Deep Neural Networks' Compression Techniques Study

Appling compressing methods on Siren based Deep Neural Networks for comparison against other compressing techniques in the field of Computer Vision

Chiarlo Francesco Maria

University Polytechnic of Turin

February 3, 2021

POLITECNICO
DI TORINO

Outline

Siren Deep Neural Network Architectures

Main reasons that fluctuate around development and employment of Siren Deep Neural Network Models, as reported by related research work describe in the corresponding publication:

- Implicitly defined, continuous, differentiable signal representations parameterized by neural networks have emerged as a powerful paradigm, offering many possible benefits over conventional representations.
- However, current network architectures for such implicit neural representations are incapable of modeling signals with fine detail, and fail to represent a signal's spatial and temporal derivatives, despite the fact that these are essential to many physical signals defined implicitly as the solution to partial differential equations.

Siren Deep Neural Network Architectures (2)

Main reasons that fluctuate around development and employment of Siren Deep Neural Network Models, as reported by related research work describe in the corresponding publication:

- Proposed to leverage periodic activation functions for implicit neural representations and demonstrating that these networks, dubbed sinusoidal representation networks or *SIRENs*, are ideally suited for repre-senting complex natural signals and their derivatives.
- Motivated by the discrete cosinetransform, Klocek et al. leverage cosine activation functions for image representation but they do not study the derivatives of these representations or other applications explored in Siren Models.

Siren Based Network Topolgy

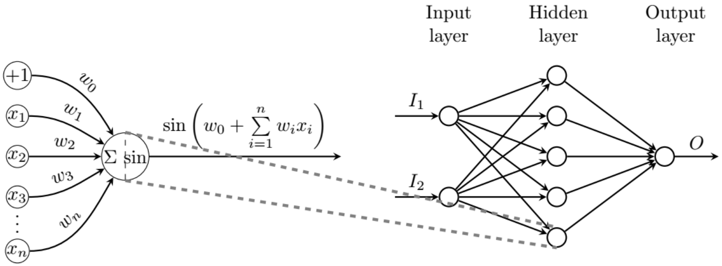


Figure: Siren Architecture Example

Example of tiny Siren like architecture, in order to show the minimal configuration that can be employed to describe network's major behavior.

Principal Siren's Paper Contributions

The major contributes we can read of from Siren's research paper that have been produced to justify architecture itself existence are:

- 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 distributionsof 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 bysupervising only with its gradients; and solving second-order differential equations.

Math Formulation of Siren Architectures

Speaking briefly about essential math details about Siren Deep Neural Net architectures, we can say that these precise kind of models are mathematically formuled as follow:

- as Siren paper's authors say, We are interested in a class of functions Φ that satisfy equations of the form:

$$F(x, \Phi, \nabla_x \Phi, \nabla_x^2 \Phi, \dots) = 0, \Phi : x \to \Phi(x) \tag{1}$$

- 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. Our goal is then to learn a neuralnetwork that parameterizes Φ to map x to some quantity of interest while satisfying the constraintpresented in Equation (1).
- Thus, Φ is implicitly defined by the relation defined by F and we refer to neural networks that parameterize such implicitly defined functions as *implicit neural representations*.

Siren paper's Related Works

Speaking briefly about essential related works cited within Siren's study, we can mention, reading the paper itself, the following earlier researches:

- They report that recent work has demonstrated the potential of fully connectednetworks as continuous, memory-efficient implicit representations for shape parts, objects or scenes.
- Also, they claimed that in previous works in addition to representingshape, some of these models have been extended to also encode object appearance, which can be trained using (multiview) 2D image data using neural rendering.

Periodic Nonlinearities Earlier Studies

While looking at preivous done researches about employing periodic nonlinearities, within Siren paper we can learn about the following facts:

- Firstly, that *Periodic Nonlinearities* have been investigated repeatedly over the past decades, but have so far failed to robustly outperform alternative activation functions
- Early work includes *Fourier neural networks*, engineered to mimic the Fourier transform via single-hidden-layer networks
- Other work explores neural networks with periodic activations for simpleclassification tasks. It has been shown that suchmodels have universal function approximation properties.
- Compositional pattern producing networks also leverage periodic nonlinearities, but rely on a combination of different nonlinearities via evolution in a genetic algorithm framework.

