Meta Learning For Feature Grid-Based INR

Andrew Tianbao Zhong
azhong13@umd.edu

Darshit Miteshkumar Desai
darshit@umd.edu

Hitesh Kyatham
hkyatham@umd.edu

Suriya Suresh
jsuriya@umd.edu

Vinay Krishna Bukka
vinay06@umd.edu

Abstract

In recent times, implicit neural representations(INRs) have gained popularity as a method of representing signals as implicit function representations that are continuous. INRs employ the use of a neural network to estimate the function to represent signals[15]. In this project, we explore the idea of meta-learning with a feature grid based INR. Feature Grid based INRs were researched as a alternative to get over the long training times of coordinate based INRs. We propose integrating a transformer based meta-network for predicting weights of a feature grid based INR.We have carried out experiments with parameter tuning and compared with baseline methods.Our findings highlight the challenges in training transformers for such tasks and provide avenues for future research. Our code can be found at https://github.com/vinay06vinay/Meta-Learning-Based-INR- .

1. Introduction

Implicit neural networks (INR) have become a hotbed of research in recent years, being commonly used as compression algorithms for any signal, predominantly images[5], videos[10], shapes[9], audio[15] or radiance fields[1] being used as inputs. They are neural networks that learn to represent a continuous function of a signal. This also makes them more efficient than discrete-based representations of images in terms of storage use. Given an input of a m dimension, they return a predicted output with n dimensions.For our work, we focus only on images whose input is $(x, y)\epsilon\mathbb{R}^2$ and the outputs are RGB values $\epsilon\mathbb{R}^3$.

Implicit neural networks are typically Multi-layer Perceptrons (MLPs) that take in image coordinates as input and predict the corresponding pixel values directly. An advantage of this type of representation is that an image can be stored and represented independently of its spatial resolution [9]. However, the cost of training scales up with higher resolution and number of images as each image has to be trained separately.This can lead to significant computational and storage costs when used in practical applications. Methods that overcome the slow training times can be categorized as; representing the images as embeddings (latent space) [3, 13, 15], using meta-learning to learn priors [2, 14, 17] and feature grid representations [5, 11].

Meta-learning is essentially learning how to learn. It can be considered as the practice of learning priors to learn new unseen target data faster. We use meta-learning here to predict the initialization parameters of a target network. There have been many meta-learning methods that have been proposed in literature. Examples of previous works that have done meta-learning for INRs are MetaSDF[14], Learnit [17], and TransINR [2]. These works are primarily focused on coordinate-based INRs, As far as we are aware, there have been no publications to date regarding meta-learning for feature grid-based INRs.

In this work, we propose using SHACIRA[5], a feature grid-based INR as the target network for which meta-learning is done. We integrate the transformer network from TransINR [2] as the meta-network for this task. We target to predict weights for the codebook and decoders used within SHACIRA using the meta-network and updating the initialized weights by minimizing the L2 loss of the ground truth value and predicted value. We use a transformer-based meta-network as it doesn't require gradient computation which can be heavy and allows the transformer to perform faster.We aim to reduce the compression time of SHACIRA by meta-learning the weights preemptively.The advantage of such a pipeline would be reductions in the encoding time of SHACIRA (or any other feature grid-based INR) even beyond the current low encoding times of SHACIRA. This would help in developing an end-to-end compression pipeline which is real-time encoding performance as well as real-time decoding performance. This is considering the transformer training time is not counted in the total encoding time and only the inference time is counted in the encoding time which is used for predicting weights.

We carry out experiments to understand the performance compared to baseline methods SHACIRA without any mod-

