by means plain fully connected layer is in charge of providing pixel level output estimate at inference time but earlier during

training stage even output signal employed as information to guide regression task compliant learning progresses. Such as choice allows us to exploit in a better way fully-precision latent representation will be provided from input embedding layer as well as floating-point precision when estimating output pixel magnitude given pairs of (x,y)-coordinates related to ordered grid of entries or cells that corresponds to pixel location that input image - we want to either learn, at train time, or infer, at inference subsequent time - is made from.
Summarizing, we can state that we have done several trials fixing a quantizing bit depth for rounding floating-points values down to integer-precision weight values choosing and one after other substituting bits representation depth in the set corresponding to {4,7,8,9,16}, that is, we have tested both very-low bits depth representation that allowed for more than $\sim 90\%$ memory requirements for handling siren models quanted down to 4-bits, as well as we have tried middle bits-depth such as those quanted architectures down to either 7, 8, or 9 where we accounted for $\sim 80\%$ of storage when dealing with those resulting networks, and finally to 16-bits integer-precision representation that allows for slightly more than just nearly 50% memory savage since we started quantizing to already slightly compressed siren networks since them have been already processed by means of a kind of pruning based algorithm known as AGP pruning approach.

Range Linear Quantization: Results

Having written all about hyper-parameters we have adopted for our downstream trials, we will discuss further, about how much well to be quanted, but yet pruned via AGP technique, siren-based deep networks, as well as we are ready for understanding what meaningful insights and knowledge we were able to find out having carried out several experiments throughout several distinct quantizing hyper-parameters setting we have applied to let training procedure to handle quantizing process, in other words, gathering statistics and learning quant-parameters for exploiting our networks to infer or predict desired output pixels magnitude as closer as possible to original unprocessed cameramen 256x256 image to be learned.

Looking at the Pnsr vs Bpp scatter plot figure 3.6 we have provided for analyzing the results we have found when we had carried out quantization training by means of Range Linear Quantization technique, we can state what follows. Starting our discussion from a quantizing level equals to 4-bits for representing floating point precision weights numbers as reduced precision integers, we can notice that we were able account for a bit rate reduction that is close enough to unity. This means that our siren models whose middle stack of hidden layers - from the layer right after embedding layer and up to the last but one layer - was forced to turn its weights representation to 4-bits depth or precision obtained a bit rate that is comparable



deepness	n_hf	n_hl	occurs_params	bpp	psnr
mid	25	6	BL-U.P	1.95	34.48
low	32	4	BL-U.P	2.13	34.59
low	55	4	BL-M.	6.12	43.86
mid	45	6	BL-M.	6.15	43.94
mid	55	7	BL-O.P	10.64	44.61
high	45	11	BL-O.P	11.21	44.50

Figure 3.6: Jpeg & Plain and Pruned Siren Networks: Scatter Plot Psnr[db] vs BPP.

with same bit rate we can achieve when applying jpeg compression algorithm assuring at least a quality factor equal to nearly 68%. However, speaking about psnr image quality measure, we can notice that we were not neither able to reach a jpeg compressed instance image through assuring 20% quality of result, thus, even if we can employ 4-bit quantization to reach more than nearly 90% percent of memory reduction when we compare those compressed siren models against our baseline reference corresponding to $(n_{hf} = 64, n_{hl} = 5)$ plain configuration, we lose more than $20db$ with respect to the original performance equal to 49.97db. So we conclude saying that such a critical quantizing level is too high for allowing trained models with such configuration to keep as much as possible unaltered the original psnr performance we got from training in a plain manner our selected baseline reference. However, comparing to other quantization levels we have take into

account in the remaining case we have to discuss, we can see lower variance when fixing quantizing n-bits but let learning rate or update frequency hyper-parameters to vary from trial to trial, demonstrating that when our siren architectures already pruned by means of AGP pruning method are even quanted via a very low degree of quantization other hyper-parameters affect slightly to the training procedure as well as to final psnr score. Instead if we are looking at the final results we have gathered when 7-bits or 8-bits depth are tried for achieving models' further memory compression to save storage capacity and at the same time keeping as higher as possible measurement performance, we notice that both lower precision mapping from floating point real numbers to integer corresponding digits, we end up with network examples that even if a somewhat low bit rate, nearly to 1.59 and 1.78 respectively, as so really close one another we still assist to psnr image quality scores that are not enough high as an achievable psnr score as we can when compressing input target image via jpeg assuming a 69% quality, though the difference is just about 3 up to 2 db, but sufficient to stating that by means those two distinct quantizing levels we reached a bit rate comparable with much more higher than a guaranteed 69 quality, but rather a jpeg compressing quality ranging from 85% up to nearly 90%, but with a dramatic psnr reduction from $\sim$ 20% up to $\sim$ 25% or even more. However, when comparing such quantized models against the nearest or closer in terms of number of parameters or bit rate plain siren architectures we can see that those lower precision networks outperform the full-precision counterpart siren models by an extent that ranges from few percentage points up to $\sim$ 10%, and but a lower rate that ranges instead from $\sim$ 15% up to $\sim$ 20%. We also notice that as we started quantizing from a already pruned model instance with a bit rate equal to 6.35 and a psnr score that achieves 45.82db we end up with quantized models with 7-bits and 8-bits weights integer presentation that account for a 70% reduction of memory footprint at the cost of 20% lower psnr values, much more similar to the results we have discussed when comparing scores we have for applying jpeg algorithm assuring 85% up to 95% quality but the pruned model we have quantized required nearly four or even more times memory. Finally, when put into relationship our original baseline plain siren trained instance to our instead quantized results from 7-bit and 8-bit quantization we can suggested that we have obtained results that seem much similar to those we can achieve when applying jpeg compression guarteeing at least 69% quality speaking about psnr score, that is, a reducition in magnitude that corresponds to minus $\sim 10db$ but we a $\sim$ 85% memory reduction. Differently from 4-bit quantization, we noticed that choices we have done for other hyper-parameters to be set before training via a quant-aware technique, which were learning rate and update frequency, have an important impact on the final gathered performance. In fact, we have observed that selecting lower learning rate while adopting a high update frequencies, models seem to learn better preserving as high as possible psnr score than otherwise preferring