Compression Techniques

Experiments Design

We decided to organize and design our experiments following a strict and precise series of subsequent steps that allow us to break down the different targets and goals we identified for accomplishing the case study we want to pursue. In particular we proceed in the following manner:

- Firstly, we selected the **computing tools** by means of which we were able to produce, collect and show meaningful results and insights about different trials.
- Then, we identified **input target image** that will represent our target image as well as training set, that should be compressed using some kind of compression technique for Deep Nets.
- We created ahead of compressing time, both *Jpeg and Plain Siren datasets* that have made up our **control group** against which comparing later should be made taking into account compressed counter part models.

Experiments Design (2)

We decided to organize and design our experiments following a strict and precise series of subsequent steps that allow us to break down the different targets and goals we identified for accomplishing the case study we want to pursue. In particular we proceed in the following manner:

- After haivng produced at least a bounch of datasets planty of data points representing examples of control group we collected data for different compressing algorithm. In particular we focused on:
    - **Automated Gradual Pruning** by *Michael Zhu, Suyog Gupta et al., 2017)* - as an instance of a possible pruning like compressing method;
    - **Linear Range Quantization** by *Benoit et al., 2018* - a *Quantization Aware Training Technique* which support **Symmetric, Asimmetric both Signed and Unsigned features along with the possibility of declaring the level of quantization** to which set each layer of a distinct deep model.

Tools Used for Train and Create Reports

Here, we report the principal tool we decide to use and employ to carry out both training phase of different models as well as producing resulting graphics showing major details we care of. In particular:

- For training Deep Neural Network models we made use of **Distiller IntelLab framework**:
  - it has been created exploiting **Pytorch Deep Learning Framework**, that employes **torch core module as back-end engine** for dealing with both tensor, i.e. multidimensional arrays of values, as well as computing procedur for tackling back-propagating algorithm
  - it was extended when necessary to accomodate specific tasks we designed for accomplishing our analyses.
- **Mathplotlin, Seaborn, Numpy and Pandas** - computing and graphics tools for dealing with post training results investigation and producing interesting charts or graphics for creating final reports. These libraries and python modules have been employed also for constructing distinct datasets for:
  - Jpeg, Plain Siren architectures, Pruned and Quantized models datasets.
  - Creating temporary *.csv* file for recording hyper-params setting to be tested or already trained, as our **database back-end**.

Identifyed Target Image for Training Siren Models

We decide to adopt, as in Siren related paper, Camera image as our target image of which we aim at inferring an implicit representation to be compared against to jpeg compressed couterpart images as well as pruned, quantized and both pruned-quantized compressed models

Figure: Camera 512x521 target image

Image Feature	Value
name	Camera
shape	(512, 512)
size_byte	262144
image_band	(L,)

Table: Camera Image Characteristics

Camera Image, which is a picture where a man was shoted while he was recoding via camera located on a top of tripod.

Identifyed Target Image for Training Siren Models (2)

Differently from Siren paper's Camera Image, we decide to resize it, by cropping the full image down to 256x256 image about its center, leading to the following update image:

Figure: Camera 256x256 target image

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

Table: Cropped Camera Image Characteristics

Data Distribution for Target Image for Training Siren Models

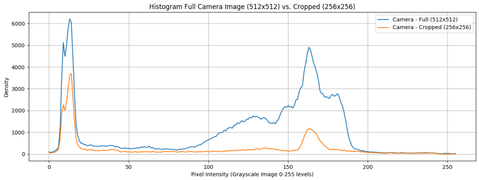
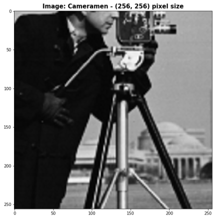
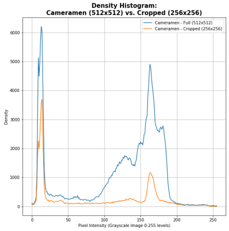
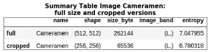


Figure: Camera target image, full and cropped data distribution

General Summarizing Image Overview






Summary Table Image Cameramen:
full size and cropped versions

	name	shape	size_byte	image_band	entropy
full	Cameramen	(512, 512)	262144	(L,)	7.047955
cropped	Cameramen	(256, 256)	65536	(L,)	6.780318