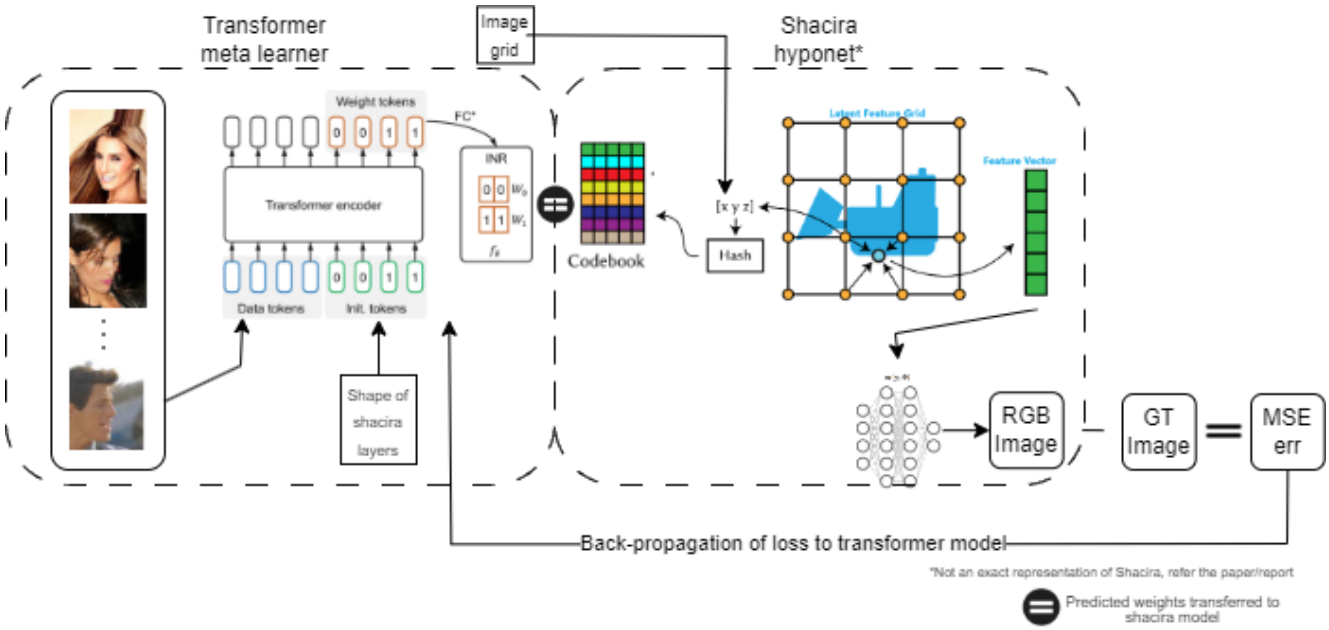

Figure 1. TransSHACIRA - The Hypo Network Working Pipeline

ifications [5] and TransINR with the default MLP[2]. The output that was obtained by our proposed pipeline did not have satisfactory results and as a result, we underwent hyper-parameter tuning and have summarized the results. We have analyzed the results and listed the potential challenges and causes that resulted in the results we have described. Our further analysis details the possible avenues that can be pursued to achieve meta-learning for feature grid-based INRs. We summarize our contributions below:

- We propose introducing a transformer-based meta-learner for meta-learning on a preexisting feature grid-based Implicit Neural Representation.
- We carry out extensive hyper-parameter tuning and carry out a study of it.
- Our analysis of the results shows the challenges of trying to carry out meta-learning using a transformer-based meta-learner on a Feature Grid-based INR.

2. Related Work

2.1. Implicit Neural Representation

Recent work on Implicit Neural Representations (INRs) has shown the power of these methods to store signals efficiently as continuous functions and independent of the source signal resolution. To alleviate the long training time, various methods were proposed by various authors to overcome this. One such method as described earlier is embedding the image in a different space and passing it on as the input or as a secondary input in addition to the input image. Other methods that are proposed in various works of literature are meta-learning and using an alternative representation of the input image in the form of feature grids.

Sitzmann *et al*. [15] proposes using sine activation functions in addition to using priors modeled in latent space that significantly reduce training time. Other works also proposed the idea of doing the same like DeepSDF [13] and Local Implicit Image Function (LIIF) [3].DeepSDF [13] uses a latent input only decoder type architecture that optimizes the randomly initialized latent vector for every backpropagation.LIIF use a similar approach where the image is represented as a set of latent codes and uses nearby latent features as inputs. A major disadvantage of these methods is that using a single latent vector can cause some finer details to be missed and limits the resolution of the embedding of the image.

2.2. Feature Grid INRs

Feature grid INRs use feature representations of the input at various levels of detail and scale so that local and global features can be effectively captured. This allows for faster training times and quicker convergence.However, the resulting feature grids can consume a lot of space making it unsuitable if resources are limited, which tends to be in real-time applications. Some examples of feature grid INRs are InstantNGP [11] and Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes [16]. Takikawa *et al*. [16] proposes the use of a voxel octree to represent a collection of feature vectors obtained from the input. Voxel octrees are efficient data structures that do not represent empty voxels and allow for space-efficient storage

of feature vectors. Higher reconstruction quality results in higher memory use by the stored vectors.

