UNIVERSITÀ
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

THE ROLE OF INTERPRETABLE NEURAL ARCHITECTURES: FROM IMAGE CLASSIFICATION TO NEURAL FIELDS

Zeno Sambugaro

Advisor

Prof. Nicola Conci

Università degli Studi di Trento

April 2024

Acknowledgements

I would like to thank:

- my advisor, Nicola, for guiding and supporting me through the non-linear journey of the PhD program. I also extend my sincere thanks to the reviewers of my thesis for their time, consideration, and suggestions for improvement.
- all the mmlabber, from the elders Nicola and Niccolò for their guidance during the start of my PhD experience and the newbies Giulia, LorenzoB, LorenzoO, Ardan and Antonio, thank you for sharing a lot of good and less good moments inside the 170! Thanks also to the Micio Miao for the ping pong and calcetto matches during the lunch break.
- Giuseppe for being my roommate right from the start of my university journey and a wonderful colleague within the university walls. It's been a fantastic journey, and I owe you a lot for your guidance, especially in the first years of the bachelor's degree. We both have grown and changed a lot during this time; whether for the better or not, that's up for debate ;)
- my lifelong friends Manuel, Michele, and Riccardo, and all the Beppa friends who have shared countless memorable nights at the TrapHouse, numerous holiday and sports with me.
- my family, who have always been supportive and caring. I also thank the Soave family for the many good times we have shared together.
- last but not least, Alessia for being my partner in crime through every adventure I've had and for all those we hope to share in the future. Your love and support have been invaluable during these times, and I could not ask for anything better.

Abstract

Neural networks have demonstrated outstanding capabilities, surpassing human expertise across diverse tasks. Despite these advances, their widespread adoption is hindered by the complexity of interpreting their decision-making processes. This lack of transparency raises concerns in critical areas such as autonomous mobility, digital security, and healthcare. This thesis addresses the critical need for more interpretable and efficient neural-based technologies, aiming to enhance their transparency and lower their memory footprint. In the first part of this thesis we introduce Agglomerator and Agglomerator++, two frameworks that embody the principles of hierarchical representation to improve the understanding and interpretability of neural networks. These models aim to bridge the cognitive gap between human visual perception and computational models, effectively enhancing the capability of neural networks to dynamically represent complex data. The second part of the manuscript focuses on addressing the lack of spatial coherency and thereby efficiency of the latest fast-training neural field representations. To address this limitation we propose Lagrangian Hashing, a novel method that combines the efficiency of Eulerian grid-based representations with the spatial flexibility of Lagrangian point-based systems. This method extends the foundational work of hierarchical hashing, allowing for an adaptive allocation of the representation budget. In this way we effectively preserve the coherence of the neural structure with respect to the reconstructed 3D space. Within the context of 3D reconstruction we also conduct a comparative evaluation of the NeRF based reconstruction methodologies against traditional photogrammetry, to assess their usability in practical, real-world settings.

Keywords
Computer Vision, Deep Learning, Computer Graphics, Neural Radiance Fields, Explainability

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the rapidly evolving landscape of computer science, neural networks have emerged as an important tool for innovation and advancement. Their exceptional ability to learn from and adapt to a wide set of data has changed the way we approach complex problems across diverse domains. The application of neural networks has transformed traditional methodologies, offering novel solutions and unprecedented efficiency.

A particularly important impact of neural networks is observed in the realms of computer vision and graphics. In this contexts, they have not only enhanced existing technologies but also introduced groundbreaking developments. Among these, Neural Radiance Fields (NeRF) stand out as a remarkable achievement. NeRF's approach to 3D rendering and representation has enabled an unprecedented level of realism. This advancement particularly highlights the transformative impact of neural networks on our vision technologies.

Despite their impressive capabilities, neural networks often operate as enigmatic "black boxes", hiding the logic behind their decision-making processes. This lack of transparency is particularly problematic in critical sectors such as autonomous mobility, digital security, and healthcare, where the necessity for safety and reliability is a must. To address this concerns we introduce Agglomerator, an innovative approach, theorized by Hinton in [40], that tries to bridge

this gap by exploiting concepts coming from cognitive science. In particular this architecture focus on discovering part-whole hierarchies in visual data to learn representations and enhance our ability to interpret and trust its decisions. In the realm of computer graphics, the efficiency of Neural Radiance Fields (NeRF) training routines received a considerable boost from InstantNGP [78]. This method has yielded outstanding results in both the quality and speed of NeRF reconstruction. However, the reliance of this method on hashtables, affected by collisions, causes the loss of Lagrangian properties within the framework. This disruption of Lagrangian properties compromise the spatial coherence between the implicit representation and the actual 3D signal reconstruction. This issue highlights a critical trade-off between efficiency and interpretability in the representation of 3D environments.

To overcome the spatial coherence challenges inherent of current NeRF training methods, we propose Lagrangian Hashing, a novel representation for neural fields. This technique combines the rapid training advantages of Eulerian grid-based systems, such as InstantNGP, with the precision and detail offered by point-based representation methods like 3D Gaussian Splatting [48] and Point-NeRF [124]. By embedding point-based features within the high-resolution layers of InstantNGP's hierarchical hash tables, Lagrangian Hashing effectively respects the Lagrangian properties essential for accurate 3D signal reconstruction, thereby enhancing both the interpretability and efficiency of neural field representations.

Lastly, we present a study on the impact of radiance-based representations on 3D reconstruction for industrial applications. Specifically, we present a comprehensive comparison of classical photogrammetry with state of the art NeRF based techniques[48, 78] in terms of fidelity of the reconstruction and precision of the resulting point cloud. More in detail, the structure of the thesis is organized as follows:

- In the first part (Chapter 2) we explore neural network architectures from

the multilayer perception to state-of-the-art Vision Transformers [114], with a particular focus on architectures focusing on discovering part-whole hierarchies in the data, like Capsule networks [93]. We then present our work on Agglomerator and its extension Agglomerator++.

- In the second part (Chapter 3) we delve into 3D representations for 3D data, starting from voxels for the first project SinGAN-3D in section 3.1 and then Neural Radiance Fields reconstruction, presenting the first pure neural methods and the further development in the field with hybrid NeRF methods. In Section 3.4 we highlight the issues with these fast and efficient representation and how we address these issues with our novel method Lagrangian Hashing. In the context of Graphics representation we show a comparison of classical photogrammetry and Implicit representations in Chapter 4.

- In Chapter 5 we draw some conclusions on the thesis and discuss about future directions in the context of Neural Network interpretability and compact 3D representations.

Chapter 2

Representing Part-whole Hierarchies in Neural Networks

2.1 Part-whole hierarchies in Neural Networks

Neural networks have demonstrated outstanding capabilities, surpassing human expertise across diverse tasks. However, a significant drawback of these advanced neural models is the poor accessibility to understand and interpret the network response to a given input. This lack of clarity becomes particularly problematic in sectors where safety and reliability are paramount, such as in autonomous mobility, digital security, and healthcare, where the absence of interpretable insights can erode trust in these otherwise highly accurate systems. Additionally, traditional performance metrics such as accuracy offer an incomplete picture of a system's effectiveness in complex, real-life situations.

Substantial psychological findings indicate that the human visual system decomposes scenes into part-whole hierarchies, establishing consistent spatial relationships through internal coordinate transformations [39]. To mirror this capability in neural networks, we must devise ways for these models to represent such hierarchies. Traditional neural networks falter here, unable to dynamically assign neuron groups to specific hierarchical nodes. This limitation prompted the exploration of "capsule" networks, where neuron clusters, or cap-

sules, are dedicated to representing distinct image segments within a specified region [43, 54, 93]. This framework enables the construction of parse trees by activating specific capsules and their linkages, facilitating a more human-like interpretation of visual data.
Efforts to make neural networks more interpretable is not a recent phenomenon, with its origins deeply embedded in the foundational periods of artificial intelligence and cognitive science. The foundational work in this field aimed to bridge the gap between human cognitive processes and computational models, particularly in how complex systems like the human brain parse and understand visual information.

A foundational theory that significantly shaped the direction of neural network interpretability originated in cognitive science. Researchers like Marr (1982) laid the groundwork with his theory of vision, proposing that visual processing involves multiple stages – from the raw image to a 3D understanding of the world. Influenced by this perspective, early efforts in computer vision aimed to replicate these processes computationally, striving to design systems capable of recognizing image features and comprehending their spatial and hierarchical connections, mirroring human visual perception.

Parallel to these developments, early neural network models started to incorporate hierarchical structures, an approach inspired by both biological neural systems and cognitive theories. These hierarchical models aimed to represent complex data in a multi-layered fashion, where lower layers captured basic features and higher layers integrated these features into more abstract representations. The Neocognitron, introduced by Fukushima in 1980, is an early example of such a model. It was designed to recognize visual patterns by building increasingly complex representations at each layer of the network, illustrating a primitive form of part-whole hierarchy.

2.1.1 The evolution of neural architectures

Multi Layer Perceptrons (MLPs) [61, 111] are characterised by fully connected layers, in which each node is connected to every other possible node of the next layer. Even though they are easier to train and have simpler architecture compared to CNNs, the fully connected layers may cause the network to grow too fast in size and number of parameters, not allowing powerful scalability. MLPs have recently experienced a resurgence, thanks to patch-based approaches [61, 111], that allowed reaching state-of-the-art performances. They can also be seen as 1x1 convolutions [40, 61, 111], which do not require the pooling operation.

Convolutional Neural Networks (CNNs) [37, 98] have risen to a prominent role in computer vision when they started to outperform the existing literature in the image classification task of the ImageNet challenge [56]. The convolution operator can effectively describe spatially-correlated data resulting in a feature map, while the pooling operation down-samples the obtained feature map by summarizing the presence of certain features in patches of the image. The pooling operation in CNNs has been the subject of criticism since it does not preserve the information related to the part-whole relationship [100] between features belonging to the same object [93].

Transformers [23, 49, 66] have proven able to outperform CNNs, thanks to their ability to encode powerful features using self-attention and patch-based analysis of images. Multi-headed transformers [20] require the query, key, and value weights to be trained differently for each head, which is a costly operation. The main advantage compared to CNNs is the ability of the multiple heads to combine information from different locations in the image with fewer losses than the pooling operation [60]. However, when compared with CNNs, Transformer-like models usually require intensive pre-training on large datasets, to achieve state-of-the-art performances.

2.1.2 Capsule Networks

The concept of capsule networks proposed by Hinton et al. in 2011, marked a significant advancement in the field of neural network, particularly in the quest for improved interpretability and representation of complex data structures. These networks introduced a novel concept in neural network architecture, fundamentally different from traditional Convolutional Neural Networks (CNNs).

Unlike CNNs, which output scalar values and are known for their translational invariance, Capsule Networks operate on the principle of equivariance through vector outputs. Each capsule, a group of neurons within the network, is designed to capture and output a vector. This vector represents not only the probability of the presence of a particular feature within the data but also a set of instantiation parameters. These parameters encapsulate vital information about the feature, such as its pose, texture, and deformation, adding a layer of interpretability that was previously elusive in traditional neural network models.

One of the defining advantages of Capsule Networks is their innate ability to recognize entities by first recognizing their constituent parts, a concept inspired by cognitive theories of visual perception. This is achieved through a sophisticated structuring of capsules into layers, where lower-level capsules (primary capsules) extract pose parameters from pixel intensities, initiating a part-whole hierarchy. These primary capsules then make predictions for higher-level capsules (secondary capsules), based on the spatial relationships of the features they represent. For example, the detection of eyes and mouth as individual features by lower-level capsules can collectively activate a higher-level capsule representing a face, provided their spatial relationships align accurately.

The journey of Capsule Networks saw several significant evolutions. The first model, termed 'Transforming Auto-encoders', was designed to recognize the pose of an object in an image. The authors showed how a neural network

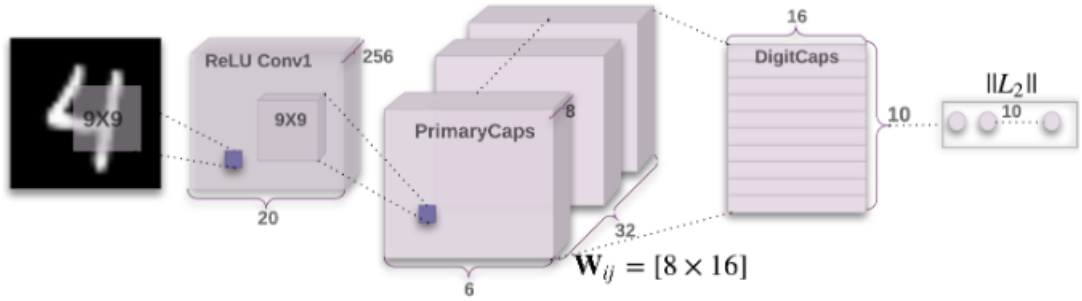


Figure 2.1: A simple capsule network with 3 layers. The PrimaryCaps are the lowest level of multi-dimensional neuronal activity (each capsule is an 8D vector). The length of the activity vector of each capsule in the DigitCaps layer indicates the presence of each class. Figure from [14].

can be used to learn features that output a vector of instantiation parameters (capsule), and argued that this is a better way of dealing with tranformations of the input. This implementation laid the groundwork for subsequent models that focused on enhancing the part-whole relationship recognition.