Jpeg Dataset

For producing jpeg data examples with which we have filled jpeg dataset as part of control group along with data points representing plain Siren like trained models, e.i., not compressed architectures. In particular, we have compressed target cropped camera image in the following manner:

- Quality tested within range going from 20% up to 95% for quality tuned setting.
- Dividing overall calculated compression images into three distinct groups that are low, mid and high quality classes, describe as follows:
  - **low quality** - data examples calculated setting a quality score lower than or equal to 50%;
  - **mid quality** - data examples calculated setting a quality score comprised between 50% ad 80% value;
  - **high quality** - data examples calculated setting a quality score greater than or equal to 80%.

Jpeg Dataset

The choices claimed lead to the following results, speaking about Jpeg Dataset, about which we reporeted three distinct instances:

	size(byte)	footprint(%)	cmprss-class	psnr	bpp
small	4133.0	4.921410	JPEG:20	33.270993	0.504517
medium	8355.0	9.948797	JPEG:70	39.907118	1.019897
large	19863.0	23.652060	JPEG:95	48.693716	2.424683

Table: Jpeg Dataset Instances example

Graphics Jpeg Dataset Insights

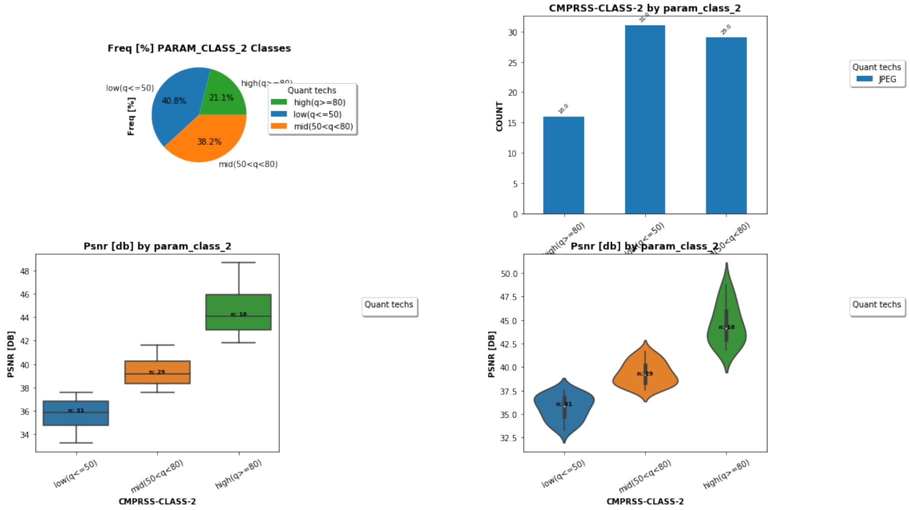


Figure: Chart about Pie, Bar, Box, and Violin plots showing summary stats about collected data for jpeg compressing tech

Graphics Jpeg Dataset Scatter Psnr[db] vs Bpp

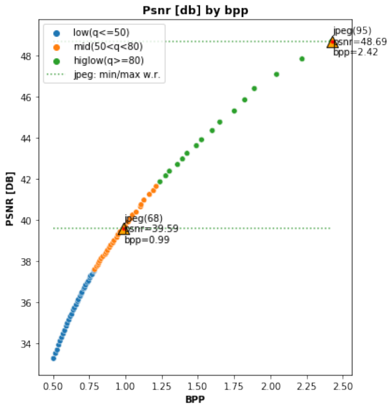


Figure: Scatter Plot Psnr [db] vs Bpp for Jpeg Dataset.

Here, it is reported a Scatter plot, where on the x-axis was located Bpp, while on y-axis was set Peak-Signal-to-Noise-Ration, shortly Psnr, expressed in decibel, where the highere the Psnr score the better the algorithm was ebla to keep signa data over noise introduced by the procedure itself; whereas speaking about Bpp metric, we know that the lower the score the better the compressing procedure was able to keep meaningful signal information while reducing the number of pixels needed to represent the overall compressed image.