Muller *et al*. [11] proposed InstantNGP which uses feature grids at various levels of details in a similar manner and are indexed in a hash table for each resolution whose access time is very low. The memory used is high for storing the feature grids. It is also very fast, giving outputs in a matter of minutes with real-time visualizations. Girish *et al*. [5] builds upon the work of InstantNGP.

2.3. Metalearning for INRs

Metalearning is used in a few applications, a common one being few shot learning of data. In this case, the model learns priors from a few examples of a given task and is able to learn and perform on unseen fresh examples of the same task. In the context of INRs, they are used to learn initialization weights for INRs that are usually fixed after it is learned. Popular examples of gradient-descent-based meta-learning methods are MAML [4] and Reptile [12].

Two types of meta-learners have been used in previous work for INRs, gradient-based meta-learners like MetaSDF [14], Learnit [17] and non-gradient-based meta-learner like TransINR [2].MetaSDF[14] uses MAML[4] to learn weights to fit SDF functions into a neural network.Learnit [17] attempts to do the same but can use MAML[4] or Reptile[12] to learn weights to fit a wider variety of input signals. This is simple in its implementation. Both offer faster convergence compared to random weight initialization for INRs.Another example is MetaSparseInR [7] which takes an approach of training a class of INRs for a set of signals so that individual signals can be trained with few optimization steps. The approach involves pruning of weights for the initial class of INRs while training. This seeks to learn a general INR for a whole of data while others seek to learn for a specific signal.

Another approach involves the use of transformers as described in TransINR [2]. These do not have any gradient optimizations, avoiding the need for multiple forward and backward passes. The use of tokens in the transformer allows optimization to occur within it after each iteration when the loss value is updated. Figure 2 shows the pipeline of TransINR.

3. Approach

The main goal of the project is to improve the performance of feature grid-based Implicit neural representations with the integration of meta-learning. Our approach involves understanding and integrating SHACIRA [5] and Trans INR [2].

Section 3.1 explains the methodology of SHACIRA, 3.2 explains the methodology of TransINR, and 3.3 details about how can these two be integrated into forming a pipeline for meta-learning a feature grid-based INR.

3.1. SHACIRA

Feature grid-based INRs address the training costs and scalability issues associated with coordinate-based INRs when dealing with larger input datasets. The feature grid allows for faster convergence by replacing a larger neural network with a multi-scale resolution lookup table of feature vectors and a smaller neural network that can decode the vectors. SHACIRA[5] uses Hash grids for the representation of feature grids inspired from InstantNGP[11]. The multi-resolution hash encoding is described as follows:

- Given an input coordinate, find the surrounding voxels at different resolution levels and hash the vertices of these grids. The hashed vertices are used as keys to look up trainable F-dimensional feature vectors.
- Based on where the coordinate lies in space, the feature vectors are linearly interpolated. The feature vectors from each grid are concatenated, along with any other parameters such as viewing direction.
- The final vector is inputted into the neural network to predict the RGB output.

This particular hashing technique consumes lots of memory. To reduce the amount of memory consumed, SHACIRA proposed the use of latent grids where quantization on initial feature vectors happens such as bringing them down from higher precision to lower precision points. The proposed approach of this work uses the same approach and is as below and shown in figure 2 which is:

1. The first step is to reparameterize the feature grids with quantized latent weights. This involves representing the feature grids as a set of continuous learnable latents, which are then quantized to reduce memory consumption.
2. After reparameterizing the feature grids, SHACIRA applies entropy regularization in the latent space to achieve high levels of compression across various domains. This involves adding a regularization term to the loss function that encourages the latents to have low entropy, which in turn leads to more efficient compression.An annealing approach to perform a soft rounding operation is done to reduce the errors after quantization.
3. The feature vector is then passed through the decoder to obtain the feature grid table which is then indexed at different levels/resolutions using the coordinates to obtain a concatenated feature vector which is passed through an MLP (Multi-Layer Perceptron) to obtain the predicted signal y.