Sabour et al. in 2017 further refined Capsule Networks by introducing a dynamic routing mechanism between capsules. This architecture, shown in Figure 2.1 overcame the need for pose data as input, instead utilizing instantiation parameters represented by activity vectors. The dynamic routing algorithm was a critical advancement, enabling the network to learn the spatial relationships and hierarchies more effectively.

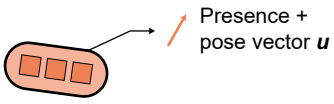


Figure 2.2: Structure of a capsule as described in [93], developed to allow dynamic routing. Classic CNNs scalar-output feature detectors are replaced by vector-output capsules. Each capsule describes both pose and presence of an entity.

Hinton et al. in 2018 [43] introduced another variant of Capsule Networks, utilizing matrix outputs instead of vectors. This approach was aimed at reducing the complexity of transformation matrices between capsules and introduced

the use of expectation-maximization routing, replacing the dynamic routing by agreement. This is achieved through a more complex capsule structure (Fig. 2.3) and an Expectation-Maximization routing (*EM-routing*) for capsules. Unfortunately, the EM-routing and the 4×4 pose matrix embedded in the capsule contribute to increasing the training time, when compared to both CNNs and [93].

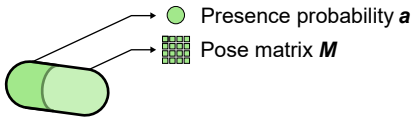


Figure 2.3: Structure of capsule as described in [43]. It contains both a scalar value and a 4X4 matrix, respectively describing the presence probability and a more robust 3D pose compared to the first capsule implementation.

Another important step in the developement of capsule network is presented by [54] by Kosiorek et al., introducing for the first time an unsupervised capsule-based autoencoder. Following this, Ribeiro et al. expanded on EM-routing with a novel approach, using VB capsule routing for the first time to fit a mixture of transforming Gaussians [91]. Their method present state-of-the-art results on smallNORB by using $\sim 50\%$ less capsules. This breakthrough opens new avenues for performance improvement and network simplification. Subsequent studies have delved into minimizing the complexity of capsule networks via quaternions [82], also enhancing their efficacy. Despite these innovations, benchmarks have predominantly focused on smaller datasets.

Capsule Networks represent a significant advacement towards more interpretable and efficient neural networks. By focusing on the representation of part-whole hierarchies and spatial relationships, they offer a more complete understanding of the data they process.

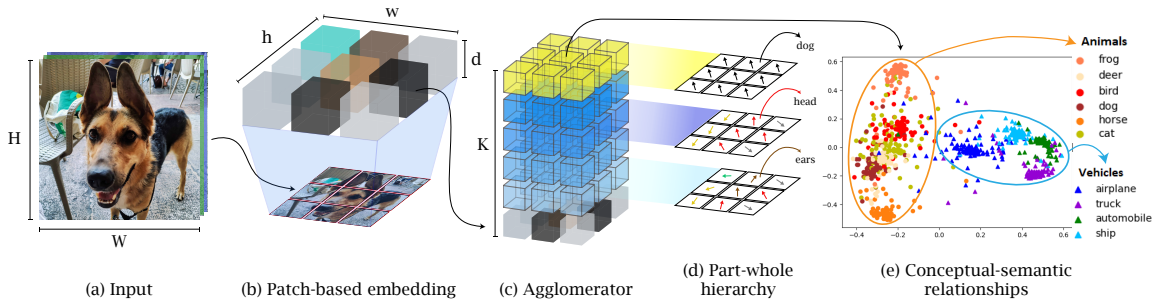
There has been a recent push toward the so-called biologically inspired Artificial Intelligence (AI) [34, 44], which tries to build deep learning networks able to mimic the structure and functions of the human brain. In [34], the au-

thors propose a column-like structure, similar to hyper-columns typical of the human neocortex. In [113], the authors build upon cortical columns implemented as separate neural networks called Cortical Column Networks (CCN). Their framework aims at representing part-whole relationships in scenes to learn object-centric representations for classification.

The author in [40] proposes a conceptual framework, called GLOM, based on inter-connected columns, each of which is connected to a patch of the image and is composed of auto-encoders stacked in levels. Weights sharing among MLP-based [61] auto-encoders allows for an easily trainable architecture with fewer weights, while knowledge distillation [41] allows for a reduction of the training parameters. The patch-based approach combined with the spatial distribution of columns allows for a sort of positional encoding and viewpoint estimation similarly to what is used in *neural fields* [71, 100]. At training time, the author recommends that GLOM should be trained using a contrastive loss function [14]. This procedure, combined with a Transformer-like self-attention [114] mechanism on each layer of the columns, aims at reaching a consensus between columns. Routing the information with layer-based attention and stacked autoencoders would theoretically allow GLOM to learn a different level of abstraction of the input at a different location and level in the columns, creating a part-whole structure with a richer representation if compared to capsule networks [93].

2.2 Agglomerator

The rapid increase in the adoption of neural networks and machine learning models has raised concerns over our ability to decipher their decision-making processes. Particularly in critical applications such as autonomous driving [30], healthcare[74], and finance [96], where stakes involve safety, life, and security, the need for neural networks to be interpretable is paramount. In fact, their



(a) Input (b) Patch-based embedding (c) Agglomerator (d) Part-whole hierarchy (e) Conceptual-semantic relationships

Figure 2.4: **[Better seen in color]**. Overview of the proposed solution. Our Agglomerator is a novel architecture for vision applications, in which column structure (c) mimics hyper-columns typical of the human visual cortex [34]. The input data (a) is fed to the columns using a patch-based embedding (b). The Agglomerator architecture iteratively routes the information across its structure, creating a neural representation of each image, similar to neural fields [71]. In the neural representation, *part-whole* hierarchies (d) emerge at different levels of the columns. The same column can represent the same patch of the image with different levels of abstraction (e.g., the ears, the head, and the dog) corresponding to each level in the column. Neighbor columns agree on a *part* representation (e.g ears, head) at lower levels, ideally representing the same *whole* (e.g. dog) at the top level. The resulting feature space represents the *conceptual-semantic relationships* between data (e) resembling the human hierarchical organization [72]. Samples belonging to the same super-class (e.g., animals, vehicles) are clustered together, with conceptually close categories (e.g., birds and airplanes) represented on the edge of the super-classes.

outstanding abilities come at a cost of model complexity, making it difficult to interpret how neural networks work [64]. Often deployed as ”black boxes,” these networks require the fine-tuning of millions of parameters, largely based on trial and error. Determining the effect of specific trainable parameters on the output for any given input is nearly infeasible.

The literature describes interpretability as ”the extent to which a human can grasp the rationale behind a decision” [73]. This concept becomes crucial when a machine learning model, tasked with classification or prediction, achieves high accuracy. The question arises: Can we trust the model without understanding why such a decision has been taken? The decision process is complex and we tend to evaluate the performance of a system in solving a given task using metrics computed at the end of the processing chain. While single metrics, such as the classification accuracy, reach super-human results, they provide an incomplete description of the real-world task [22]. For example, when dealing with an image classification problem, the learning model might tell the class the represented object belongs to. In this case we can obtain a prediction on **what** the network assigned the image to, but we have little understanding about **why** we it made such a prediction [75]. Humans, by contrast, use reasoning and intuition to associate parts with their wholes, drawing on experiences and cognitive frameworks to make inferences—even about unfamiliar animals—based on visual cues and hierarchical object organization [1, 8, 34, 72]. We would like neural networks to display a similar behavior, so that objects that are *close* in the conceptual-semantic and lexical relations are adjacent in the feature space as well (as shown in Fig. 2.4e). By doing so, it would be intuitive to identify hierarchical relations between samples and how the model has learned to build a topology describing each sample. Consequently, we can agree on the definition of interpretability in deep learning as the *“extraction of relevant knowledge from a machine-learning model concerning relationships either contained in data or learned by the model”* [79].

The landscape of image classification has been revolutionized by methods such as transformers [20, 23, 114], neural fields [71], contrastive learning representation [14], distillation [41] and capsules [93], each bringing forward breakthroughs like powerful attention-based features and per-patch analysis, positional encoding, similarity-based self-supervised pre-training, model compression and deep modeling of part-whole relationships. Despite their individual contributions to improving network interpretability, these methods often fall short in fully articulating both the data-centric relationships (such as conceptual-semantic ties [14, 20, 23, 71, 114]) and the intricacies of relationships learned by the models (like part-whole dynamics [41, 93]).
Addressing this, the conceptual framework of GLOM [40] integrates these diverse technologies, aiming to emulate the human process of parsing visual inputs into coherent structures. GLOM aims at mimicking the human ability in learning to parse visual scenes. Drawing inspiration from this theoretical concept, also described in [34], we developed the Agglomerator system, which achieves part-whole agreement [42] at different levels of the model (*relationships learned by the model*) and hierarchical organization of the feature space (*relationships contained in data*), as shown in Fig. 2.4.

Our contribution is summarised as follows:

- we introduce a novel model, called Agglomerator, mimicking the functioning of the cortical columns in the human brain [35];
- we explain how our architecture provides interpretability of *relationships learned by the model*, specifically part-whole relationships;
- we show how our architecture provides interpretability of *relationships contained in data*, namely the hierarchical organization of the feature space;
- we provide results outperforming or on par with current methods on multiple common datasets, such as SmallNORB [59], MNIST [58], Fashion-MNIST [120], CIFAR-10 and CIFAR-100 [55];

- we show that our model relies on fewer parameters and can generalize to multiple datasets.

2.2.1 Method

The framework we propose aims at replicating the column-like pattern, similar to hyper-columns typical of the human visual cortex [34]. An overview is shown in Fig. 2.4.

Agglomerator brings together concepts and building blocks from multiple methods, such as CNNs [61], transformers [20, 23, 114], neural fields [71], contrastive learning representation [14], distillation [41], and capsules [93].

Differently from the transformer architecture Agglomerator is a recurrent architecture, organized in different hierarchical levels, representing different levels of granularity of the parts. In the next paragraph, we introduce the mathematical notation needed to explain the details of the main building blocks of the architecture.

Each input image is transformed into a feature map divided into $N = h \times w$ patches. The n-th patch, with $n \in \{1,\dots,N\}$ is fed to the corresponding column $C_n(h,w)$, spatially located at coordinates (h,w). The subscript n is omitted in the next equations for better readability. As shown in Fig. 2.5, each column $C(h,w)$ consists of K embedding levels $\{l_t^{(h,w),k} \mid k = 0,\dots,K\}$ connected by a stack of auto-encoders at location (h,w) at time $t \in \{0,\dots,t-1,t,t+1,\dots,T\}$. The superscript (h,w) is omitted in the next instances of l_t^k for better readability. Each level l_t^k of the column is an embedding vector representation of size d. Levels l_t^{k-1} and l_t^k represent consecutive levels; l_t^{k-1} represents a *part* of the *whole* l_t^k. We indicate as $l_t^k \in L_t^k$ all the levels l_t^k in all columns $C(h,w)$ sharing the same k value and belonging to the same layer L_t^k. Being K the last layer of our architecture at the last time step T, it is represented as L_T^K.

Patches embedding

At the embedding stage, as in [61], we apply a convolutional Tokenizer to extract the feature map of each image of size $H \times W$ pixels, which provides a richer representation compared to the original image. Following the implementation in [61], the obtained feature map has size $h \times w \times d$ where $h = H/4$ and $w = W/4$. We then embed each of the n d-dimensional embedding vectors into the bottom levels $l_t^0 \in L_t^0$ at the corresponding coordinates (h,w) of the corresponding column $C(h,w)$. Feeding the n-th each patch to a spatially located column $C(h,w)$ resembles the positional encoding of neural fields [71], where each d-sized embedding l_t^k represents at the same time the sample and its relative observation viewpoint. At each time step t, we embed each image sample into the first layer of the columns, which is represented as the bottom layer L_t^0.

Hypercolumns

Consecutive levels in time and space in a column $C(h,w)$ are connected by an auto-encoder. The auto-encoders are based on an MLP, which allows for model reduction [41] and faster training time. Each auto-encoder computes the top-down contribution of a level l_{t-1}^{k+1} to the value of the level below at the next time step l_t^k using a $N_{TD}(l_{t-1}^{k+1})$ top-down decoder. Similarly, each auto-encoder computes the bottom-up contribution of a level l_{t-1}^{k-1} to the value of the level above at the next time step l_t^k using a $N_{BU}(l_{t-1}^{k-1})$ bottom-up encoder. $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$ share a similar structure, but for the activation functions, as described in Fig. 2.5(e). The top-down network uses GELU activation functions [38], while the bottom up network relies on Siren activation functions [99]. All the $N_{TD}(l_{t-1}^{k+1})$ connecting L_{t-1}^{k+1} to layer L_t^k share the same weights. The same is true for the $N_{BU}(l_{t-1}^{k-1})$ connecting L_{t-1}^{k-1} to layer L_t^k.