Baseline Siren Dataset Design - Hyper-Params Choices

For the construction of Baseline Siren data points, which are nothing but instances and rows within the resulting dataset representing trained Siren Deep Nets, we proceeded in the following manner. In more precise words, we established which degree of freedom employing for training several different configurations, and doing so, we end up claiming that we allowed for each different trial to set arbitrarily the number of hidden layers and the numbber of hidden features. Where, we can say furthermore that:

- **"Number of Hidden Features"** - which are the number of weigths (also referred to as parameters to be learnt), we decided to try the following list of possible values: $[75., 55., 85., 64., 45., 90., 35., 25., 8., 32.]$, a total of 11 possible choices.
- **"Number of Hidden Layers"** - whose values indicate the deepness of a Net, we decided to try the follwoing list of possible values: $[10., 11., 12., 13., 3., 4., 5., 6., 7., 8., 9., 2.]$, a total of again 11 possible choices.

As we can understand from the proposed list of values one for number of hidden features and the other of the number of hidden layers, we also

Baseline Siren Dataset Design - Dividing Data into Groups depending on Model's Size (Byte) (2)

As we can understand from the proposed list of values one for number of hidden features and the other of the number of hidden layers, we also include some combinations that lead to distinct settings that for sure do not correspond to real, suitable, and proper Net's hyper-param settings, that is, some possible model's configurations can be discarded since are too tiny or too large and deep and furthermore correspond to overly under-parameterized or over-parameterized Architectures. In fact for describing this observation, indeed, we also decide to split the overall datasets of plain trained Siren models into three classes which corresponds to three categories described as follows:

- **"under"** - which corresponds to those trials where trained siren architectures are lower or equal to $35844.0B$
- **"mid"** - which corresponds to those trials where trained siren architectures whose sinzew in byte ley in between of the following range $[35844.0, 381484.0]$B
- **"over"** - trials where models's size is larger or equal to $381484.0B$

Baseline Siren Dataset Design - Dividing Data into Groups depending on Model's Deepness (3)

Another possible, meaningful, and interesting observation we can propose regard the fact that we can even organize and so describe baseline data points for plain siren trained models dividing them again into three arbitrary categories depending on the depth reached by each model regardless the net's size, as follows:

- **"tiny"** - which corresponds to those trials where trained siren architectures that ahve a lower or equal to 5 levels of hidden layers;
- **"mid"** - which corresponds to those trials where trained siren architectures whose depthley in between of the following range $[5, 9]$B
- **"deep"** - which corresponds to those trials where trained siren architectures that ahve a greater or equal to 9 levels of hidden layers;

Baseline Siren Dataset Design - Summary Tables (4)

param_class	size(byte)	psnr	bpp	n_hf	n_hl	deepness
under	2724.0	17.202	0.333	8.0	9.0	mid($[5,9]$)
mid	34308.0	39.487	4.188	32.0	8.0	mid($[5,9]$)
over	293764.0	57.340	35.860	85.0	10.0	deep(≥ 9)

Table: Example of table with samples from plain Siren Dataset when grouped by Number of parameters, expressed as overall Model's Size in byte

deepness	size(byte)	psnr	bpp	n_hf	n_hl	param_class
tiny(≤ 5)	708.0	18.517	0.086	8.0	2.0	under($\geq 35.0KB$)
mid($[5,9]$)	117508.0	54.111	14.344	64.0	7.0	over($\geq 372.5KB$)
deep(≥ 9)	83524.0	46.880	10.196	45.0	10.0	over($\geq 372.5KB$)

Table: Example of table with samples from plain Siren Dataset when grouped by Model's Depth