larger learning rate as well as lower update frequecies. In other words, due to those differences in registered performance we have recorded for distinct hyper-params quantizing configs, we end up saying that such quantizing levels are more affected and show larger variance than trained instances we have found out when adopting 4-bit level of quantization. So, in such cases the proper selection or fine-tuning of other quant-parameters other than bith depth is meaningful and can impact in a sever manner on final models' measurament metrices.
Only arriving at a quantizing level equal to 9-bits we were able to obtain psnr values comparable with a jpeg resulting compression that assume 69% of saved quality, however, we need more than half of the data for representing our learned image with respect to the amount of information we should store when 69% jpeg quality is adopted for compressing our target picture. Furthermore, when picking 9-bits quantizing level up for representing weights parameters within the middle hidden stack of our already compressed siren architecture, via pruning technique as AGP pruning, we got instance that in the best cases are comparable in terms of bit rates to higher than compressed image instances we could gain when jpeg algorithm is exploited fixing quality to 90% or more, but with a $\sim 20\%$ lower psnr final score. So the quantized instances that we found out training our siren models, already compressed via AGP tech, were 5 times smaller than full-precision reference baseline from which all experiments started but we have been able to preserver up to $\sim 83\%$ of psnr score, in the best case that is adopting a somewhat low learning rate as low as $5.5e-5$ and a high update frequency that allows for learning to properly approximate pixel scores even if we reduced network capacity, optimizing the resources in terms of available learnable parameters. Moreover, among the tested n-bits quantization levels, 9-bit quantization degree was the only one that allowed for reaching the highest distance from both those plain siren architectures, that are those learned without any further compressing technique, comparable from bit rate standpoint as well as from psnr score viewpoint. In fact, in the former case we can notice that we were able to save nearly 16% of memory while at the same time we increased by nearly 6% the psnr score. Instead, in the latter case we were able to save two times more memory and still competitive to those plain siren models. Also for 9-bits quantizing hyper-parameter choice we have noticed that allowing other quant-parameters as well as learning rate to vary when sticking with such reduce-precision bit-width size resulting network models suffer from high variance, where lower learning rate and high update frequencies allow for better results than higher learning rate and lower update frequencies that instead get worst in terms of achievable measurement scores such as psnr value, demonstrating that also such n-bist depth is affected from distinct hyper-paraks configs we decided to adopt in different cases we will train.
Lastly, the final n-bit quantizing level we have tried was based on 16-bit floating-point to integer-precision mapping. It represented the biggest integer-representation

we have tried in order to accomplish and fulfill our goal we have beard in mind at that precise stage within our experiments setting, that is, quantizing our already pruned siren based architecture via AGP pruning technique, in order to understand whether we such low depth n-bit quantization we can really take advantage from such low deepness transformation that might corresponds to a negligible reduction in quanted models' performance in terms of psnr image quality measurements, as well as, a somewhat acceptable memory footprint reduction when comparing those results we have obtained with such quantizing configuration against both jpeg's psnr performance cases, pruned and plain siren deep nets examples psnr score measures. However the fact that our experiments we have carried out when 9-bits quantization was fixed lead to both highest enhancement over plain siren architectures of same size, as well as highest enhancement over plain siren architectures sharing similar psnr score but we advantage that 9-bits quantized models occupy less memory when integer representation is involved, we still can appreciate somehow a kind of measurable and impotant improvement when also 16-bit whole-precision is adopted when comparing such quantized models against plain siren models with both a comparable size or with a comparable psnr score, where, in the latter case we can say in advance that 16-bit models required less memory with respect to comparable models in terms of psnr score. More specifically, when exploiting 16-bit integer representation, we noticed that in the best case, our results lead to examples characterized from a bit rate of ~ 3.31, at the cost of a psnr score equals to $\sim 41.25db$, which corresponds to a nearly 17% reduction of image quality with respect to our baseline siren architecture that was firstly compressed via AGP pruning technique and then quanted, but saving nearly 67.80% of memory usage, from a bit rate of $10.28bpp$ down toward slightly over $3bpp$. While comparing such results against compressed siren instance from which quantizing experiments have been carried out, we notice that halving the memory requirement, we reduced image quality about just $\sim 10\%$ which can be considered as a great advantage since with an important memory reduction we do not assisted to a dramatically image quality score decreasing. So we can consider ourselves satisfied from both results we have reached when quantization was invoked adopting either 8-, 9-, 16-bit depth quantization levels, even if 8-bit quantizing depth was not so satisfying as it were the latter two. However, when 16-bit depth level was considered, we should have said that such quant-parameter in the context also of the particular pruning degree employed earlier for compressing our initial plain siren architecture, we obtained results that in terms of bit rate were still higher than the maximum jpeg quality we have reported, corresponding to 95% of ensured image quality result with a nearly 49db psnr score, while in terms of guaranteed image quality measurements our results seems to be remarkably higher than a jpeg compressed image obtained fixing a 69% quality, by rather a resulting compressed image of ranging more or less from 75% to 78% quality. Finally, we can end up saying that when we focused on an