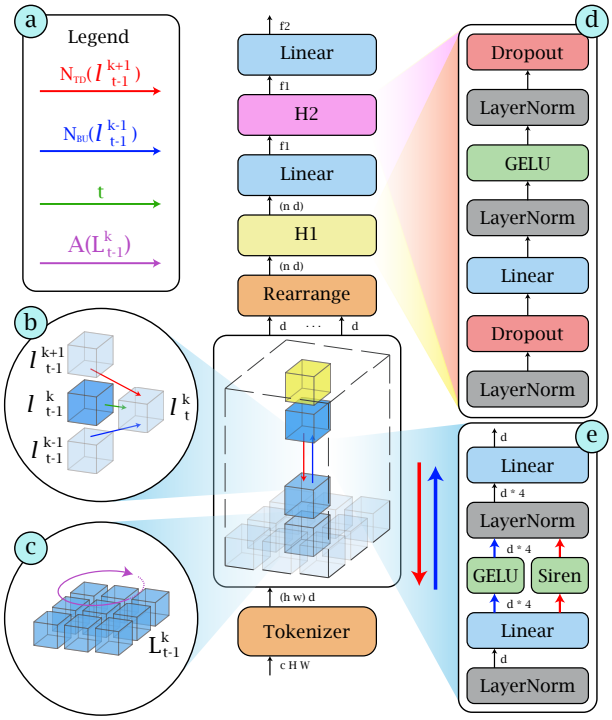


Figure 2.5: **[Better seen in color]**. Architecture of our Agglomerator model (center) with information routing (left) and detailed structure of building elements (right). Each *cube* represents a level l_t^k. **Left:** (a) legend of the arrows in the figure, representing the top-down network $N_{TD}(l_{t-1}^{k+1})$, the bottom-up network $N_{BU}(l_{t-1}^{k-1})$, attention mechanism $A(L_{t-1}^k)$ and time step t. (b) Contribution to the value of level l_t^k given by l_{t-1}^k, $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$. (c) The attention mechanisms $A(L_{t-1}^k)$ share information between $l_{t-1}^k \in L_{t-1}^k$. **Center:** bottom to top, the architecture consists of the Tokenizer module, followed by the columns $C(h,w)$, with each level l_t^k connected to the neighbors with $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$. On top of the structure, the contrastive $H1$ and cross entropy $H2$ heads. **Right:** (d) structure of heads $H1$ and $H2$. (e) Structure of the top-down network $N_{TD}(l_{t-1}^{k+1})$ and the bottom-up network $N_{BU}(l_{t-1}^{k-1})$.

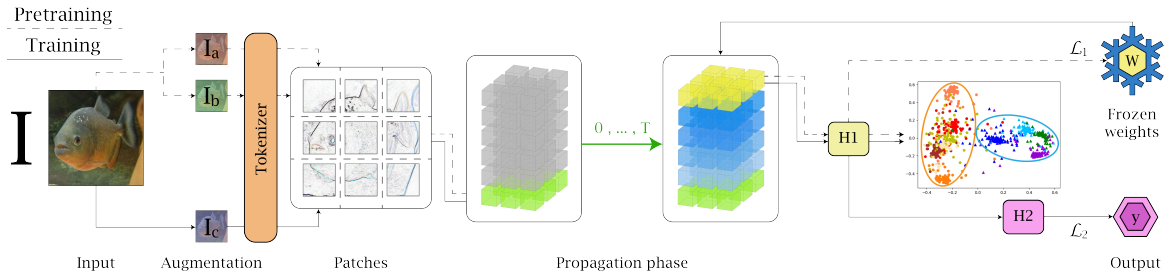


Figure 2.6: **Contrastive pre-training (dashed lines)** and **supervised training (continuous lines)** procedures. During the contrastive pre-training, two images I_a and I_b are produced by applying random data augmentation to the input image I. Through the Tokenizer, we compute feature maps for both I_a and I_b, which are then divided in patches and embedded into the bottom layer of the columns L_t^0. During the *propagation phase*, the information is routed through the Agglomerator architecture to obtain the neural representation L_T^K for each sample. We pre-train the network with the contrastive head $H1$ using a supervised contrastive loss $\mathcal{L}_1$, obtaining weights W. During the supervised training, we first load the frozen weights W in the network. Then, augmentation RandAugment [18] is applied on the input image I to obtain I_c, which follows the same steps as the pre-training phase. The network, with the classification head $H2$, is trained for the classification task by minimizing the cross-entropy loss $\mathcal{L}_2$.

Routing

The key element of our architecture is how the information is routed to obtain a representation of the input data where the part-whole hierarchies emerge.

Before computing the loss, we need to iteratively propagate each batch N through the network, obtaining a deep representation of each image. This procedure, *propagation phase*, encourages the network to reach *consensus* between neighbor levels $l_t^k \in L_t^k$. Ideally, this means that all neighbor levels in the last layer $l_t^K \in L_t^K$ should have similar values, representing the same *whole*; neighbor levels at bottom layers $l_t^k \in L_t^k | k \neq K$ should instead share the value among smaller groups, each group representing the same *part*. Group of vectors that "agree" on a similar value have reached the *consensus* on the image representation at that level, and they are called *islands of agreement*. An example of such representation is shown in Fig. 2.4(d). In capsules-based approaches [93], group of neurons are activated to represent the part-whole hierarchy with limited

expressive power. Our d-dimensional layers l_t^k provide a richer representation of the same hierarchy.

To obtain such representation, at time step $t = 0$, we randomly initialise all the values l_0^k and we embed a batch of B samples into the bottom layer L_0^0. Once the values are initialized, we compute the attention $A(L_t^k)$. Instead of the self-attention mechanism used in Transformers [20, 23, 114], a standard attention weighting is deployed as in [123]. Each attention weight ω_n is computed as

$$\omega_n = \frac{e^{\beta \lambda_n \cdot l_t^k}}{\sum e^{\beta \lambda_n \cdot N(\lambda_n)}} \tag{2.1}$$

where λ_n represents each possible level l_t^k belonging to the same layer L_t^k as l_t^k, $N(\lambda_n)$ is an indicator function which indexes all the neighbors levels of λ_n belonging to the same layer L_t^k and β is a parameter that determines the sharpness of the attention.

At each time step $t \mid t \in \{1, \ldots, T\}$, a batch with B samples is fed to the bottom layer L_t^0 network as described in Sec. 2.2.1. We compute the values l_t^k as

$$\begin{aligned} l_t^k = avg(&\omega_l l_{t-1}^k, \omega_{BU} N_{BU}(l_{t-1}^{k-1}), \\ &\omega_{TD} N_{TD}(l_{t-1}^{k+1}), \omega_A A(L_{t-1}^k)) \end{aligned} \tag{2.2}$$

where $avg()$ indicates the arithmetical average, and $\omega_l, \omega_{BU}, \omega_{TD}, \omega_A$ are trainable weights. For layer L_t^K, contribution $N_{TD}(l_{t-1}^{k+1})$ is not included, as L_t^{K+1} does not exist. The *propagation phase* takes T time steps to reach the final representation of each image at each layer L_k^T.

Training

The training procedure of our architecture is shown in Fig. 2.6. It is divided in two steps: (i) a pre-training phase using a supervised contrastive loss function

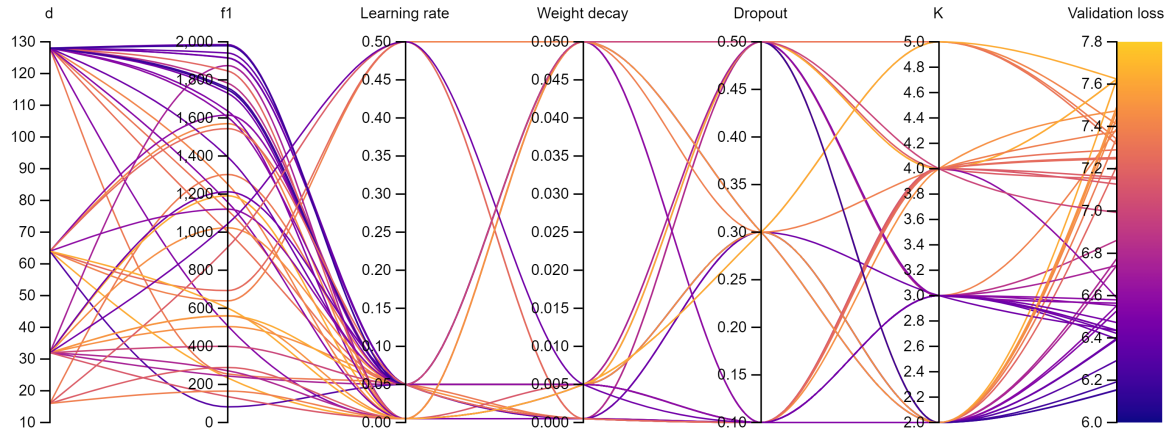


Figure 2.7: **Agglomerator hyper-parameters sweep**. Hyper-parameters sweep. Each line represents a combination of parameters setup, with the darker lines representing the models achieving the lowest validation loss. Image obtained with [9].

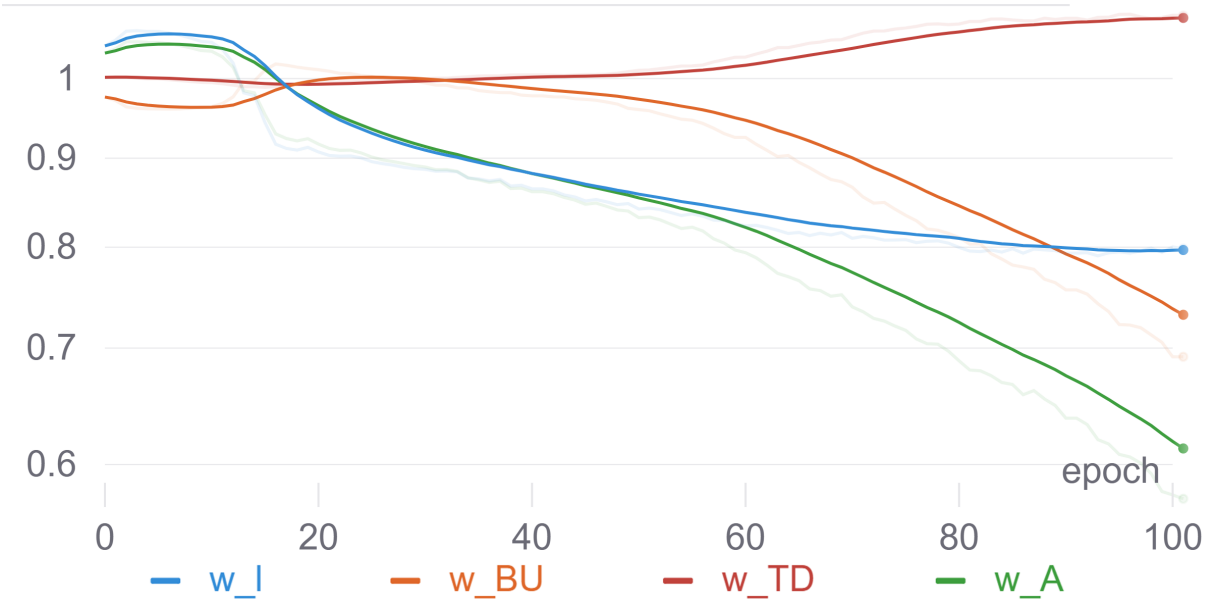


Figure 2.8: **Agglomerator weights balancing**. Each line represents the variation of weights $\omega_l, \omega_{BU}, \omega_{TD}, \omega_A$ across epochs. Image obtained with [9].

[14] and (ii) a training phase for the image classification using a Cross-Entropy loss.

We first pre-train our network using an image-based contrastive loss [14]. Given a batch with B samples, we duplicate each image I to obtain pairs of samples (I_a, I_b), for a total of $2B$ data points. We then apply data augmentation RandAugment [18] to both (I_a, I_b). Both samples are fed to the network as described in Sec. 2.2.1, and we perform the propagation phase in Sec. 2.2.1 to obtain the representation at the last layer L_T^K. Then we rearrange the n levels $l_T^K \in L_T^K$ to obtain a vector of dimensions $n \times d$, given as input to the contrastive head $H1$, as described in Fig. 2.5. At the output of the contrastive head, each sample is described by a feature vector of dimension $f1$. We take all the possible sample pairs (I_a, I_b) from the batch and we compute the contrastive loss defined as:

$$\mathcal{L}_1 = ContrLoss(I_a, I_b) = -log\frac{e^{sim(I_a,I_b)}}{\sum_{k=1}^{2B} \mathcal{I}_{[k\neq a]} e^{sim(I_a,I_b)}} \tag{2.3}$$

where $\sim (u,v) = \frac{u^T v}{\|u\|\|v\|}$ indicates the dot product between the normalized versions of u and v, $\mathcal{I}_{[k\neq a]}$ is an indicator function valued 0 if k and a belong to the same class, and 1 otherwise.

Once the network is pre-trained using the contrastive loss, the weights are frozen. We apply augmentation [18] to each sample I_c in a batch of size B, which is then fed to the network for the *propagation phase* to obtain for each sample the representation L_T^K. Then, the cross-entropy head $H2$ is added on top of the contrastive head $H1$. A linear layer resizes $f1$-dimensional features to dimension $f2$, which corresponds to the number of classes to be predicted for each dataset. The new layers are then trained using the cross-entropy function:

$$\mathcal{L}_2 = CE(y, \hat{y}) = -\frac{1}{f2}\sum_{i=1}^{\text{f2}} y_i \log(\hat{y}_i) \tag{2.4}$$

where y is the label of a sample taken from the batch and $\hat{y}$ is the label to be predicted.