Graphics Plain Siren trained Dataset Insights

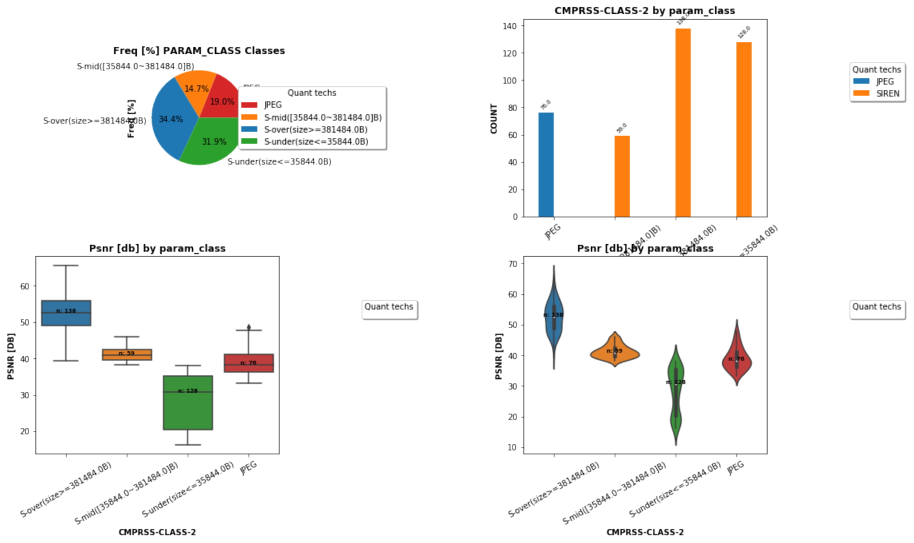


Figure: Chart about Pie, Bar, Box, and Violin plots showing summary stats about collected data grouped by *Param's Number* for plain Siren trained models

Graphics Plain Siren trained Dataset Insights (2)

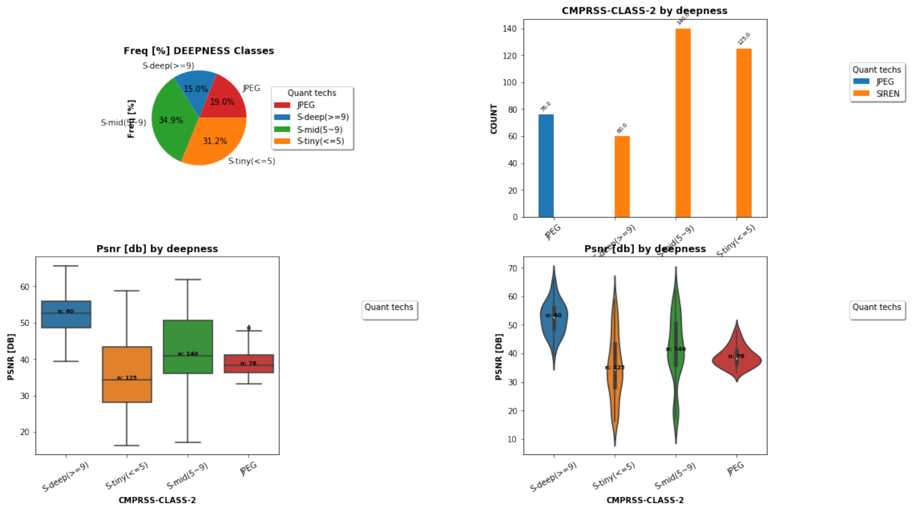


Figure: Chart about Pie, Bar, Box, and Violin plots showing summary stats about collected data grouped by *Deepness* for plain Siren trained models

Graphics Plain Siren Dataset Scatter Psnr[db] vs Bpp

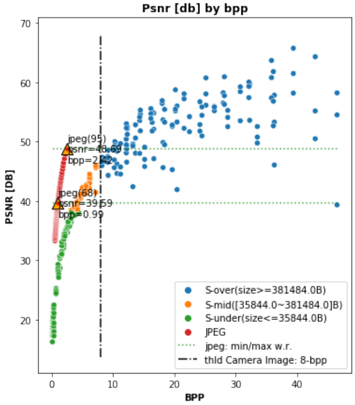


Figure: Scatter Plot Psnr [db] vs Bpp for Plain Siren Dataset, data grouped by *Weigths's Number*.

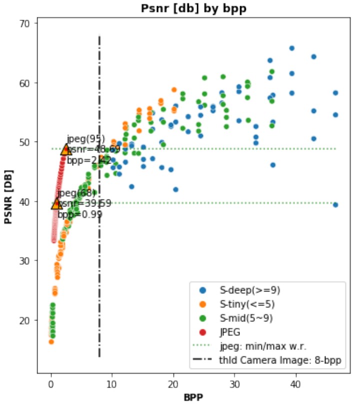


Figure: Scatter Plot Psnr [db] vs Bpp for Plain Siren Dataset, data grouped by *Model's Depth*.