3.2. TransINR

TransINR[2] is a meta-learning hypernetwork that uses transformers to directly predict the weights of an INR network (the ”hyponetwork”), such as the network parameters of a multilayer perceptron INR. Transformers originate from natural language processing applications where they

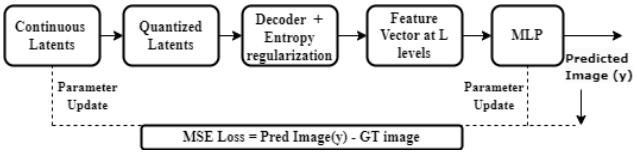

Figure 2. SHACIRA Working Pipeline

are used to process sequences of language tokens. The key idea behind transformers is the attention mechanism, which allows for more important input tokens in a sequence to be weighted higher for each given output prediction.

In contrast to vanilla neural networks, the contributions of each token are computed by the model for each input datum rather than trained to be a fixed value. TransINR formulates INR meta-learning as a function mapping observation tokens to hypo-network weights. For example, in the image INR application, image patches are used as input tokens which are then decoded to weight tokens. The weight tokens are then passed into a fully connected layer to obtain the image INR weights. The resulting image produced by the hypo-network is used to compute mean squared error loss. The pipeline can be seen in Figure 5.

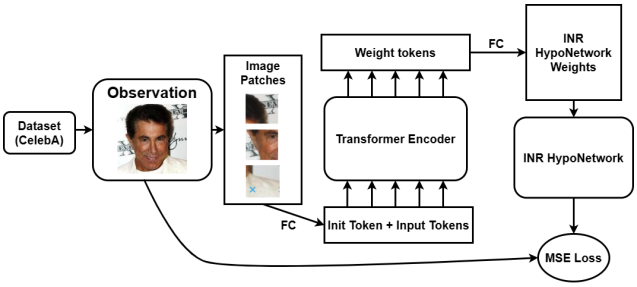

Figure 3. TransINR Training Pipeline

3.3. Meta Learning for Feature Grid INR

Section 3.1 showed how Scalable Hash grids provide an end-to-end network for learning neural fields for images, videos, and NERFs providing high-quality compression and reconstruction quality with a large decrease in memory use. Following it, section 3.2 explained how Transformers are utilized to predict the weights of INR networks thereby leading to faster convergence time of coordinate-based INRs during training and improving the inference time of an INR network during test.

Leveraging the methodology from both works, an approach as below has been implemented and executed to use Transformers in predicting the weights for SHACIRA thus leading to faster training and inference times. Figure 1 shows the system implemented.

1. First, the images undergo a process where they are divided into patches and encoded to create data tokens, which function as inputs for the transformer. Simultaneously, initial weight tokens are generated to represent a set of column weight vectors present in the FC network.
2. Additionally, the initial weight tokens in this Hypo-Network take into account the codebook (or continuous latent vector) which is the input for SHACIRA. This feature distinguishes it from the INR MLP network of Vanilla TransINR.
3. The transformer encoder processes both the data tokens and initial weight tokens, producing weights as its output. These weights are then mapped to column vectors representing INR weights for the layerwise FC layers of the SHACIRA MLP and to the latent codebook feature vector of SHACIRA. This integration replaces the MLP network in the vanilla TransINR network with the SHACIRA network.
4. Following this integration, the transformer now produces weights for the entire SHACIRA network downstream. The Mean Square Error (MSE) loss is then computed between the predicted signal (RGB image) of SHACIRA's MLP and the ground truth image.
5. Based on the calculated loss, the network weights (Initial Weight Tokens) of the transformer encoder, which are learnable parameters, are updated along with the appropriate learning rate.
6. Consequently, the implemented Hypo-Network learns to generalize the network weights of the transformer, providing output weights for use by the SHACIRA network. This overall pipeline accelerates the convergence of the feature-grid-based INR network during both training and inference, while also delivering high-quality compression and reconstruction for images and videos.

4. Experiments And Results

4.1. Datasets

Primarily we used the CelebA[8] dataset as the primary comparison dataset for our results.It was used to compare the results of our proposed method with baseline methods.It was also used for hyperparameter tuning in addition with Imagenette [6] to try to get the best results possible.