2.2.2 Experiments

We perform our experiments on the following datasets:

SmallNorb (S-NORB) [59] is a dataset for 3D object recognition from shape. It consists of roughly 200000 images of size 96×96 pixels of 5 classes of toys.

MNIST [58] and **FashionMNIST** [120], consist of 60000 training images and 10000 test images of grayscale handwritten digits and Zalando's articles of size 28×28 pixels.

CIFAR-10 and **CIFAR-100** [55] both consist of 50000 training images and 10000 test images of size 32×32 pixels, with 10 and 100 classes, respectively.

Our network is trained in an end-to-end fashion using PyTorch Lightning on a single NVIDIA GeForce RTX 3090. Input images for each dataset are normalized using each standard dataset's normalization. We train our network on each dataset's native resolution, except for SmallNorb, which is resized to 32×32 pixels, following the standard procedure as in [43, 91]. The Tokenizer embedding creates $n = H/4 \times W/4$ patches, thus the corresponding number of columns is 8×8 for CIFAR-10, CIFAR-100, and SmallNorb, and 7×7 for MNIST FashionMNIST. During the pre-training, we deploy the following hyper-parameters: 300 epochs, cyclic learning rate [101] in the range $[0.002, 0.05]$, batch size $B = 1024$, levels embedding $d = 128$, number of levels $K = 3$, number of iterations $T = 2K = 6$, dropout value 0.3, contrastive features dimension $f1 = 512$, and weight decay $5e^{-4}$. During the training phase, we resume the network training with the same hyper-parameters, $f2$ being the number of classes corresponding to each dataset.

Method	Ref	Backbone	Error %					# of params (Millions)	Training Arch.
			S-Norb	MNIST	F-MNIST	C-10	C-100		
E-CapsNet	[70]	Caps	2.54	0.26	-	-	-	0.2	GPU
CapsNet	[77, 93]		2.70	0.25	6.38	10.6	82.00	6.8	GPU
Matrix-CapsNet	[43]		1.40	0.44	6.14	11.9	-	0.3	GPU
Capsule VB	[91]		1.60	0.30	5.20	11.2	-	0.2	GPU
ResNet-110	[5, 37, 45]	Conv	-	2.10	5.10	6.41*	27.76*	1.7	GPU
VGG	[5, 98]		-	0.32	6.50	7.74*	28.05*	20	GPU
ViT-L/16	[23]	Transf	-	-	-	0.85*	6.75*	632	TPU
ConvMLP-L	[61]	Conv/MLP	-	-	-	1.40*	11.40*	43	TPU
MLP-Mixer-L/16	[111]	MLP	-	-	-	1.66*	-	207	TPU
Ours		Conv/MLP/Caps	0.01	0.30	7.43	11.15	40.97	72	GPU

Table 2.1: Error percentages on the Top-1 accuracy results on datasets SmallNorb (S-Norb), MNIST, FashionMNIST (F-MNIST), CIFAR-10 (C-10), and CIFAR-100 (C-100). The $*$ notation indicates results obtained with networks pre-trained on ImageNet.

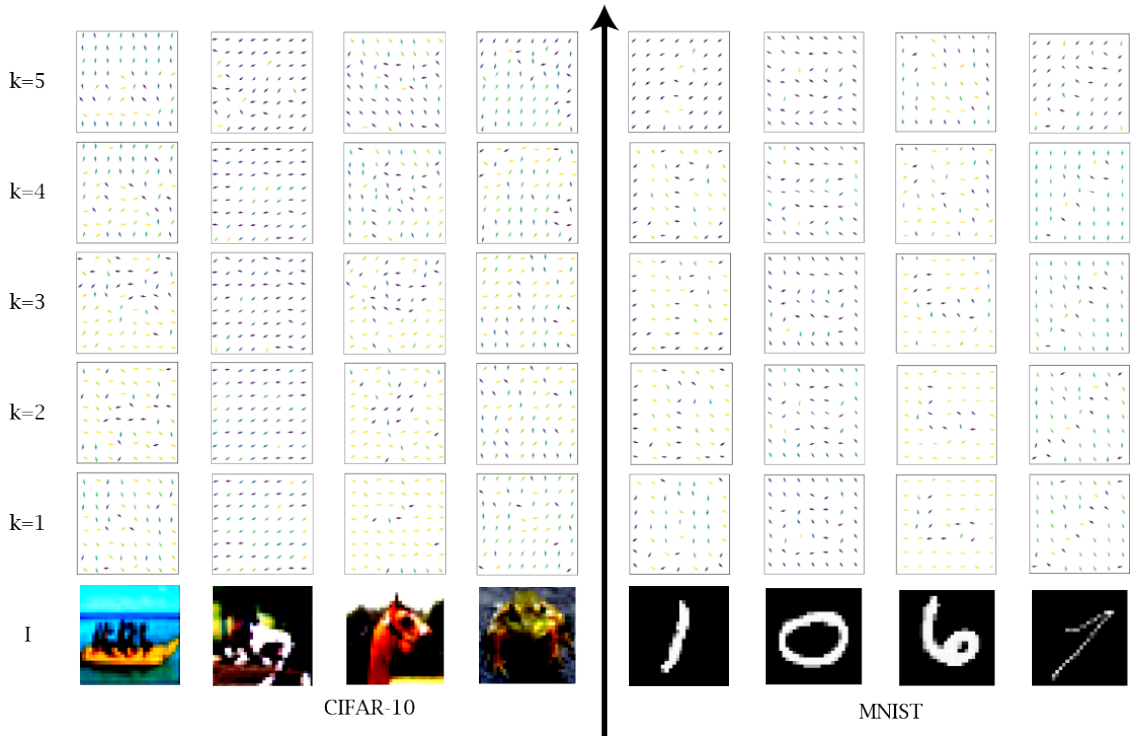


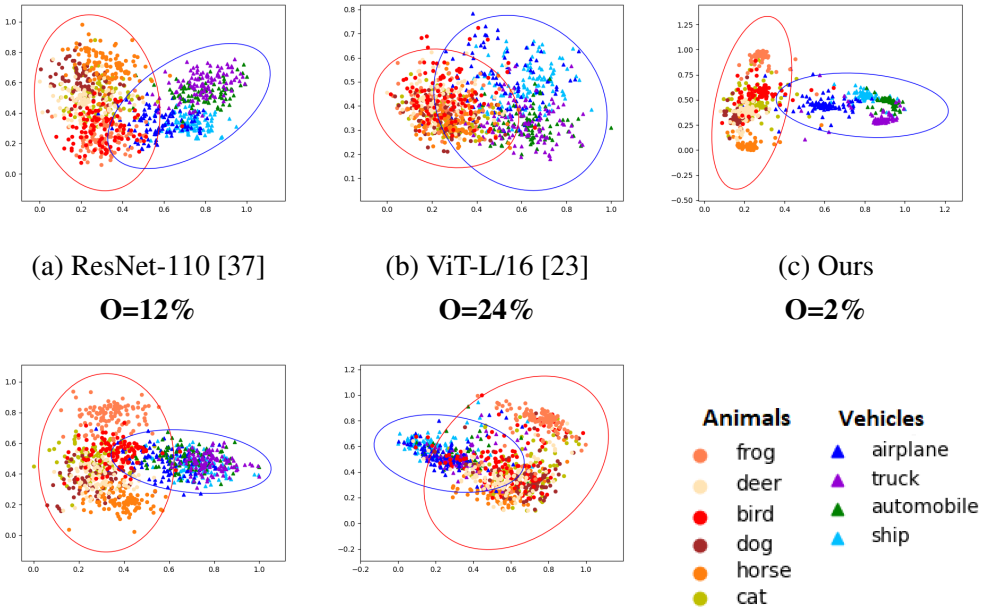
Figure 2.9: Vectorial representation of emerging islands of agreement at different K levels of sample from MNIST and CIFAR-10 datasets.

2.2.3 Quantitative results

We report the quantitative results for each dataset in Tab. 2.1. Capsule-based models [43, 70, 77, 91, 93] can achieve good performances on simple datasets (SmallNorb, MNIST, and FashionMNIST), but they fail to generalize to datasets with a higher number of classes (CIFAR-100). Convolutional-based models [5, 37, 45, 98] can generalize to different datasets, at the expense of weak model interpretability, mainly due to the max-pooling operation. Transformer-based [23] and MLP-based methods [61, 111] are able to achieve the best performances on more complex datasets, but they do not provide tests for smaller datasets. However, to achieve such levels of accuracy they rely on long pre-training (thousands of TPU days) on expensive computational architectures, implementing data augmentation on ImageNet [56] or the JFT-300M [104] dataset, not available publicly. As can be seen, our method performs on par with capsule-based methods on simpler datasets, while achieving better generalization on more complex ones. In addition, our method has fewer parameters than most transformer-based and MLP-based methods, and it can be trained in less time on a much smaller architecture.

Ablation study. We analyze the contribution of the different components of our architecture evaluating their influence on the validation loss. The considered parameters, in descending order of correlation with the validation loss value are: the embedding dimension d, the contrastive feature vector $f1$, learning rate, weight decay, dropout, and the number of levels K. The results are reported in Fig. 2.7. We perform 50 different training on CIFAR-10 with different combinations of parameters. We monitor the validation loss value after 50 epochs.

In Fig. 2.8, we show how the contributions to the values of levels l_t^k in Eq. 2.2 are weighted over the first 100 epochs by observing how the trainable weights $\omega_l, \omega_{BU}, \omega_{TD}, \omega_A$ change. The attention contribution ω_A and the previ-



(a) ResNet-110 [37] **O=12%**

(b) ViT-L/16 [23] **O=24%**

(c) Ours **O=2%**

(d) ConvMLP-L [61] **O=12%**

(e) Matrix-CapsNet [43] **O=20%**

(f) Legend

Figure 2.10: 2D representation of the latent space for multiple methods trained only on the CIFAR-10 dataset obtained using Principal Component Analysis (PCA) [117]. The PCA provides a deterministic change of base for the data from a multidimensional space into a 2D space. The legend (f) displays the classes, which are divided between super-classes *Vehicles* and *Animals* following the WordNet hierarchy [72]. The different methods (a,b,c,d,e) are all able to cluster the samples between the two super-classes. However, while (a,b,e) display a latent space where classes are close to each other, the two MLP-based methods (c,d) are able to provide a clearer separation between the super-classes. Both methods show conceptual-semantically close samples on the edge of each superclass, such as airplanes and birds. Inside each superclass, semantically close samples are represented contiguously, such as deers and horses, or cars and trucks. Our method (c) provides better inter-class and intra-class separability. The overlap percentage O is reported for each method. The overlap area is the area where a mistake with a higher hierarchical severity [7] has a higher probability to occur.

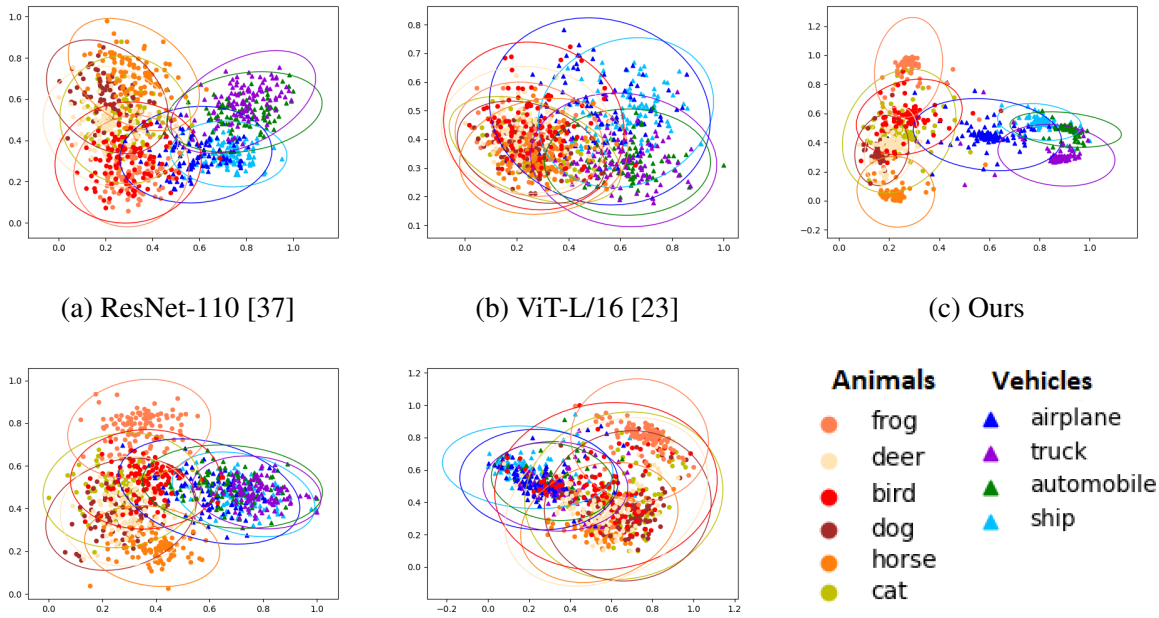
ous level one ω_l decrease over time, with ω_l reaching a plateau. After 15 epochs of weights self-calibration, the top-down contribution ω_{TD} increases over the epochs, as the network can infer useful information from predicted "wholes" at the upper levels. The bottom-up contribution ω_{BU} decreases over the epochs since the bottom-up networks have learned to map efficiently map the input into the embeddings, especially at lower levels.