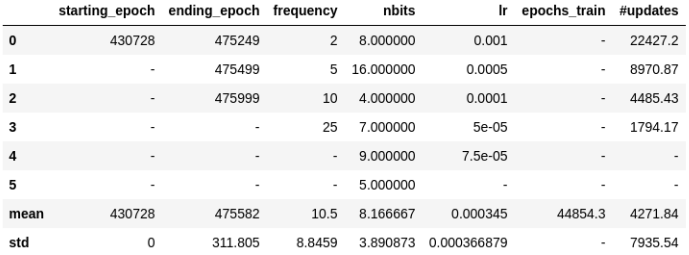
Quantized Siren Dataset Design

For producing quant dataset we adopted firstly Quantization Aware Training Technique known as Linear Range Quantization as described in *Benoit et al., 2018* . More precisely, we also levarage *InteLab's Distiller* implementation of such quantazing technique, and via a configuration file called scheduler in *".yaml"* format we set different setting that we attempt for training quantized resulting models. In particular the scheduler is working around the following tunable options, which represent the degree of freedom the technique is based on:

- **"Frequency"** - it is in charge of activating distiller framework's algorithm with which update stats for carrying out quantization, where tested frequencies have been: $[2, 5, 10, 25]$
- **"Per Channel"** - it is a boolean setting that let experimenter to switch between per Layer or per Channel quantization approach, we fixed to per layer behaviour;
- **"Weight Bits"** - it allows us to set the number of bits with which pass from full floating-point precision to integer-precision, where we tested: $[4, 5, 7, 8, 9, 16]$ bits reduce-precision quantization;

starting_epoch	ending_epoch	frequency	nbits	lr	epochs_train	#updates

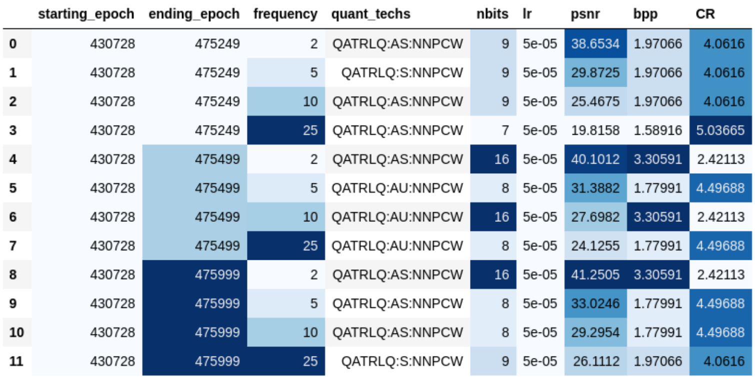
Quantized Siren Dataset Design - Table Hyper-params (2)



	starting_epoch	ending_epoch	frequency	nbits	lr	epochs_train	#updates
0	430728	475249	2	8.000000	0.001	-	22427.2
1	-	475499	5	16.000000	0.0005	-	8970.87
2	-	475999	10	4.000000	0.0001	-	4485.43
3	-	-	25	7.000000	5e-05	-	1794.17
4	-	-	-	9.000000	7.5e-05	-	-
5	-	-	-	5.000000	-	-	-
mean	430728	475582	10.5	8.166667	0.000345	44854.3	4271.84
std	0	311.805	8.8459	3.890873	0.000366879	-	7935.54

Figure: Table Hyper-Params Settings for leading Quantization Training Aware techniques as Linear Range Quantization

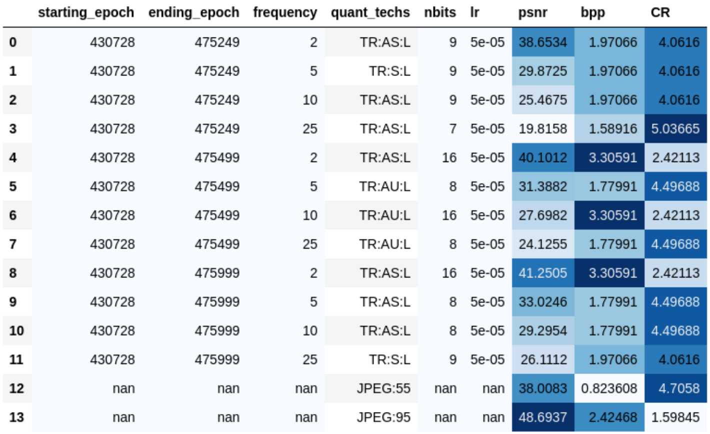
Quantized Siren Dataset Comparing Results - Table (1)