instance of pruned siren architecture which was compressed up to $\sim 38\%$ of plian siren original size and then quantized passing through several distinct attempts between which a number of diverse n-bits quantizing levels have been substituted for quantizing and so representing floating-point full precision hidden weights into integer numbers for whole-precision, pursuing the target of being able to further compressing deep net siren-based architecture for reducing memory footprint but at the same time attempting to keep image quality performance measures as high as possible, we have obtained remarkable results that suggest us what follows. We were able to quantizing an existing already partly compressed via pruning technique siren architecture finding some examples thanks to which we can state that it is possible to take advantage of both pruning and quantizing compression techniques and steps to reduce memory footprint or requirements of compressed, quantized models but still with final trained models that were in the best cases better than plain siren architectures of comparable size as well as lead to a interesting memory compression even when those quantized models have compared against plain siren architectures that instead were similar in terms of psnr image quality measure score.

3.2.6 Generalization to other test images

Once we have done with our results we have precisely drawn from training several distinct SIREN-based models applying first AGP-pruning technique and then followed by LRQ-quantization algorithm where we stick on cropped by its center Cameramen input image, we attempt to understand whether our pruning and quantization choices can generalize to different test images. Particularly, we moved on another image which is the one coming from the so called BSD68 dataset as we can find cited in here[28] which has been This widely used for measuring image denoising algorithms performance, however I could not find it easily. It includes the original .jpg files, converted to lossless .png, and noisy with Additive White Gaussian Noise of different levels. However, we focused our attention on grayscale image to be aligned with our earlier analyses done by fixing as input target image cameramen picture. Moreover, we even crop, where it was possible, the second picked image by its center down to 256 by 256 pixels. In particular, we adopted *test066.png* grayscale picture, which is the image 3.2.6:

As done before when cameramen 512x512 image was cropped by its center down to 256x256 image, also for *test066.png* input target image we have cropped it down by its center, that corresponded passing from full-image 481x321 to 256x256 cropped image, which results into the subsequent image **??**:

Figure 3.7: Test066.png image from BSD68 dataset.

Figure 3.8: Cropped Test066 256x256 target image

Image Feature	Value
name	Test066
shape	(256, 256)
size_byte	65536
image_band	(L,)

Table 3.9: Cropped Image Test066 Main characteristics

Currently, having cropped test066 image as illustrated above does not changed its own pixel values distribution shape in an appreciable manner, as shown in the following picture where original and cropped test066 image version pixel values distribution have been reported:

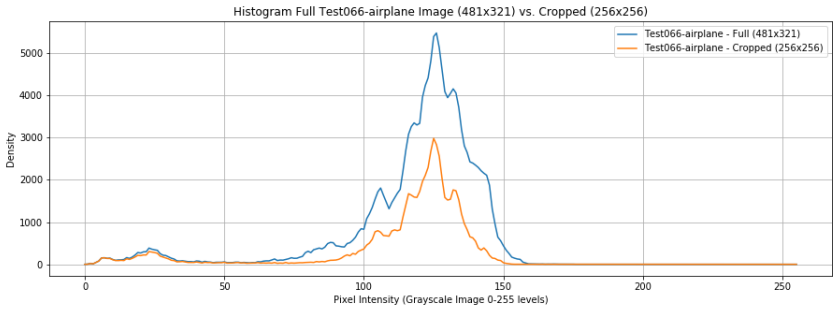
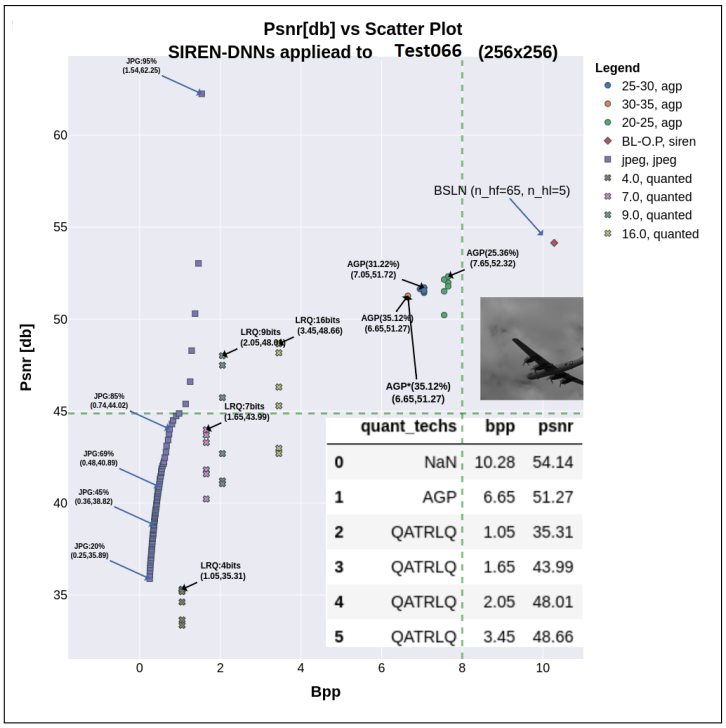


Figure 3.9: Pixel Values Distributions for both Original and Centered Cropped Test066 image

In fact, we still can see how the most of values about cropped image 3.9 are laying between 100 to 150 pixel magnitude values, as in the original test066 picture , and even pixel values corresponding to very low magnitude seems somewhat untouched from cropping preprocessing phase. So, we still kept most of the data from low intensity pixels which corresponds somehow to airplane shape, as one might think out. In other words, even if our pre-processed image have been cropped we can state that even if the actual learnt image would be the full reference, we should face safe issues when attempting to learnt weight parameters necessary to represent the airplane related form.