2.2.4 Qualitative results: interpretability

Our method provides interpretability of the *relationships learned by the model* by explicitly modeling the part-whole hierarchy, and of the *relationships contained in data* through the hierarchical organization of the feature space.

Island of agreement as a representation of multi-level part-whole hierarchy. During the *propagation phase*, neighbor levels on the same layer L_t^k are encouraged to reach a *consensus* by forming *islands of agreement*. The *islands of agreement* represent the part-whole hierarchies at different levels. In Fig. 2.9, we provide a few examples of the islands of agreement obtained on MNIST and CIFAR-10 trained with $K = 5$ levels. Each arrow represents the value of a level l_k^t at location (h, w), reduced from d-dimensional to 2D using a linear layer. As k for L_t^k increases, neighbor $l_k^t \in L_t^k$ tend to agree on a common representation of the *whole* represented in the image sample. At lower levels, smaller islands emerge, each representing a part of the *whole*. Samples of MNIST present fewer changes in the islands across levels because the data is much simpler, indicating that fewer levels in the hierarchy can be sufficient to obtain similar results. Our Agglomerator is thus able to represent a patch differently at different levels of abstraction. At the same level, spatially adjacent patches take the same value, agreeing on the representation of parts and wholes.

Latent space organization as the representation of conceptual-semantic relationship in data. Recent networks aim at maximizing inter-class distances and minimizing intra-class distances between samples in the latent space. While



(a) ResNet-110 [37] (b) ViT-L/16 [23] (c) Ours

(d) ConvMLP-L [61] (e) Matrix-CapsNet [43] (f) Legend

Figure 2.11: 2D representation of the latent space for multiple methods trained only on the CIFAR-10 dataset obtained using Principal Component Analysis (PCA) [117]. The PCA provides a deterministic change of base for the data from a multidimensional space into a 2D space. The legend (f) displays the classes, which are divided between super-classes *Vehicles* and *Animals* following the WordNet hierarchy [72]. The different methods (a,b,c,d,e) are all able to cluster the samples. However, while (a,b,e) display a latent space where classes are close to each other, the two MLP-based methods (c,d) are able to provide a clearer separation between classes. Both methods show conceptual-semantically close samples on the edge of each superclass, such as airplanes and birds. Inside each superclass, semantically close samples are represented contiguously, such as deers and horses, or cars and trucks. Our method (c) provides better inter-class and intra-class separability. We provide numerical results of the classes overlap in Fig. 2.12.

ResNet-110	Frog	Deer	Bird	Dog	Horse	Cat	Airplane	Truck	Automobile	Ship
Frog	100	32	70	12	3	30	24	0	0	3
Deer	32	100	42	50	22	73	14	1	1	0
Bird	70	42	100	18	7	39	27	3	3	6
Dog	12	50	18	100	39	52	5	0	0	0
Horse	3	22	7	39	100	33	4	4	4	0
Cat	30	73	39	52	33	100	17	3	4	2
Airplane	24	14	27	5	4	17	100	19	21	44
Truck	0	1	3	0	4	3	19	100	70	15
Automobile	0	1	3	0	4	4	21	70	100	18
Ship	3	0	6	0	0	2	44	15	18	100

(a) ResNet-110 [37]

ViT-L/16	Frog	Deer	Bird	Dog	Horse	Cat	Airplane	Truck	Automobile	Ship
Frog	100	75	55	73	59	73	25	18	13	14
Deer	75	100	71	56	53	58	30	18	13	18
Bird	55	71	100	48	56	52	39	22	16	22
Dog	73	56	48	100	65	82	20	19	16	11
Horse	59	53	56	65	100	65	27	28	24	20
Cat	73	58	52	82	65	100	24	23	20	14
Airplane	25	30	39	20	27	24	100	42	35	60
Truck	18	18	22	19	28	23	42	100	71	41
Automobile	13	13	16	16	24	20	35	71	100	35
Ship	14	18	22	11	20	14	60	41	35	100

(b) ViT-L/16 [23]

Ours	Frog	Deer	Bird	Dog	Horse	Cat	Airplane	Truck	Automobile	Ship
Frog	100	2	17	0	0	17	0	0	0	0
Deer	2	100	26	63	14	29	0	0	0	0
Bird	17	26	100	19	0	54	12	0	0	0
Dog	0	63	19	100	14	27	0	0	0	0
Horse	0	14	0	14	100	14	0	0	0	0
Cat	17	29	54	27	14	100	10	0	0	0
Airplane	0	0	12	0	0	10	100	18	11	16
Truck	0	0	0	0	0	0	18	100	19	11
Automobile	0	0	0	0	0	0	11	19	100	44
Ship	0	0	0	0	0	0	16	11	44	100

(c) Ours

ConvMLP-L	Frog	Deer	Bird	Dog	Horse	Cat	Airplane	Truck	Automobile	Ship
Frog	100	6	23	5	0	18	9	0	3	0
Deer	6	100	42	85	33	58	18	0	9	3
Bird	23	42	100	39	8	62	35	5	21	11
Dog	5	85	39	100	30	57	17	0	9	3
Horse	0	33	8	30	100	16	6	0	1	0
Cat	18	58	62	57	16	100	23	1	13	5
Airplane	9	18	35	17	6	23	100	38	61	53
Truck	0	0	5	0	0	1	38	100	59	70
Automobile	3	9	21	9	1	13	61	59	100	66
Ship	0	3	11	3	0	5	53	70	66	100

(d) ConvMLP-L [61]

Matrix-CapsNet	Frog	Deer	Bird	Dog	Horse	Cat	Airplane	Truck	Automobile	Ship
Frog	100	21	5	15	25	50	19	60	73	18
Deer	21	100	28	66	74	16	77	24	28	57
Bird	5	28	100	25	33	6	31	6	8	10
Dog	15	66	25	100	58	10	75	16	21	54
Horse	25	74	33	58	100	22	76	31	32	44
Cat	50	16	6	10	22	100	15	69	49	9
Airplane	19	77	31	75	76	15	100	21	25	48
Truck	60	24	6	16	31	69	21	100	70	18
Automobile	73	28	8	21	32	49	25	70	100	25
Ship	18	57	10	54	44	9	48	18	25	100

(e) Matrix-CapsNet [43]

Figure 2.12: The overlap percentage O between classes in the latent space is reported for each possible class and each method. For each table, the top-left quadrant represents the overlap percentage between classes belonging to the super-class *animals*, while the bottom-right one for the superclass *vehicles*. The top-right and bottom-left quadrants represent the area where a mistake with a higher hierarchical severity [7] is possible. It would be ideal to have the table with zeros for all the values, but the diagonal. That would represent perfect separation between all the classes. Our method provides the best separation between the two superclasses. It is interesting to notice that the highest intra-superclasses correlation is present between the two classes *bird* and *airplanes* which share features like the wings and the ability to fly.

the accuracy is high, they provide little interpretability in their data representation. As a result, mistakes are less likely to happen, but the mistake severity, defined as the distance between two classes in WordNet lexical hierarchy [72], does not decrease [7]. As shown in Fig. 2.10, our network provides an organization of the input data, which is semantically closer to the human lexical hierarchy.

2.2.5 Limitations

Our novel neural network structure inherently incorporates new types of hyper-parameters such as embedding dimensions, number of levels, and size of patches, which need to be tuned. We believe a better parameters setting can be found for all the datasets, increasing accuracy while still retaining interpretability. Moreover, a higher number of parameters generally causes architectures to be more prone to over-fitting and more difficult to train. To improve the accuracy of our network, we would need a pre-training on large datasets (e.g., on ImageNet), which requires large computational resources to be performed in a reasonable time frame. While hoping that powerful TPU architectures become publicly available in the future, we are currently investigating efficient pre-training strategies for our network.

2.2.6 Discussion

We presented Agglomerator, a method that makes a step forward towards representing interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks. We believe that interpretable networks are key to the success of artificial intelligence and deep learning. With this work, we intend to promote a preliminary implementation and the corresponding results on the image classification task, and we hope to inspire other researchers to adjust our solution to solve more complex and diverse tasks.

2.3 Agglomerator++

While deep neural networks consistently outperform humans across fields like computer vision [37], natural language processing [114], and data analysis [96], the attained high performance often comes at the cost of increased complexity. An established practice is to train and evaluate the output of neural networks consisting of billions of parameters, and this is accomplished mostly following experience and the rule of thumb. Interpreting how a trainable parameter in the network setup directly affects the desired output from a given input becomes then nearly impossible [67].

Interpretability, defined as the capacity to provide understandable explanations to humans [22, 64], is crucial in many areas [30] [96], although it is generally hard to understand why a specific decision was made. Humans, on the other hand, are adept at understanding objects, their parts, and their interrelationships. We can categorize and recognize an object from its parts and infer its concept from its visual features [8]. This hierarchical representation is often missing in deep learning networks.

Various techniques have been introduced in the image classification field, such as transformers [23, 114], neural fields [71], contrastive learning representation [14], distillation [41], and capsules [93]. These methods have improved interpretability to a certain extent. However, they often lack emphasis on data relationships or model-learned relationships, such as part-whole hierarchies. Inspired by the GLOM framework [40], our Agglomerator [24] aims to address these shortcomings. It integrates multiple methods, such as CNNs [61], transformers, and positional encoding [23, 114], contrastive learning representation [14], distillation [41], and capsules [93].

Agglomerator++ is an evolution of our previous contribution, Agglomerator [24], which had proven effective for representing part-whole hierarchies while dealing with the image classification task. Agglomerator++ shares a similar

structure but improves the results while reducing the model size. Adopting a similar technique as in [71], we introduced the positional encoding and we adopt a training procedure with masked patches as performed in [36, 122]. The sample latent representation after the masked pre-raining allows the architecture to achieve better performances in the classification stage than the previously adopted contrastive pre-training, at a smaller computational cost. We perform an extensive evaluation of Agglomerator++, focusing on the hyperparameters that are peculiar to our architecture and cannot be found in different neural networks, such as the number of levels and column structure (details in the coming paragraphs). Experimental results show that our model can compete with much larger models on big datasets while retaining a small number of parameters and outperforming capsule networks on smaller ones.

Our contribution is summarised as follows:

- we introduce a novel model, called Agglomerator++, mimicking the functioning of the cortical columns in the human brain [34];

- we show how our architecture provides interpretability of *relationships contained in data*, namely the hierarchical organization of the feature space closely resembling human lexical similarities [72];

- we provide results outperforming or on par with current methods on common datasets, such as SmallNORB [59], MNIST [58], FashionMNIST [120], CIFAR-10 and CIFAR-100 [55];

- we show how the input masking during the pre-training for self-supervised reconstruction leads to a better, more efficient neural representation than the previously deployed contrastive pre-training [24].

2.3.1 Method

The framework we propose aims at replicating a column-like pattern, similar to hyper-columns typical of the human visual cortex [34]. The proposed network, called *Agglomerator++*, approaches the classification task in a patch-based fashion, constructing on top of each patch so-called columns. Each column consists of several layers which, from bottom to top, progressively agree according to a part-whole paradigm, to the definition of the image content.

Compared to the previous solution proposed in Agglomerator [24], the method also benefits from an unsupervised pre-training [33], setting up the network as an autoencoder to reconstruct the masked input image as in [122], which leads to a better neural representation. This procedure, combined with a simplified version of the Transformer self-attention [114] mechanism on each layer of the columns, aims at reaching a consensus between columns. Routing the information with layer-based attention and stacked autoencoders allows GLOM to learn a different level of abstraction of the input at a different location and level in the columns, creating a part-whole structure with a richer representation if compared to capsule networks [93].

Here, we introduce the mathematical notation needed to explain the details of the main building blocks of the architecture.

As can be seen in Fig. 2.14, each input image is transformed into a feature map divided into $N = h \times w$ patches. The n-th patch, with $n \in \{1, \ldots, N\}$, located at coordinates (h, w) is fed to the corresponding column $C(h, w)$, as in Fig. 2.13(c).