	starting_epoch	ending_epoch	frequency	quant_techs	nbits	lr	psnr	bpp	CR
0	430728	475249	2	QATRLQ:AS:NNPCW	9	5e-05	38.6534	1.97066	4.0616
1	430728	475249	5	QATRLQ:S:NNPCW	9	5e-05	29.8725	1.97066	4.0616
2	430728	475249	10	QATRLQ:AS:NNPCW	9	5e-05	25.4675	1.97066	4.0616
3	430728	475249	25	QATRLQ:AS:NNPCW	7	5e-05	19.8158	1.58916	5.03665
4	430728	475499	2	QATRLQ:AS:NNPCW	16	5e-05	40.1012	3.30591	2.42113
5	430728	475499	5	QATRLQ:AU:NNPCW	8	5e-05	31.3882	1.77991	4.49688
6	430728	475499	10	QATRLQ:AU:NNPCW	16	5e-05	27.6982	3.30591	2.42113
7	430728	475499	25	QATRLQ:AU:NNPCW	8	5e-05	24.1255	1.77991	4.49688
8	430728	475999	2	QATRLQ:AS:NNPCW	16	5e-05	41.2505	3.30591	2.42113
9	430728	475999	5	QATRLQ:AS:NNPCW	8	5e-05	33.0246	1.77991	4.49688
10	430728	475999	10	QATRLQ:AS:NNPCW	8	5e-05	29.2954	1.77991	4.49688
11	430728	475999	25	QATRLQ:S:NNPCW	9	5e-05	26.1112	1.97066	4.0616

Figure: Table of just Quantization Training Aware techniques as Linear Range Quantization

Quantized Siren Dataset Comparing Results - Table (2)



	starting_epoch	ending_epoch	frequency	quant_techs	nbits	lr	psnr	bpp	CR
0	430728	475249	2	TR:AS:L	9	5e-05	38.6534	1.97066	4.0616
1	430728	475249	5	TR:S:L	9	5e-05	29.8725	1.97066	4.0616
2	430728	475249	10	TR:AS:L	9	5e-05	25.4675	1.97066	4.0616
3	430728	475249	25	TR:AS:L	7	5e-05	19.8158	1.58916	5.03665
4	430728	475499	2	TR:AS:L	16	5e-05	40.1012	3.30591	2.42113
5	430728	475499	5	TR:AU:L	8	5e-05	31.3882	1.77991	4.49688
6	430728	475499	10	TR:AU:L	16	5e-05	27.6982	3.30591	2.42113
7	430728	475499	25	TR:AU:L	8	5e-05	24.1255	1.77991	4.49688
8	430728	475999	2	TR:AS:L	16	5e-05	41.2505	3.30591	2.42113
9	430728	475999	5	TR:AS:L	8	5e-05	33.0246	1.77991	4.49688
10	430728	475999	10	TR:AS:L	8	5e-05	29.2954	1.77991	4.49688
11	430728	475999	25	TR:S:L	9	5e-05	26.1112	1.97066	4.0616
12	nan	nan	nan	JPEG:55	nan	nan	38.0083	0.823608	4.7058
13	nan	nan	nan	JPEG:95	nan	nan	48.6937	2.42468	1.59845

Figure: Table of Quantization Training Aware techniques as Linear Range Quantization compared over Jpeg