(a) Ground Truth Image (b) Prediction Image

Figure 4. SHACIRA Standalone result

4.2. SHACIRA vs TransINR

While analyzing SHACIRA and TransINR, we observed some findings. We can observe that SHACIRA can perform well on smaller images as shown in 4 as well as larger images as indicated by Girish *et al*. [5] on the Kodak dataset. In TransINR, the transformer is solely responsible for generalizing over the image class and producing weights for the INR, which then generates an output. We also observed performance differences between SHACIRA and TransINR, as indicated in the table below.We also noticed the faster convergence of SHACIRA in comparison to TransINR.

Figure 5. TransINR Standalone result. Note TransINR only ran for 1 epoch because of 1.5 hr/epoch training time

Model	Loss	PSNR
SHACIRA	0.0001	40
TransINR	0.01	20

Table 1. Runs of TransINR and SHACIRA standalone models. Note TransINR only ran for 1 epoch because of 1.5 hr/epoch training time

4.3. SHACIRA with TransINR

The advantage of SHACIRA is faster encoding and decoding times, while the advantage of TransINR is single-shot weight prediction for the encoder network without gradient computation. The removal of gradient computation should reduce the already low encoding time of SHACIRA further.Combining the strengths of TransINR and SHACIRA, we leverage the integrated pipeline to attempt to get faster encoding times while maintaining the superior decoding performance of SHACIRA. As depicted in Figures 7 and 6, they represent the general range of results that were obtained with our proposed pipeline.

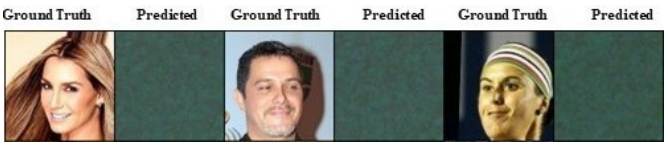

Figure 6. Output on CelebA with color decoder disabled. Note disabling latent quantization didn't make any changes in the quality

Figure 7. Output on CelebA with color decoder enabled Note: disabling latent quantization didn't make any changes in the output quality

4.4. Hyperparameter Tuning

Using SHACIRA as the hyponet for TransINR, we trained the model to predict the hyponet feature grid values on the CelebA[8] and Imagenette[6] datasets. With the default hyperparameters used by Transformer network used in TransINR and the base SHACIRA parameters, reconstruction quality is very poor with an average PSNR of 10 on the CelebA dataset. In comaprison, SHACIRA as a standalone trained model obtains a PSNR on CelebA images of 40. To improve upon this, we conducted the following experiments to determine a good training regime that could create SHACIRA model weights with acceptable reconstruction quality: varying learning rate, batch size, codebook size, and enabling/disabling latent quantization and prediction of the color decoder MLP's weights.

The results for varying the color decoder and latent quantization are mentioned in Figures 6 and 7. The loss and PSNR values are also mentioned in table 2.

LR	Batch Size	LQ	Color MLP	Loss	PSNR
0.05	9	Enabled	Enabled	0.0708	11.80
0.001	8	Disabled	Enabled	0.0625	11.108
0.1	6	Disabled	Disabled	0.0668	12.078
0.01	9	Enabled	Disabled	0.0710	11.781

Table 2. Hyperparameter Tuning Experiments, LQ: Latent Quantization, LR: Learning Rate

Codebook Size	Loss	PSNR
47737	0.0708	11.806
26704	0.0668	12.07
4096	0.0714	11.737

Table 3. TransSHACIRA Codebook variations

4.5. Architecture Changes

Transformer depth change: We performed a few major changes to the existing architectures of both SHACIRA and TransINR when combining both, We dropped the latent decoder and varied transformer depth to understand the impact

of meta network's size on the performance. The following variations were tried:

Transformer Depth	Loss	PSNR
13	0.089	10.806
6	0.0559	12.737
3	0.0743	12.085
1	0.066	12.179

Table 4. Transformer depth variations in TransSHACIRA, Note: for depth=13 only 3 epochs were run because of training time $>$30 mins/epoch

SHACIRA Codebook Shape Change: We tried changing the codebook shape. Since the codebook shape is a $(N, 1)$ vector by design, we tried variations of reshaping the codebook while initializing transformer init tokens. It was reshaped to a (M, N) shape. But that resulted in minor changes in the results, giving the same range of Loss and PSNR. The results of GT vs prediction in case of codebook shape change are given in Figure 8 and 9.The loss and PSNR values are also given in table 3.