As shown in Fig. 2.13, each column $C(h, w)$ consists of K embedding levels $\{l_t^{(h,w),k} \mid k = 0, \ldots, K\}$ connected by a stack of auto-encoders at location (h, w) at time $t \in \{0, 1, \ldots, T\}$, as suggested in [40]. The (h, w) is omitted in the next instances of l_t^k for better readability. Each level l_t^k of the column is a vector representation of size d, which encodes the patch information at position k at

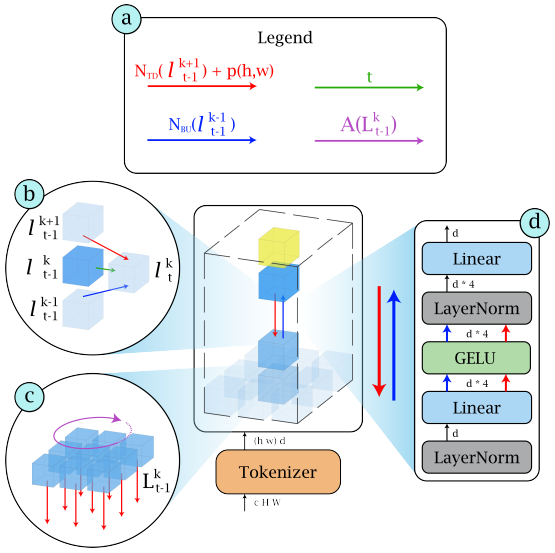


Figure 2.13: Architecture of our Agglomerator++ model (center) with information routing (left) and detailed structure of building elements (right). Each *cube* represents a level l_t^k. **Top:** (a) legend of the arrows in the figure, representing the top-down network $N_{TD}(l_{t-1}^{k+1})$ and the positional embedding $p(h,w)$, the bottom-up network $N_{BU}(l_{t-1}^{k-1})$, attention mechanism $A(L_{t-1}^k)$ and time step t. **Left:** (b) Contribution to the value of level l_t^k given by l_{t-1}^k, $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$. (c) The attention mechanism $A(L_{t-1}^k)$ shares information between $l_{t-1}^k \in L_{t-1}^k$. The positional embedding $p(h,w)$ is different for each column $C(h,w)$. All levels belonging to the same hyper-column $C(h,w)$ share the positional embedding $p(h,w)$. **Center:** bottom to top, the architecture consists of the tokenizer module, followed by the columns $C(h,w)$, with each level l_t^k connected to the neighbors with $N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$. **Right:** (d) Structure of the top-down network $N_{TD}(l_{t-1}^{k+1})$ and the bottom-up network $N_{BU}(l_{t-1}^{k-1})$.

time t; l_t^{k-1} and l_t^k represents consecutive embedding levels; l_t^{k-1} represents a *part* of the *whole* l_t^k.

We denote as the layer L_t^k, all the levels l_t^k in all columns sharing the same k. Being K the last layer of our architecture at the last time step T, it is represented as L_T^K.

Patches embedding

At the embedding stage, following the approach in [61], we utilize a convolutional tokenizer to extract the feature map of each image of size $H \times W$ pixels.

This process provides a richer representation compared to the original image. Unlike Agglomerator [24], which employs a ResNet [56] as the tokenizer for the input image, our model uses a straightforward convolutional tokenizer consisting of three 3x3 convolutional layers.

Following the implementation in [61], the obtained feature map has size $h \times w \times d$ where $h = H/H_p$ and $w = W/P_p$, H_p and W_p being the dimensions in pixels of each patch. We then embed each d-dimensional embedding vector into the bottom levels $l_t^0 \in L_t^0$ at the corresponding coordinates (h,w) of the corresponding column $C(h,w)$. Feeding the each patch to a spatially located column $C(h,w)$ resembles the positional encoding of neural fields [71], where each d-sized embedding l_t^k represents at the same time the sample and its relative observation viewpoint.

Hypercolumns

Every auto-encoder bridges consecutive levels in time and space within a column $C(h,w)$. Based on an MLP, these auto-encoders facilitate model reduction [41] and quicker training. The auto-encoder computes the top-down contribution of level l_{t-1}^{k+1} to the value of the lower level at the next time step l_t^k using a $N_{TD}(l_{t-1}^{k+1})$ top-down decoder. Likewise, it calculates the bottom-up contribution of level l_{t-1}^{k-1} to the value of the level above at the next time step l_t^k using a $N_{BU}(l_{t-1}^{k-1})$ bottom-up encoder.

$N_{TD}(l_{t-1}^{k+1})$ and $N_{BU}(l_{t-1}^{k-1})$ share a similar structure, except for the activation functions as shown in Fig. 2.13(e). Both these networks use GELU activation functions [38]. All $N_{TD}(l_{t-1}^{k+1}) \mid k = 0,\ldots,K$ networks share the same weights, as do all $N_{BU}(l_{t-1}^{k+1}) \mid k = 0,\ldots,K$ networks. This weight sharing among top-down decoders and bottom-up encoders significantly reduces the number of parameters and model size. Additionally, it promotes the part-whole hierarchy since identical networks connect embeddings at successive levels with similar hierarchical relationships (a part at level $k-1$ to its whole at level k), but varying

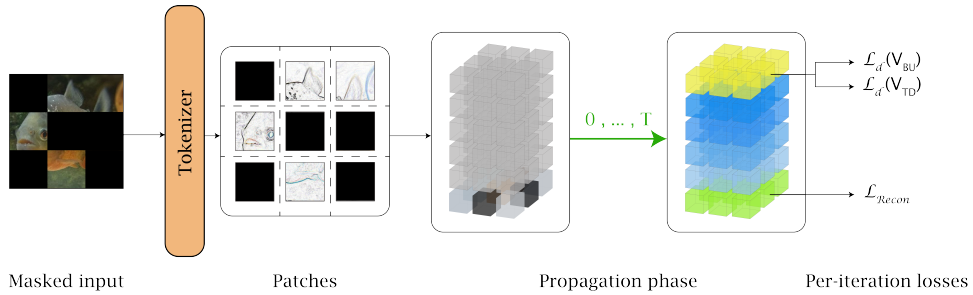


Figure 2.14: **Pre-training to obtain a rich neural representation**. During the pre-training phase, a masked input is given to the network. The reconstruction loss $\mathcal{L}_{Recon}$ depends on how well the masked patches of the input image are reconstructed. The loss is attached to level L_t^1 because the network present more detailed features at a lower level, while the representation becomes more abstract at higher levels, thus less suitable for reconstructing the input image. At the same time, enforcing the minimization of the regularisation losses $\mathcal{L}_d$ on the last level L_t^K encourages the network to display more definite *islands of agreement* at higher levels.

representation abstractions (a whole at level k is a part relative to level $k+1$). Thus, the same auto-encoder can represent part-whole relationships independently from the level of abstraction.

Routing

The key element of our architecture is the routing of information to obtain a representation of the input data where part-whole hierarchies naturally emerge. Leveraging Masked pretraining [36], the model learns a richer representation that facilitates the routing process. To propagate batch B through the network, to achieve deep image representations, we employ the *propagation phase*, fostering *consensus* between neighbor levels $l_t^k \in L_t^k$. This process yields similar values across the last layer's neighbor levels $l_t^K \in L_t^K$, each depicting the same *whole*, while lower layers share values amongst smaller groups signifying the same *part*. Vectors with similar values, or *islands of agreement*, symbolize image representation consensus at a level [40].

Attention weights. The attention $A(L_t^k)$ allows the information to be routed between embeddings belonging to the same layer. In our architecture we em-

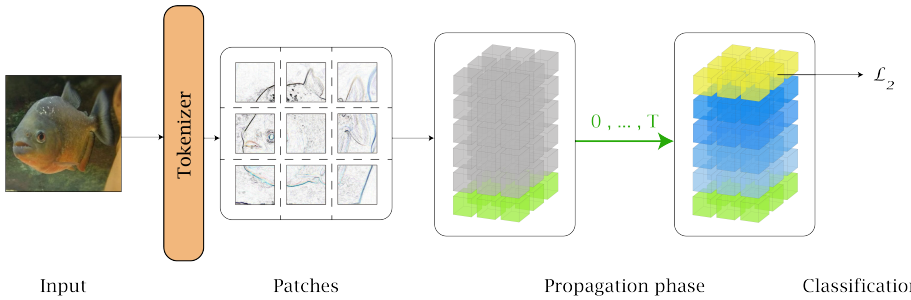


Figure 2.15: **Training for image classification**. During the training phase, image samples are fed through the network to obtain their neural representation. The information is routed for at least $2K$ iterations for the information to be propagated all along the network from the bottom level thanks to the bottom-up networks and back down thanks to the top-down nets. The classification loss $\mathcal{L}_2$ is attached to the last level because the higher-level features are more suitable for the classification task.

ploy a simple version of the attention, which is similar to a weighted average of the embedding belonging to the same levels of the network [123]. Each attention weight Ω_n is computed as

$$\Omega_n = \frac{e^{\beta \lambda_n \cdot l_t^k}}{\sum e^{\beta \lambda_n \cdot a(\lambda_n)}} \tag{2.5}$$

where λ_n represents each possible level l_t^k belonging to the same layer L_t^k, $a(\lambda_n)$ is an indicator function, which indexes all the neighbors levels of λ_n belonging to the same layer L_t^k and β is a parameter that determines the sharpness of the attention.

Positional embedding. Similarly to [23], we define a set of positional embeddings to retain positional information. The same d-dimensional positional embedding $p(h,w)$ is shared among all the levels l_t^k belonging to the same column, one for each patch. As suggested in [40], to obtain a rich image representation. Similar to neural fields [71], the positional embedding vector is added to the output of the top-down network. The bottom-up networks to propagate the information to upper levels, causing the emergence of *islands of agreement* between neighbor patches $a(\lambda_n)$ corresponding to neighbor object representations.

Method	Backbone	Error %					# of params (Millions)	Training Arch.
		S-Norb	MNIST	F-MNIST	C-10	C-100		
E-CapsNet	Caps	2.54	0.26	-	-	-	0.2	GPU
CapsNet	Caps	2.70	0.25	6.38	10.6	82.00	6.8	GPU
Matrix-CapsNet	Caps	1.40	0.44	6.14	11.9	-	0.3	GPU
Capsule VB	Caps	1.60	0.30	5.20	11.2	-	0.2	GPU
ResNet-110	Conv	-	2.10	5.10	6.41*	27.76*	23	GPU
VGG	Conv	-	0.32	6.50	7.74*	28.05*	20	GPU
ViT-L/16	Transf	-	-	-	0.85*	6.75*	632	TPU
ConvMLP-L	Conv/MLP	-	-	-	1.40*	11.40*	43	TPU
MLP-Mixer-L/16	MLP	-	-	-	1.66*	-	207	TPU
Agglom (**Ours**)	Conv/MLP/Caps	0.01	0.30	7.43	11.15	40.97	72	GPU
Agglom++ (**Ours**)	Conv/MLP/Caps	0.01	0.30	5.74	9.35	35.6	1.3	GPU

Table 2.2: Error percentages on the Top-1 accuracy results on datasets SmallNorb (S-Norb), MNIST, FashionMNIST (F-MNIST), CIFAR-10 (C-10), and CIFAR-100 (C-100). The $*$ notation indicates results obtained with networks pre-trained on ImageNet.

The top-down network, using both object representations $a(\lambda_n)$ and positional encoding $p(h,w)$, decodes whole object representation into different parts. For instance, starting from a uniform vector l_{t-1}^{k+1} representing a dog's face (*whole*), the addition of positional embedding allows the network to decode this vector into diverse embeddings l_t^k, such as a mouth or ears (*parts*). Through the integration of positional embeddings, our network enables the emergence of *islands of agreement* without the necessity of contrastive learning.

Propagation phase. At each time step $t \in \{1,\ldots,T\}$, we compute the values l_t^k as

$$\begin{aligned} l_t^k = avg(&\omega_l l_{t-1}^k, \omega_{BU} N_{BU}(l_{t-1}^{k-1}), \\ &\omega_{TD}(N_{TD}(l_{t-1}^{k+1}) + p(h,w)), \omega_A A(L_{t-1}^k)) \end{aligned} \tag{2.6}$$

where $avg()$ indicates the arithmetical average, and $\omega_l, \omega_{BU}, \omega_{TD}, \omega_A$ are trainable weights.[1]

[1]For layer L_t^K, contribution $N_{TD}(l_{t-1}^{k+1})$ is not included, as L_t^{K+1} does not exist.

	Configuration	Error %	# of levels K	Levels embedding *d*	# of params (Millions)
I	Agglomerator [24]	11.15	2	128	72
II	Agglomerator++ (Ours)	9.35	5	192	1.3
III	Without $A(L_t^k)$	13.99	5	192	1.1
IV	Without $p(h,w)$	13.03	5	192	1.3
V	Decrease K	12.50	3	192	0.8
VI	Increase K	11.90	8	192	1.9
VII	Decrease d	12.70	5	128	0.6
VIII	Increase d	9.55	5	512	9.0
IX	Without regularization $\mathcal{L}$	9.57	5	128	1.3
X	Without conv tokenizer	16.91	5	128	3.4

Table 2.3: Ablation study results of different Agglomerator configurations obtained on CIFAR-10.

2.3.2 Training

The training procedure of our architecture is divided into two steps: (i) a pre-training phase where we train Agglomerator++ as an auto-encoder with a reconstruction loss for masked patches of the image [122] coupled with a *consensus* regularization loss [40] and (ii) a training phase for the image classification using a Cross-Entropy loss.

Pre-training to obtain a rich neural representation: As shown in Fig. 2.14 and similarly to [122], we pre-train our network to reconstruct the masked pixels x_M of image input. Specifically, we set 50% of the total patches to random values before applying the convolutional tokenizer. The final neural image representation is derived from iterative information routing, with reconstruction loss $\mathcal{L}Recon$ attached to layer Lt^1. This detailed representation is a blend of low-level feature encoding and high-level abstract aggregation [40]. The recon-

struction loss is defined as:

$$\mathcal{L}_{Recon} = \frac{1}{\Omega(x_M)} ||y_M - x_M||_1 \tag{2.7}$$

where $x, y \in \mathcal{R}^{3H_pW_p}$ are the input and predicted RGB pixel values, M denotes the set of masked pixels, $\Omega(.)$ is the number of total pixels, and $||.||_1$ is the 1-norm.