Quantized Siren Dataset Current Tested Weigh Bits Choices

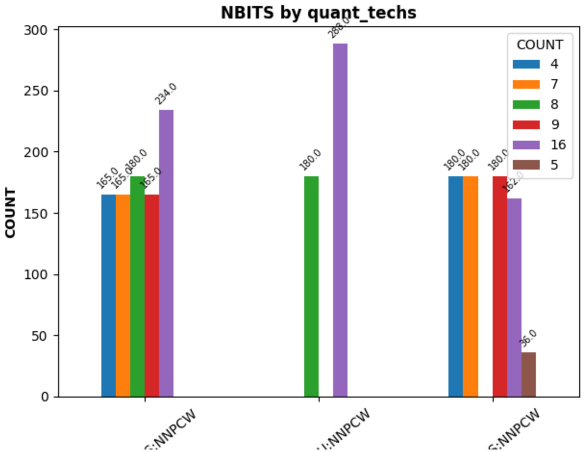
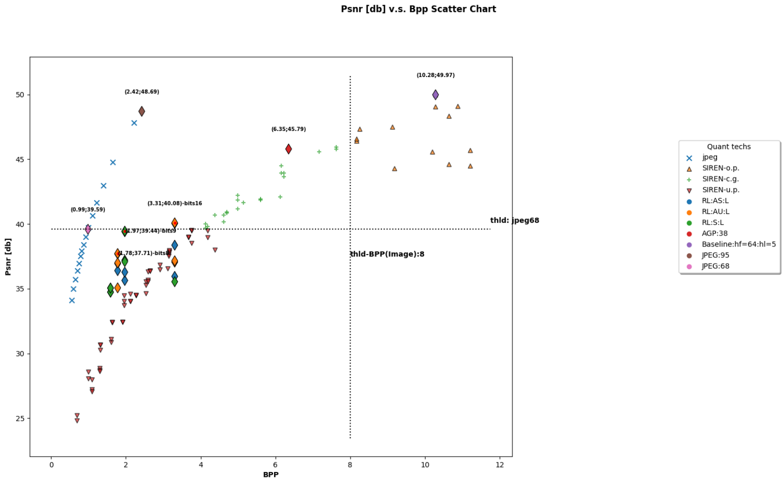
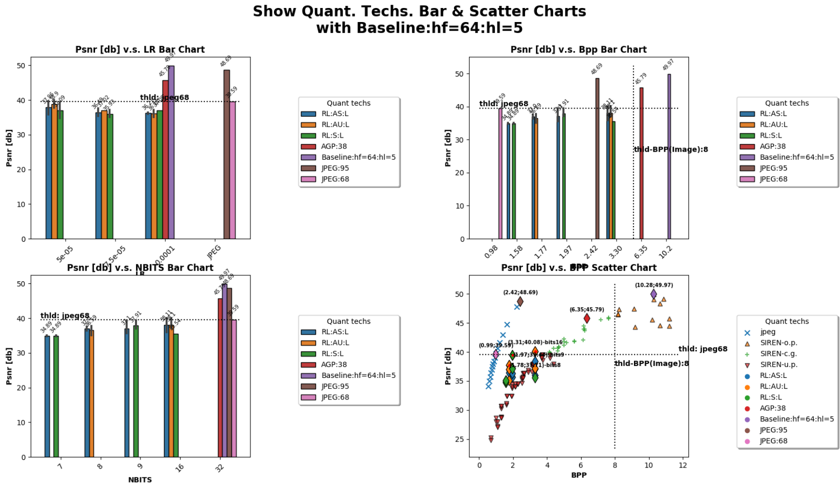


Figure: Quantized Siren Dataset Current Tested Weigh Bits Choices

Quantized Siren Dataset Comparing Results - Scatter plot Psnr[db] vd Bpp



Quantized Siren Dataset Comparing Results - Summary Plot


Show Quant. Techs. Bar & Scatter Charts
with Baseline:hf=64:hl=5
Psnr [db] v.s. LR Bar Chart
Psnr [db] v.s. Bpp Bar Chart
Psnr [db] v.s. NBITS Bar Chart
Psnr [db] v.s. BPP Scatter Chart
Psnr [db]
LR
NBITS
BPP
thld: jpeg68
thld-BPP(Image):8
Quant techs
RL:AS:L
RL:AU:L
RL:S:L
AGP:38
Baseline:hf=64:hl=5
JPEG:95
JPEG:68
jpeg
SIREN-o.p.
SIREN-c.g.
SIREN-u.p.
(2.42;48.69)
(10.28;49.97)
(6.35;45.79)
(0.99;39.59)
(3.31;40.08)-bits16
(1.97;39.48)-bits9
(1.78;37.71)-bits8