Figure 8. Output on Imagenette with Color Decoder disabled and codebook Shape as (M, N) shape with Latent quantization disabled

Figure 9. Output on Imagenette with Color Decoder enabled and Codebook Shape as (M, N) shape with Latent quantization disabled

We also tried changing the type of Loss for transformer model training.The use of L1 loss and Smoothed L1 loss gave a slight increase in PSNR of up to 13 and a loss of 0.04 with the values stagnating at the same after a few epochs.

We also tried to use a Learnit like weight initialization technique where we took the model parameters predicted by Transformer (i.e., weights predicted of SHACIRA which give an MSE loss of 0.03 and PSNR of 11) and use it as an initial starting point for SHACIRA such that it converges faster, but we observed that when using the transformer initialized weights doesn't make a significant dent in the rate of convergence of Shacira and the loss and PSNR profile are pretty much the same i.e. if the Shacira would've been run with randomly initialized parameters it would take the same amount of iterations to converge as the former method.

5. Conclusion and Discussion

We can conclude that, despite our numerous efforts in experimentation and hyperparameter tuning, we have not been able to net any major changes in the metrics nor the output image predictions. Figure 11 show the changes in the PSNR and loss for each of the iterations per epoch. Figure 10 shows the changes in the PSNR and loss per epoch. We can see that the loss and PSNR tend to stagnate after a certain number of epochs.

In this paper we have explored the use of transformer-based meta-learners for feature grid-based INR. With the use of meta-learning, we aimed to build upon the fast training and evaluation times of SHACIRA by using a single forward pass to predict network weights. We hoped to achieve a real-time compression pipeline that performed fast in encoding and decoding images.

We demonstrated that using meta-learning to predict feature grids is a nontrivial task and requires exploration into different training or tuning techniques that could improve performance. Our experiments showed that training regimes with various hyperparameter configurations and architectural changes resulted in only minor changes in performance. To achieve similar results to standalone SHACIRA, further work could be done to directly address why meta-learning using a Transformer struggles with feature grid methods, as well as what strategies could be used to overcome these problems. Additionally, the same architecture could be tested on 3D data like NeRFs. Another approach could have been to use Learnit's[17] average weight initialization to reduce the encoding time of standalone SHACIRA down to 10-20 epochs.

6. Credits

We would like to thank the authors of SHACIRA[5] and TransINR[2] for open-sourcing their code. We would like thank the author of SHACIRA for his suggestions and assistance in this paper.