For the different trials we have done sticking with test066.png image we had followed same kind of tests and we have adopted more or less similar hyper-parameters configuration values, in order to understand and see how diverse the performance statistics we may collect from those successive attempts are from the ones we have observed from processing cameramen image. The computed output

results are reported below within the Figure 10:



	quant_techs	bpp	psnr
0	NaN	10.28	54.14
1	AGP	6.65	51.27
2	QATRLQ	1.05	35.31
3	QATRLQ	1.65	43.99
4	QATRLQ	2.05	48.01
5	QATRLQ	3.45	48.66

Figure 3.10: Cropped Test066 256x256 target image

As we can observe from the graphics shown just above in 3.10, we decided to select a baseline model represented from a hyper-parameter configuration equals to $(n_{hf} = 65, n_{hl} = 5)$ where leads to a similar configuration, in terms of number of weights as the one picked up for cameramen image. However, the major difference among the two initial configurations, even if both correspond to over-parameterized instances, is in terms of achieved performance. In fact, the baseline model corresponding to test066 learnt image had gained an higher psnr score rate equals, greater than 50 db. But, if we look at the difference with highest Jpeg quality for the same picture we can notice that the gap is significant due to the fact that the difference is nearly ten folds as 54.14, reminding at the same time that while the 95% guaranteed quality for Jpeg compressed image leaded to nearly a bit rate lower than 2 bpp, adopting a plain siren architecture as the one shown for our baseline instead caused a bit rate much grater as 10.28, overcoming at the same time untouched original image bit rate, which instead corresponds to 8 bpp. However, this time for the subsequent pass corresponding to applying pruning technique to our baseline model, we established to focus on less attempts, just the most promising, which instead lead to restricting interesting prune rate intervals to just three, corresponding to 20-25, 25-30, 30-35. This choice has been also

motivated by the observation that other intervals do not guarantee any kind of interesting gain in terms of performance and Psnr over Bpp score ratio, as also already observed for previously processed cameramen image.
As can be noticed looking at the graphics, we can observe that all the tests we have done varying pruning rate seem to lie within the same range of values related to Psnr score, while guaranteeing that we have been able to reduce bit rate so that resulting model instances obtained a bit rate lower than that corresponding to the original cropped test066 image. As also noticed for cameramen image, even for test066 image as the pruning rate increases we were able to reduce variance among different tested hyper-parameter configurations, confirming that as we increase weights pruning varying update frequency rate we approximate somewhat similar pruned models that are able to still represent input image with minor difference between each others. So, improving pruning rate as encouraging more aggressive weights filtering we end up with models that behave more or less in the same way with similar performance in terms of preserved Psnr score values. For improving result analysis, we have filled the following summary table with most important results among the tests we have performed for measuring how well automated gradual pruning has gone when test066 was the input data set.

Image Test066	(n_{hf}, n_{hl})	**size(byte)**	**psnr(db)**	**bpp**
(baseline model)	(64, 5)	84228	54.14	10
agp (prune intervals)	**prune_rate(%)**	**size(byte)**	**psnr(db)**	**bpp**
20-25	25.0	62688	52.32	7.65
25-30	30.0	58592	51.64	6.95
30-35	35.0	57780	51.27	6.65

Table 3.10: Best Siren AGP pruned cases: center cropped 256x256 test066 image

As one can notice in table 3.10, pruned models caused the resulting image to get some noise when attempting to estimate its pixel intensity values, in fact on average we have lost nearly 5% db units relative to Psnr performance quality measure, but we have been able to reduce memory footprint nearly to 30% with respect the original plain siren architecture. This suggest that even if we decided to restrict pruning rate to be quite not aggressive, we have shown that we can still get rid of a portion of less salient subset of weights for the task at hand of learning weights for image compression that have lead to pruned SIREN architectures still able to represent input image in a suitable manner.

For the subsequent final stage, corresponding to weights quantization we decided to fix our choice among the prune test066 architectures to an instance corresponding to an example which have been pruned within 25-30 pruning sub-interval. More precisely we picked up the second entry among those reported just above within such table where we have recorded results we have obtained from pruning phase. So, we decided to quantize that pruned model which gained ($Pnsr = 51.64db, Bpp = 6.65$) at a 30% level of pruning rate. As we have already done before for cameramen quantization analysis, we decided to test the following set of quantization levels, 4, 7, 6, 8, 9, and 16. Such quantization levels have been set just for hidden layers weights parameters, avoiding to quantize both very first hidden layer as well as output layer and finally we have not quanted biases vectors. As done for camermen image we followed similar hyper-parameter settings, where in particular encouraged lower learning rate and higher frequency update so that quantization technique at hand was able to get over learning issues improving notably image quality performance represented by Psnr score with respect those other quant-configurations that taken into account higher learning rate or higher frequency updated, where we recall that frequency updated correspond to how long takes before adotped quantization technique takes place for updating quant parameters over different epochs through which fine-tuning process came across. As for pruning cases, also for quanted examples obtained from adopting test066 as data set for accomplishing fine-tuning phase by means of quantization technique, we have reported best entries within the following summary table:

Image - Test066	(n_{hf}, n_{hl})	**psnr(db)**	**bpp**
baselines	(64, 5)	54.14	10.28
quant models bith depth	(n_{hf}, n_{hl})	**psnr(db)**	**bpp**
4	(64, 5)	20.99	1.09
7	(64, 5)	34.09	1.72
8	(64, 5)	45.88	1.93
9	(64, 5)	48.05	2.14
16	(64, 5)	49.27	3.60

Table 3.11: Best Siren Quanted Cases: center cropped 256x256 test066 image

Looking at the results in table 3.11 we can appreciate from the above summary table we can notice what follows. Adopting very low bit depth as 4 bits in order to represent reduced precision weights we end up with resulting tuned models that lost

more than half of Pnsr score value on average when compared a Jpeg compression image with a quality of 85% which shares with those poor models a similar bit rate. In other words a very hard quantization level applied on weights affects widely image quality performance so that not enough precision for representing our 4-bit quanted models's weights loose totally the advantage we gotten early when pruning plain reference baseline model. The difference between such low bit resolution models and baseline model is even greater attaining nearly 34(db) loss in Psnr score, which means that without enough bit-width per weight for representing such reduced precision weights the networks are not able to correctly predict from pixel coordinates the corresponding magnitude level. Different situation we met when decide to select a 7-bit width for our hidden already pruned models' weights. In fact we have gained 10 more db units when comparing such models over 4-bit smaller architectures, but we still face performance issue, in fact we require from these models a bit rat greater than the number of pixels we end up when original image is compressed via Jpeg guaranteeing 95% quality of the final compression result, while the image quality score looks like to Jpeg compressed image cases where we adopted a quality score as equal as 20% which is normally not employed since leads to unacceptable compressed images since with a bit rate as lower as 25% those jpeg compressed images have not enough data per pixel to guarantee a recognizable compressed image when aligned with original image.
When looking at results we gotten observing trained models via 8-, 9-bits quantization we have can state that the difference among these two close levels are grater than the difference we have recorded from cameramen image in terms of Psnr db units. In other words, applying 8-bit quantization for turning hidden parameters to reduced precision integer-weights we have seen a wider reduction from 9-bit quantized models. However, training already pruned SIREN architecture setting a 8-bit quantization level when test066 has been selected as our data set, we were able to cross that horizontal line corresponding to Psnr score of nearly 44 db, when we compressed via Jpeg original image so that the resulting compression version has a unit bit rate. This observation is necessary to understand that even if we have obtained Psnr score for 8-,9-bit quantization that are similar to Jpeg compressed images with a guaranteed quality greater than 85% when still go worst than Jpeg compressed images at 90-95% of guaranteed quality in terms of bit rate, since we are still further, with on average 28% more bit per pixels.
Arrived at 16-bit depth quantization setting, we have noticed that instead the difference with 9-bit quant level is lower than the same difference if calculated to the same kind of quantization levels when looking back to cameramen data quantization results. However, when 16-bit level is fixed as hyper-parameter for performing models' quantization phase, we observed as comparing compression results from pruned instances with such 16-bt quanted models we have noticed that existing gap is much lower while still halving the number of bits or the amount of

data per pixels. This means that, adopting quantization after just having pruned baseline models even just halving the information to include for representing as reduced precision hidden layers we do not waste or impact critically on Psnr score of resulting quanted models's adopting 16-bit quantization.
Also the results we have collected when another image, different from the main cameramen image that was adopted as reference image for performing exhaustive test analyses, such as test066 we end up to similar performance in terms of difference among the intermediate results coming from the different phases we have come across, starting from baseline training, then followed firstly from AGP pruning and after by quantization, where quantization technique was fundamental to rapidly decreasing significantly bit rate, even if that widen reduction with respect pruning phase, arrived at the cost of lower image quality scores that imply more noise is introduced when not enough reduced precision depth is guaranteed for representing as integer number hidden layers' weight parameters.

Chapter 4

Conclusions

4.1 Conclusions

4.1.1 Summarizing Results

The major remarks we have found while appling Automated Gradual Pruning (AGP) have been the following. Models seem to need a reasonable extensive number of epochs to improve over performance reduction, when weights are pruned. However, we were able to outperform baseline architectures, that are plain Siren-based Nets as well as satisfy constraints of being at least lower in size with respect to Learned Image as well as still performing better than some comparing in size plain Siren-nets. Laslty, compared to Jpeg compression results we still notice that the existing gap between AGP-pruning bit rate and jpeg instances rate.
On the other and, the main results we have found while appling Quantization-Aware Training Procedure, which is Linear Range quantization-aware training were the subsequent. Firstly, by means of a wide reduction in bit rate score we noticed that such choice corresponded a drammaticaly reduction in Pnsr image quality score, high distortion (as for 4-bit representation). Nevertheless, we have been able to train quantized model that show an acceptable Distorsion level with respect to both Jpeg compressed counterpart, and overcome with a large extent plain Siren-based models (as for 9-, 16-bit representations). Moreover, Learning Rate hyper-param was crucial for identifying promising Linear Range Quant Configurations, and lower Learning Rate values seem to outperform larger ones. We end up saying that, Learning Rate hyper-param was crucial for identifying promising Linear Range Quant Configurations, and lower Learning Rate values seem to outperform larger ones. Finally, even by means of a combination of both a Pruning Approach and next a quantization method, we had still faced the problem of having models that did not approached significantly closely to results we can instead reach by simply appling well-estabished compression techniques for image compression such as

Jpeg. But we know that diving deep in the sudy of more sophisticated techique for accomplishing neural networks compression or looking for more suitable hyper-parameters configurations there exist room for potential improvements that have not yet been found, suggesting how the problem of attempting to compress images by means of deep nets such as SIREN is still an open and interesting field.