As suggested in [40], we use a *consensus* loss to regularise the network and encourage the formation of *islands of agreement* at the top level of the architecture. We define the *consensus* vector V_t^k, the bottom-up loss vector $V_{BU}(L_t^K)$ and the top-down loss vector $V_{TD}(L_t^{K-1})$ as

$$V_t^k = l_t^k(1) \frown l_t^k(N) \tag{2.8}$$

$$V_{BU}(L_t^K) = N_{BU}(l_{t-1}^{K-1}(1)) \frown N_{BU}(l_{t-1}^{K-1}(N)) \tag{2.9}$$

$$V_{TD}(L_t^{K-1}) = N_{TD}(l_{t-1}^{K}(1)) \frown N_{TD}(l_{t-1}^{K}(N)) \tag{2.10}$$

where the $. \frown .$ operator denotes the concatenation between all the d-dimensional vectors belonging to the same layer, thus obtaining a vector of size $N \times d$.

Defining the cosine distance loss between two vectors x and y as $\mathcal{L}_d(x,y) = 1 - \cos(x,y) = \frac{\boldsymbol{x}^T\boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$, the resulting loss to be minimized at the pre-training stage is

$$\mathcal{L}_1 = \mathcal{L}_{Recon} + \mathcal{L}_d(V_{BU}(L_t^K), V_t^K) + \mathcal{L}_d(V_{TD}(L_t^{K-1}), V_t^{K-1}) \tag{2.11}$$

The last two terms act as a regularizer to improve the coherence of *islands of agreement* as described in Sec. 2.3.1.

Fine tuning the architecture As shown in Fig. 2.15 Once the network is pre-trained using the $\mathcal{L}_1$ loss, we detach the pre-training losses from the architecture. To calculate the input to the cross-entropy loss for the classification, we compute the mean of all d-dimensional l_t^K vectors like in [23]. The resulting d-dimensional vector is fed through a layer normalization and a linear stage which reduces the size to c, namely the number of classes to be predicted for each dataset. Then we apply the standard cross-entropy function:

$$\mathcal{L}_2 = CE(y, \hat{y}) = -\frac{1}{c}\sum_{i=1}^{\mathrm{c}} y_i \log(\hat{y}_i) \tag{2.12}$$

where y and $\hat{y}$ are the label and the prediction of the sample taken from the batch, respectively.

2.3.3 Experiments

We perform our experiments on the following datasets:

- **SmallNorb (S-NORB)** [59] is a dataset for 3D object recognition from shape.
- **MNIST** [58] and **FashionMNIST** [120].
- **CIFAR-10** and **CIFAR-100** [55].

Our network is trained in an end-to-end fashion on a single NVIDIA GeForce RTX 3090. All datasets are employed at native resolution, but SmallNorb, which is resized to 32×32 pixels as in [43, 91]. The tokenizer embedding creates $n = H/4 \times W/4$ patches represented by n d-dimensional vectors, where H and W are the pixels dimension of the input image. Thus, the corresponding number of columns is 8×8 for CIFAR-10, CIFAR-100, and SmallNorb, and 7×7 for MNIST FashionMNIST. During the pre-training, we set the following hyper-parameters: 1000 epochs, cyclic learning rate [101] in the range $[0, 1e^{-3}]$,

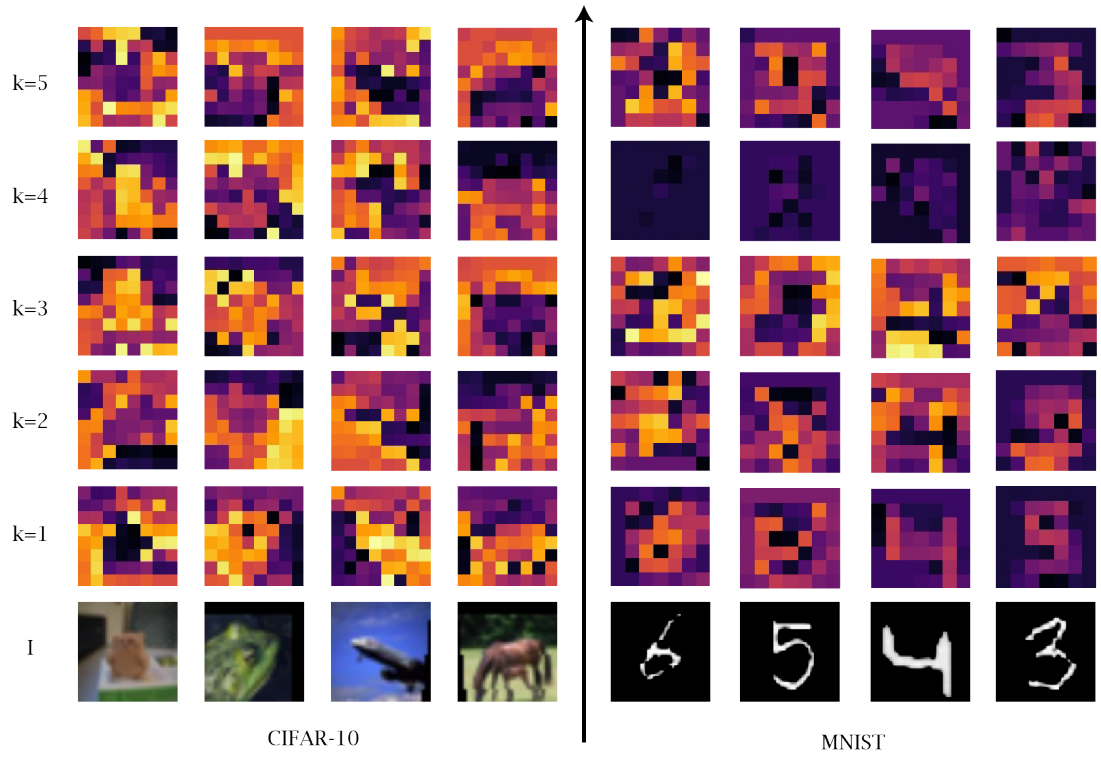


Figure 2.16: Illustration of the evolving islands of agreement at varying K levels for MNIST and CIFAR-10 datasets samples. Displaying agreement vectors for each patch at each k level post 300 epochs of pre-training. Level $k = 1$ functions akin to a feature extractor with minimal neighbor agreement. Lastly, at level $k = 5$, two islands surface representing the object and the background.

batch size $B = 512$, levels embedding $d = 192$, number of levels $K = 5$, number of iterations $T = 2K = 10$, and weight decay $5e^{-2}$.

2.3.4 Quantitative results

Quantitative results from Tab. 2.2 indicate our Agglomerator++ performs well on both simple (SmallNorb, MNIST, Fashion MNIST) and complex datasets (CIFAR10, CIFAR100) without architecture modification or pre-training. It matches capsule-based models on simple datasets [43, 91, 93], outperforms them on complex datasets, yet requires fewer parameters and training time than transformer-based [23] and MLP-based methods [61, 111]. While convolu-

tional models [37, 45] generalize well, they lack model interpretability. Agglomerator++ further enhances efficiency by sharing weights among networks and reducing extra layer requirements, thereby achieving comparable parameter count to capsule networks and superior performance on complex datasets. Agglomerator [24] has fewer parameters than most transformer-based and MLP-based methods, and it requires less training time on a much smaller architecture. While improving the numerical results, Agglomerator++ further reduces the number of parameters by enforcing the sharing of the weights among all bottom-up and top-down networks, and it avoids building extra layers on top of the architecture to perform the contrastive pre-training and the classification as in [24]. Thus, the number of network parameters is now comparable with capsule-based networks. The improvement in performance on more complex datasets with respect to capsules highlights the expressive power of our d-dimensional embeddings with respect to a group of neurons.

Ablation study. Our Agglomerator++ architecture's key components—attention $A(L_t^k)$, positional embedding $p(h,w)$, number of levels K, and embedding dimension d—underwent ablation studies, showing their impact on performance. Tab. 2.3 illustrates how Agglomerator++ (I) surpasses the original network version [24] (II) while decreasing parameter count through weight sharing across all networks and levels. Performance deteriorates when attention $A(L_t^k)$ (III) or positional embedding $p(h,w)$ (IV) are removed, indicating their crucial role. Modifying level number K (V)(VI) also affects performance, yet the expressive power d remains unchanged.

2.3.5 Qualitative results

The embedding size d crucially affects performance. Reducing d to 128 (VII) weakens results, while increasing it to 512 (VIII) maintains performance, but lengthens convergence time and model complexity. The elimination of the two regularization losses $\mathcal{L}_d(x,y)$ (IX) has minor impact on performance, primarily

affecting island formation. Removing the convolutional tokenizer (X) worsens results, hindering inter-patch information exchange at the embedding stage.

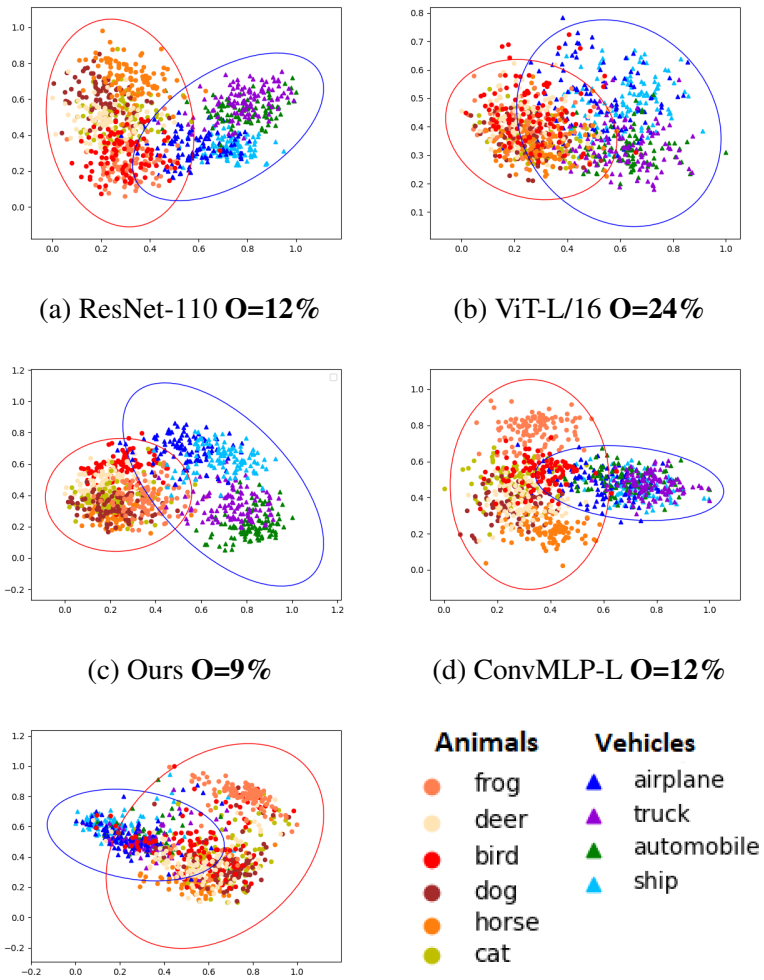
Our method provides interpretability of the *relationships learned by the model* by explicitly modeling the part-whole hierarchy, and of the *relationships contained in data* through the hierarchical organization of the feature space.

Part-whole hierarchy via island of agreement. During the *propagation phase*, neighbor levels on layer L_t^k are driven to form *islands of agreement*, showcasing part-whole hierarchies. Examples from MNIST and CIFAR-10 with $K = 5$ levels are shown. Instead of the convolutional tokenizer, a linear embedding, although lowering numerical results (Tab. 2.3), is used for better visualization. Vectors from the same island are grouped by cosine similarity between each embedding l_k^t and all level K embeddings. Embeddings with similarity under a set threshold are depicted by same-color squares in Fig. 2.9. As k for L_t^k increases, neighbors tend to agree on the *whole* representation. Lower levels display smaller islands each representing a part. Our Agglomerator++ represents a patch at different abstraction levels and the same level's patches agree on the representation.

Latent space organization as the representation of conceptual-semantic relationship in data. Recent networks aim at maximizing inter-class distances and minimizing intra-class distances between samples in the latent space. While the accuracy is high, they provide little interpretability in their data representation. As a result, mistakes are less likely to happen, but the mistake severity, defined as the distance between two classes in WordNet lexical hierarchy [72], does not decrease [7]. As shown in Fig. 2.17, our network semantically organizes the input data resembling the human lexical hierarchy, even though, differently from [24], no contrastive loss is enforced.

2.3.6 Discusssion

We presented Agglomerator++, a method that makes a step forward towards representing interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks. We believe that interpretable networks are key to the success of AI and deep learning. With this work, we show how our network can obtain better representations using a reconstruction, thus leading to better results with respect to similar networks trained with contrastive loss. The biological processes happening in our brains when learning are probably a combination of the two processes, which we view as the next step to obtain even richer and better explainable representations.



(a) ResNet-110 **O=12%** (b) ViT-L/16 **O=24%**

(c) Ours **O=9%** (d) ConvMLP-L **O=12%**

(e) Matrix-CapsNet **O=20%** (f) Legend

Figure 2.17: The 2D latent space representation of multiple methods trained on the CIFAR-10 dataset through PCA is illustrated. The PCA reduces data from multidimensional to 2D. The legend (f) classifies samples into *Vehicles* and *Animals* following WordNet hierarchy [72]. All methods (a,b,c,d,e) cluster samples between super-classes. The MLP-based methods (c,d) offer superior super-class separation, while placing similar samples together. Our method (c) optimizes inter-class and intra-class separability. The overlap percentage O denotes areas prone to severe hierarchical errors [7].