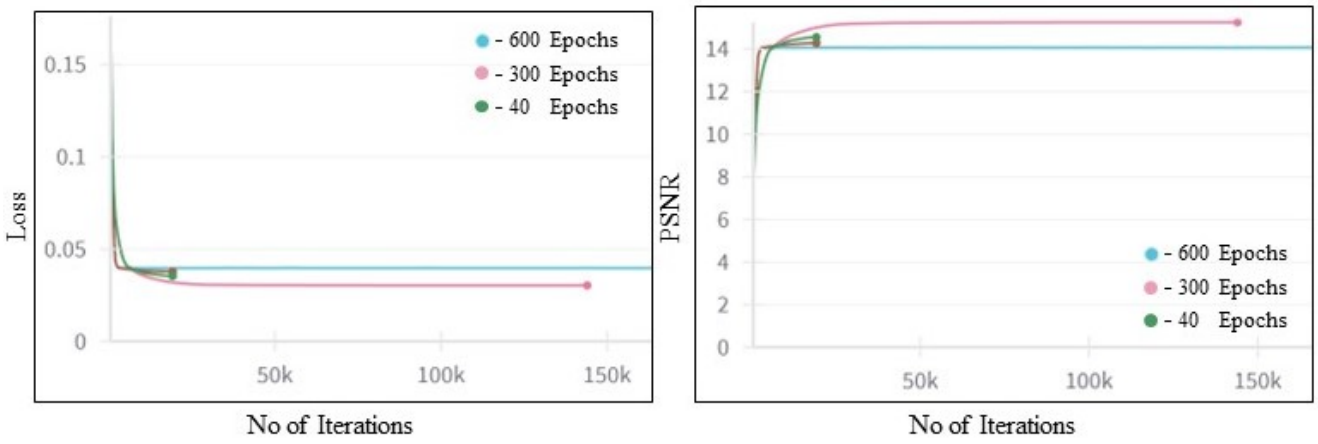

Figure 10. Average Loss and PSNR graphs for different number of Epochs (Note the average psnr and loss was logged at iteration counts at the end of an epoch so x-axis denotes iterations), The purpose of this plots is to give a summary of the various runs which ultimately resulted in stagnated losses. Note the plot with 40 epochs depict the results of running celeba dataset and the plot for 300-600 epochs depict the result of running imagenette dataset

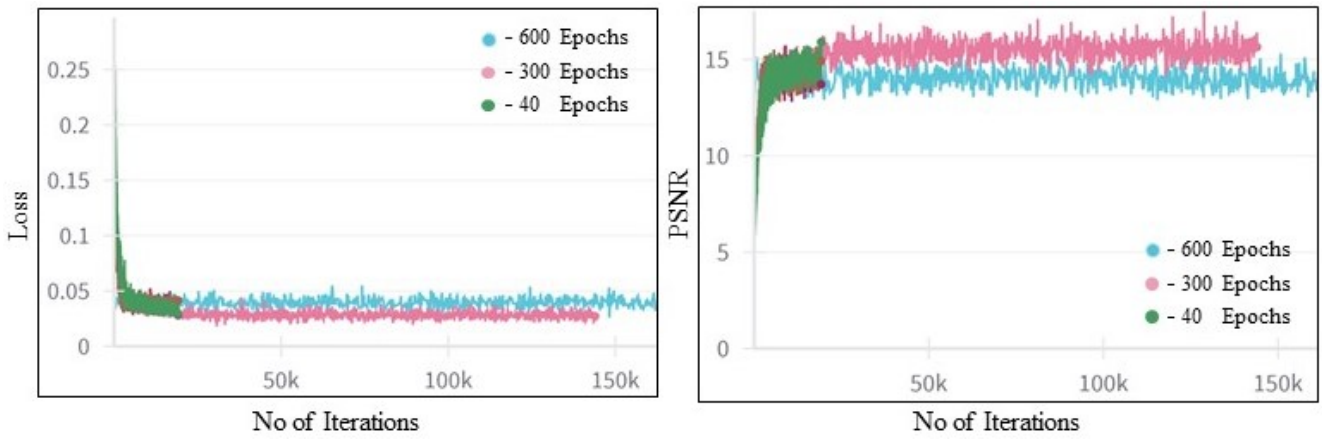

Figure 11. Loss and PSNR graphs per iteration for different number of Epochs, The purpose of this plots is to give a summary of the various runs which ultimately resulted in stagnated losses. Note the plot with 40 epochs depict the results of running celeba dataset and the plot for 300-600 epochs depicts the result of running imagenette dataset

References

[1] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior, 2023. 1

[2] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations, 2022. 1, 2, 3, 6

[3] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function, 2021. 1, 2

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. 3

[5] Sharath Girish, Abhinav Shrivastava, and Kamal Gupta. Shacira: Scalable hash-grid compression for implicit neural representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17513–17524, 2023. 1, 2, 3, 5, 6

[6] Howard, J,Imagenette. https://github.com/fastai/imagenette. 4, 5

[7] Jaeho Lee, Jihoon Tack, Namhoon Lee, and Jinwoo Shin. Meta-learning sparse implicit neural representations. *CoRR*, abs/2110.14678, 2021. 3

[8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 4, 5

[9] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano. Deep learning on implicit neural representations of shapes, 2023. 1

[10] Shishira R Maiya, Sharath Girish, Max Ehrlich, Hanyu

Wang, Kwot Sin Lee, Patrick Poirson, Pengxiang Wu, Chen Wang, and Abhinav Shrivastava. Nirvana: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling, 2022. 1

[11] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2, 3

[12] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018. 3

[13] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. 1, 2

[14] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. In *arXiv*, 2020. 1, 3

[15] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *CoRR*, abs/2006.09661, 2020. 1, 2

[16] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes, 2021. 2

[17] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations, 2021. 1, 3, 6