4.1.2 Interesting further Deep Neural Network related Compression Techniques

As we already mentioned, NN Pruning Techniques can reduce the parameter counts of training networks, decreasing storage requirements and improving computational performance of inference, without compromising accuracy. Though, that was not really our case, when attempting to pruning *SIREN* based architectures, we still want to suggest other interesting pruning based methods that can be employed to further explore *SIREN* behaviors and learning capabilities when exposed to such deep nets class of compressing algorithms. As an instance we recall the so called *Lottery Ticket Hypothesis (LTH)* pruning approach studied and proposed by Jonathn Frankle et al. (March2019)[9], where the main reason for using it that them illustrate are as follows. In fact, they found out standard pruning techniques uncover trainable subnetworks, i.e. winning tickets, from FCs and CNNs. So, they understood that winning tickets, i.e. best initial params intialization, lead to connections having initial weigths that make training particularly effective. Throughout such suggested hypotheses Jonathn Frankle et al. show in their paper that there consistently exist smaller subnetworks that when trained from the start are able to learn at least as fast as their larger counterpart while reaching similar test accuracy. While, other interesting LTH characteristics are represented by the fact that It supports both one-shot and iterative-pruning behaviour, making it suitable in a variety of context when hardware and time constraints arise. Furthermore, LTH follows unstructured-pruning heuristics, focusing on reaching sparisiy. So, we can end up saying that by means of Lottery Ticket Hypothesis pruning method we can provided another iterative based pruning algorithm that however follows another pruning workflow that makes such technique a interesting alternative to be tested in order to later compare performance it let gain by *SIREN* based architecture we are going to prune over AGP pruned models to understand which might be the most suitable to be exploited for fulfilling our goal, that is being able to fit a subset of suitable, salient learnable models' parameters that will make up the pruned architecture that implicitly should represent the target image we aim at representing throughout a deep neural network architecture.

Having shown a potential candidate among pruning based deep neural network techniques for extending our initial thesis work for also including more results

coming from distinct pruning centered approaches so that we can further compare and cross checking the results we would record by testing different hypotheses, pruning-parameters configurations or settings, we also propose and suggest other interesting quantization technique that we might desire to explore and investigate for improving our baseline analyses we have performed by means of current thesis scenario. In other words, we do not want to just fix or stick to a unique and precise quantization-methods such as the one that we have explored somehow and that we even want to extend by attempting to further test other missed possible quant-parameters settings, as well as want collect and gather data results that instead we should obtain by running other quant-based algorithms for as earlier said when speaking about pruning method, to enlarge our beginning report and analysis performed around compressing *SIREN*-based deep-net architectures. As an examples of another distinct and interesting quantization algorithm that we may suggest and purpose as a valid alternative to the earlier quant-method that instead we have employed along with the experiments we have performed for current thesis context, we indicate the work done by Xiandong Zhao et al. (2020) [47], *Linear Symmetric Quantization of Neural Networks for Low-precision Integer Hardware* where them introduced a *learned linear symmetric quantization (LLSQ)* to quantize the whole network including the bias and scaling factors. Among the main reasons to trying employing such quantization method might be the observation that Xiandong Zhao et al. have done since they have opinion that even linear symmetric quantizer can be trained to outperform asymmetric or non-linear quantization when also a low quantization depth is employed. Thus, it might be curios to attempt investigating whether we may able to fit low precision counterpart *SIREN*-like architectures as lower as 4-bit to take advantage of such quantization depth in order to understand if we can overcome jpeg related image performance quality scores when fitting deep models with such quant configuration.

Furthermore, we propose to extend our proposed future works, to improve baseline current analyses we have finished throughout our thesis study, suggesting not only to focus mainly upon on investigating and searching for other potential deep-nets compression compliant techniques or algorithms to still fit and learn to the data we provided at training time for feeding models' from their input layers all the way down toward output layer for let both forward and backward procedures to act in order to learn suitable set of net's parameters for the task we aim at satisfying. But even want to suggest another interesting way of thinking about lossy compression algorithm normally employed for reaching less expensive deep-archs in terms of memory consumption both at inference time as well as when we store and memorize such compressed deep networks. In other words, provide interesting evidence about a study lead by Yochai Blau et al. (2019) [2] where the main focus leverage around how we can rethinking about lossy compression

deep learning based procedure when we have to take into account quantization transformations as it is the case in our precise investigation process. In fact, as Yochai Blau et al. (2019) reported, lossy compression algorithms are typically de-signed and analyzed through the lens of Shannon's rate-distortion theory, where the goal is to achieve the lowest possible distortion at any given bit rate. This was, precisely, the workflow we mainly have followed when attempting to carry out our experiments. Moreover, Yochai Blau et al. (2019) started their reasoning from the comparing we can have between *low distortion* and *high perceptual quality*, where they observed that many times optimization of one often comes at the expense of the other. In light of this understanding, it is natural toseek for a generalization of rate-distortion theory which takes *perceptual quality* into account. Furthermore, within their paper, Yochai Blau et al. (2019) adopt the mathematical definition of perceptual quality recently proposed by Blau Michaeli (2018) [3], and use it to study the *three-way tradeoff between rate, distortion, and perception*. Finally, they show that restricting the perceptual quality to be high, generally leads to an elevation of the rate-distortion curve, thus necessitating a sacrifice in either rate or distortion. Then, they prove several fundamental properties of this triple-trade off, calculate it in closed form for a Bernoulli source, and illustrate it visually on a toy MNIST example. So, it might be interesting to understand as their work could be adapted and integrated within training workflow applied instead to SIREN based models when we are focused onto fit such models to input data image that we aim at implicitly representing via the architecture final configuration itself.
Even if such approach is somewhat almost recent, we can still take into account already existing solutions to tackle the quantizing problems, attempting to implement by ourselves or either adopting existing code, if any, to be incorporated into our initial training scheme we have derived for such starting phase. More precisely the solution that might result interesting can be the following:

- The **Deep Compression** by Han et al. (2015) [16], which uses clusters to categorize weights into groups of quantized values. More precisely, Deep Compression stands for a three stage pipeline:pruning, trained quantization and Huffman coding, that work together to reduce the storage requirement of neural networks.

- The DoReFa-Net by Zhou et al. (2016) [48], in which quantization is achieved through quantizing weights, activation, and gradients with different width of bits via absolute values of full-precision weights as layer scaling factor.

- Finally, the Trained Ternary Quantization by Chenzhou Zhu et al. (2017) [49], improvement over TWN by Li & Liu(2016) [26]. In fact, it resembles Aggressive quantization that enabling at the same time for even a bit of layers sparsity, embedding such feature directly in the designed quantization scheme.

Lastly, we propose to extend our initial work based on MSE default constructed target function employed for carrying out training of our baseline, pruned, as well as quanted model instances by incorporating another constraining terms to the overall loss function to take into account perception. Such idea arose from previous research works by Yochai Blau et al. (2019), in The Rate-Distortion-Perception Tradeoff [2], where the authors seek to achieve a minimizing distortion level so that decoded signal still have good perceptual quality, exploring the effect of the balance between bit rate, distortion and perception in **image (input signal) restoration**, and and, to achieve their goal they defined a ***perceptual quality index*** embedded within loss target definition, as a regularization term, together with λ, tuning parameter to control the balance between perception and distortion scenario.

4.1.3 Further Acknowledgments

I'm going to acknowledge for the support offered to me, during the whole period of time that required for carry on experiments and suggesting which choices to follow in order to test several different ideas linked to the purposes of leading the analyses of different compressing solutions and setting Diego Valsesia, which, so, provided me again suggestions and held helpful discussions for letting me overcome troubles when they sometimes have faced. Lastly, I'm going to acknowledge my parents and my friends for supporting me, the former economically and morally, the latter especially offering kind words when fatigue sometimes arose.

Bibliography

[1] Matan Atzmon and Yaron Lipman. «SAL: Sign Agnostic Learning of Shapes from Raw Data». In: *CoRR* abs/1911.10414 (2019). arXiv: 1911.10414. URL: http://arxiv.org/abs/1911.10414.

[2] Yochai Blau and Tomer Michaeli. «Rethinking Lossy Compression: The Rate-Distortion-Perception Tradeoff». In: *CoRR* abs/1901.07821 (2019). arXiv: 1901.07821. URL: http://arxiv.org/abs/1901.07821.

[3] Yochai Blau and Tomer Michaeli. «The Perception-Distortion Tradeoff». In: *CoRR* abs/1711.06077 (2017). arXiv: 1711.06077. URL: http://arxiv.org/abs/1711.06077.

[4] D. Brunet, E. R. Vrscay, and Z. Wang. «On the Mathematical Properties of the Structural Similarity Index». In: *IEEE Transactions on Image Processing* 21.4 (2012), pp. 1488–1499. DOI: 10.1109/TIP.2011.2173206.

[5] Rohan Chabra et al. «Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction». In: (2020). arXiv: 2003.10983 [cs.CV].

[6] Zhiqin Chen and Hao Zhang. «Learning Implicit Fields for Generative Shape Modeling». In: *CoRR* abs/1812.02822 (2018). arXiv: 1812.02822. URL: http://arxiv.org/abs/1812.02822.

[7] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. «BinaryConnect: Training Deep Neural Networks with binary weights during propagations». In: (2016). arXiv: 1511.00363 [cs.LG].

[8] J. Erfurt et al. «A Study of the Perceptually Weighted Peak Signal-To-Noise Ratio (WPSNR) for Image Compression». In: (2019), pp. 2339–2343. DOI: 10.1109/ICIP.2019.8803307.

[9] Jonathan Frankle and Michael Carbin. «The Lottery Ticket Hypothesis: Training Pruned Neural Networks». In: *CoRR* abs/1803.03635 (2018). arXiv: 1803.03635. URL: http://arxiv.org/abs/1803.03635.

[10] Y. Gao, A. Rehman, and Z. Wang. «CW-SSIM based image classification». In: (2011), pp. 1249–1252. DOI: 10.1109/ICIP.2011.6115659.

[11] Kyle Genova et al. «Deep Structured Implicit Functions». In: *CoRR* abs/1912.06126 (2019). arXiv: 1912.06126. URL: http://arxiv.org/abs/1912.06126.

[12] Kyle Genova et al. «Learning Shape Templates with Structured Implicit Functions». In: *CoRR* abs/1904.06447 (2019). arXiv: 1904.06447. URL: http://arxiv.org/abs/1904.06447.

[13] Xavier Glorot and Yoshua Bengio. «Understanding the difficulty of training deep feedforward neural networks». In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

[14] Yunchao Gong et al. «Compressing Deep Convolutional Networks using Vector Quantization». In: (2014). arXiv: 1412.6115 [cs.CV].

[15] Amos Gropp et al. «Implicit Geometric Regularization for Learning Shapes». In: (2020). arXiv: 2002.10099 [cs.LG].

[16] Song Han, Huizi Mao, and William J. Dally. «Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding». In: (2015). cite arxiv:1510.00149Comment: Published as a conference paper at ICLR 2016 (oral). URL: http://arxiv.org/abs/1510.00149.

[17] Song Han, Huizi Mao, and William J. Dally. «Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding». In: (2016). arXiv: 1510.00149 [cs.CV].

[18] Song Han et al. «Learning both Weights and Connections for Efficient Neural Networks». In: (2015). arXiv: 1506.02626 [cs.NE].

[19] B. Hassibi, D. Stork, and G. Wolff. «Optimal Brain Surgeon and general network pruning». In: *IEEE International Conference on Neural Networks* (1993), 293–299 vol.1.

[20] A. Horé and D. Ziou. «Image Quality Metrics: PSNR vs. SSIM». In: (2010), pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[21] Itay Hubara et al. «Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations». In: (2016). arXiv: 1609.07061 [cs.NE].

[22] Sergey Ioffe and Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[23] Benoit Jacob et al. «Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference». In: (2018).

[24] Diederik P. Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: (2017). arXiv: 1412.6980 [cs.LG].

[25] Yann Lecun et al. «Optimal brain damage». English (US). In: 2 (1990). Ed. by David Touretzky.

[26] Fengfu Li and Bin Liu. «Ternary Weight Networks». In: *CoRR* abs/1605.04711 (2016). arXiv: 1605.04711. URL: http://arxiv.org/abs/1605.04711.

[27] Zhouhan Lin et al. «Neural Networks with Few Multiplications». In: (2016). arXiv: 1510.03009 [cs.LG].

[28] D. Martin et al. «A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics». In: 2 (2001), pp. 416–423.

[29] Lars M. Mescheder et al. «Occupancy Networks: Learning 3D Reconstruction in Function Space». In: *CoRR* abs/1812.03828 (2018). arXiv: 1812.03828. URL: http://arxiv.org/abs/1812.03828.

[30] Mateusz Michalkiewicz et al. «Implicit Surface Representations As Layers in Neural Networks». In: (2019).

[31] Ben Mildenhall et al. «NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis». In: (2020). arXiv: 2003.08934 [cs.CV].

[32] Sharan Narang et al. «Exploring Sparsity in Recurrent Neural Networks». In: (2017). arXiv: 1704.05119 [cs.LG].

[33] Jim Nilsson and Tomas Akenine-Möller. «Understanding SSIM». In: (2020). arXiv: 2006.13846 [eess.IV].

[34] Jeong Joon Park et al. «DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation». In: *CoRR* abs/1901.05103 (2019). arXiv: 1901.05103. URL: http://arxiv.org/abs/1901.05103.

[35] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: (2019). arXiv: 1912.01703 [cs.LG].

[36] Songyou Peng et al. «Convolutional Occupancy Networks». In: (2020). arXiv: 2003.04618 [cs.CV].

[37] Mohammad Rastegari et al. «XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks». In: (2016). arXiv: 1603.05279 [cs.CV].

[38] Abigail See, Minh-Thang Luong, and Christopher D. Manning. «Compression of Neural Machine Translation Models via Pruning». In: (2016). arXiv: 1606.09274 [cs.AI].

[39] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. «Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations». In: *CoRR* abs/1906.01618 (2019). arXiv: 1906.01618. URL: http://arxiv.org/abs/1906.01618.

[40] Vincent Sitzmann et al. «Implicit Neural Representations with Periodic Activation Functions». In: (2020). arXiv: 2006.09661 [cs.CV].

[41] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. «Improving the speed of neural networks on CPUs». In: (2011).

[42] Z. Wang and A. C. Bovik. «Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures». In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117. DOI: 10.1109/MSP.2008.930649.

[43] Z. Wang, E. P. Simoncelli, and A. C. Bovik. «Multiscale structural similarity for image quality assessment». In: 2 (2003), 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.

[44] Zhou Wang et al. «Image Quality Assessment: From Error Visibility to Structural Similarity». In: *Trans. Img. Proc.* 13.4 (Apr. 2004), 600–612. ISSN: 1057-7149. DOI: 10.1109/TIP.2003.819861. URL: https://doi.org/10.1109/TIP.2003.819861.

[45] B. Widrow and I. Kollár. «Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications». In: (2008).

[46] Yonghui Wu et al. «Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation». In: (2016). arXiv: 1609.08144 [cs.CL].

[47] Xiandong Zhao et al. «Linear Symmetric Quantization of Neural Networks for Low-precision Integer Hardware». In: (2020). URL: https://openreview.net/forum?id=H1lBj2VFPS.

[48] Shuchang Zhou et al. «DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients». In: *CoRR* abs/1606.06160 (2016). arXiv: 1606.06160. URL: http://arxiv.org/abs/1606.06160.

[49] Chenzhuo Zhu et al. «Trained Ternary Quantization». In: *CoRR* abs/1612.01064 (2016). arXiv: 1612.01064. URL: http://arxiv.org/abs/1612.01064.

[50] Michael Zhu and Suyog Gupta. «To prune, or not to prune: exploring the efficacy of pruning for model compression». In: (2017). arXiv: 1710.01878 [stat.ML].

[51] Neta Zmora et al. «Neural Network Distiller: A Python Package For DNN Compression Research». In: (2019). arXiv: 1910.12232 [cs.LG].