Chapter 3

Lagrangian Properties for efficient NeRFs

Deep learning has revolutionized the field of computer science. Over the last decade, it has transitioned from being a theoretical exploration to the core of various real-world applications, that touch every aspect of our daily lives, from autonomous vehicles and personalized recommendation engines to advanced image and speech recognition systems, not to mention the revolution in content creation through generative AI. These deep learning methods have a huge impact in our lives and more in general on the society. For instance, text-to-image models like Stable Diffusion [92] have opened up the art world to the masses, altering how artists conceive and create their work. Similarly, AI chatbots, like ChatGPT, serve as versatile tools that assist in everything from writing and summarizing to programming and problem-solving.

The integration of deep learning into computer graphics, most importantly through the development of neural implicit representations, has revolutionized our approach to processing and manipulating visual information. This field was previously ”dominated” by explicit representations like voxels and meshes. However, Neural Radiance Fields (NeRF) [71], as a leading illustration of this shift, have shown their ability to generate photorealistic images from sparse data. However, this innovation is doesn't come without challenges. The inherent computational demands associated with NeRFs, characterized by long

training and inference times, pose significant barriers to their broader application and integration into real-time systems.

In this context, the quest for more efficient neural field representations has led to the exploration of various strategies designed to accelerate these processes without compromising the quality of the generated outputs. Among these, the introduction of techniques that leverage an external source of parameters for fast training and inference has opened new direction for research and development. Yet, despite these advancements, an open issue remains the spatially non-homogeneity structure of 3D data: more features should be allocated for parts of the data with higher complexity. Achieving this objective would let the representation use the available memory footprint more efficiently.

These feature grid-based neural fields typically encode data through an Eulerian perspective, using vector fields aligned to a coordinate system. This methods, despite incorporating sparse or optimized data structures, maintains a uniform grid distribution for ease of indexing. On the other hand, Lagrangian methods provide the advantage of spatially adaptive grid point allocation. Yet, this adaptability comes at the cost of more intricate indexing algorithms, possibly requiring methods like approximate nearest neighbor searches to manage the complexity.

In this chapter we first discuss the initial research conducted as an exploration of the computer graphics field, focusing on SinGAN-3D a novel 3D generative model. This model is capable of unconditionally generate 3D shapes, utilizing voxels as a representation method for 3D data. Next, we introduce the core contribution in this chapter, Lagrangian Hashing, a novel approach that marries the simplicity and performance of Eulerian representations with the flexible, spatial adaptability of Lagrangian point-based frameworks. Building on the foundation of hierarchical hashes set by InstantNGP [78], our approach differs by assigning a set of points, each with its distinct feature rather than limiting a bucket to a single feature. Most importantly, as point-clouds can

adaptively allocate representation budget by increasing the density of the point cloud where needed, our Method is able to learn an effective representation for storing high frequency information.
In the last section 4.1, we perform a study on the impact of Neural Radiance Fields based techniques in the field of 3D reconstruction with a against classical photogrammetry.

3.1 SinGAN-3D

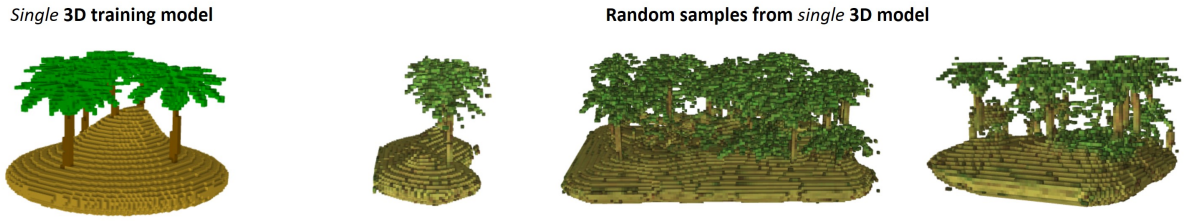


Figure 3.1: Given a single 3D model as input, SinGAN-3D is able to learn its latent structure and then generate new diverse voxel models, at any aspect ratio.

Generative adversarial networks (GANs) are deep generative models conceived by Goodfellow et al. in 2014 [28]. GANs have been adopted in many different domains, including view synthesis, image editing and style transfer [17] [21] [131], to name a few. A big step forward in the use of GANs for image generation, editing, and resampling, is presented in SinGAN [97]; in their work, the authors propose a novel generative model capable of learning the underlying distribution of a single image, allowing to perform any kind of generative task. The main advantage brought by this solution consists in the ability to learn from a single image, instead of relying on an expensive and diversified dataset.
In our work, we present SinGAN-3D, an extension of SinGAN [97] meant to be applied to 3D data. Our network takes a single voxelized model as input, and learns the structure of its spatial patches. Once trained, it can generate new

voxel models, respecting the semantic content of the input model, whilst exhibiting different spatial structures and configurations. While the extension of the model could be trivially considered as to a ”one-more dimension” problem, the extension of SinGAN to the 3D domain has shown a number of challenges; in the first place the increased data complexity, due to the addition of the third dimension, which poses serious spatial and semantic challenges. In fact, it is worth noting that the generation of 2D images, as also presented in the original work, leaves a certain freedom to the algorithm, sometimes also giving to the generated model an *artistic* flavour, which is deliberately unnatural, though preserving the general layout and appearance of the source model content. 3D representations, instead, need to comply with strict spatial and semantic constraints, in order to look acceptable to a human observer. Another challenge is posed by the identification of a data representation strategy suitable to reproduce colors in the 3D domain. To relax memory and computation requirements we represent density and color (R,G,B) separately obtaining a 4-dimesional representation. This 4-dimensional representation is in line with GPU rendering techniques based on 'mesh + texture', as voxels occupancy and shapes provide the geometric information, while colors or texture provide the color information for the corresponding volumes or vertices. The original implementation of SinGAN is composed of a fully convolutional set of GANs, intent on capturing the distribution of internal patches at a different scale. In our model, we replace the 2D convolutions with 3D ones; in addition we exploit periodic activation functions [99] as a supplementary tool to capture the internal structure of the 3D images. We demonstrate the ability of our model to generate samples of a single voxel model characterized by different resolutions and aspect ratios, as shown in Figure 3.2. We further demonstrate the applicability of our network to modify the aspect ratio of the generate model, as well as to perform super resolution from a single 3D model. Our contribution is summarised as follows:

- we introduce SinGAN-3D, a generative model capable of learning the un-

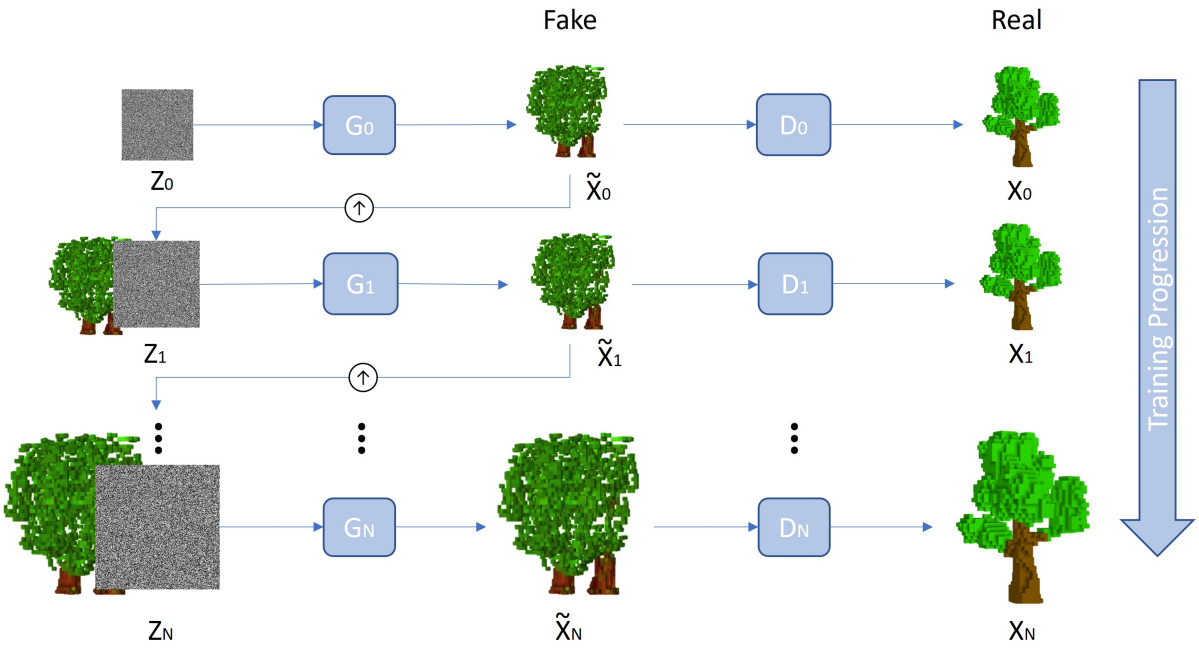


Figure 3.2: SinGAN 3D's multi-scale generation pipeline on the "simple tree" voxelized model. At each scale the 3D generator G_n learns to generate voxelized models respecting the latent structure of the down-sampled training volumetric image x_n. The 3D discriminator D_n tries to distinguish the generated samples from the original ones. G_n takes as input (together with the downsampled voxelized model) a random 3D Gaussian noise map z_n, except for the first generator G_0, which takes as input only the 3D noise map.

derlying structure of voxelized models and generating similar samples at different resolutions.

- we demonstrate the applicability of sinusoidal activation functions to the 3D generation task, showing promising results.
- we also demonstrate the use of our models for super resolution applications, with high quality results.

3.1.1 Method

We introduce this section with a brief background description of the original SinGAN [97] method, and how it is applied to 2D images. We then present in

detail our solution, showing how the approach can be extended to work with three dimensional inputs.

Background: SinGAN

SinGAN [97] is an unconditional generative model, capable of learning the latent structure of a single 2-dimensional image and use it to perform both generative and manipulation tasks. The model consists of a pyramid of Wasserstein GANs [3] where each layer of the network learns the distribution of patches in the single image at the corresponding resolution via adversarial learning. SinGAN sample generation starts from the highest and coarsest layer and progresses sequentially down to the last layer. The result is then passed through the generator convolutional network and the output of the latter is summed to the up-sampled image from the previous layer. As each layer works with increasing resolutions, details are progressively added as the sample descends the pyramid and the noise introduced at each layer ensures variance in the generated details. Finally, as the lowest layer is reached, the fine-grained details are generated and the output result is a sample having the same size as the original input data. For an exhaustive explanation we invite the reader to refer to the original SinGAN manuscript [97].

SinGAN-3D

Our proposal to extend SinGAN to the third dimension allows the use of voxelized models as input to the network. We maintain the pyramid of generators presented in SinGAN; more specifically, each of them is trained against a 3D image pyramid obtained from the down-sampled version of the input. This is obtained via adversarial learning, where each generator is responsible of producing an output retaining the latent structure of the input. The generation of an image sample starts at the coarsest scale (which is purely generative) and the output is iteratively fed through all generators up to the finest scale, as shown in

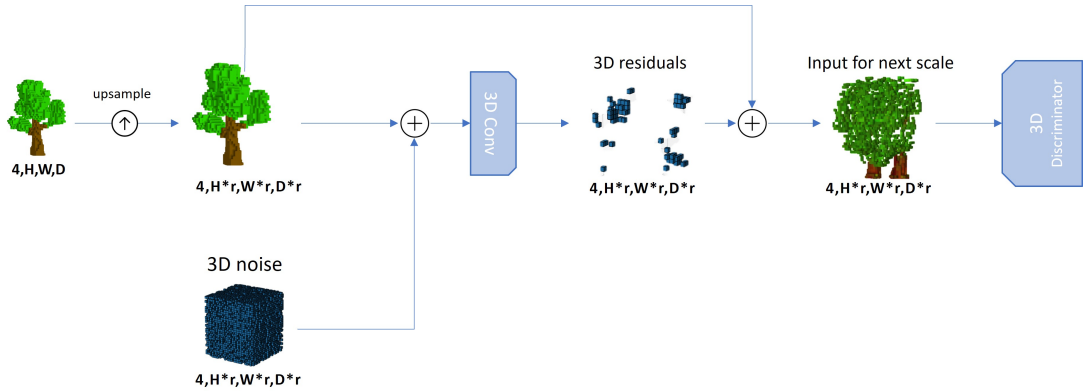


Figure 3.3: SinGAN-3D's training pipeline at single scale. Given a voxelized model at scale n with spatial dimensions H, W and D, we first upsample it by factor r. The upsampled model is then added to a 3D Gaussian noise and fed to a 3D CNN, with periodic activation function, to estimate a residual model, which is added back to the input upsampled model. The obtained model is trained with adversarial loss from a 3D discriminator, and used as input for the next scale, iteratively.

Figure 3.2, with noise injected at every scale. The pipeline of SinGAN-3D for the generation of random samples at a single scale is shown in Figure 3.3. An important difference with respect to the original work is the transformation from a continuous to a quantized space. The main implication of this transformation is that, while pixels in an image carry color intensity information, voxels convey the volumetric appearance of the object being represented, thus resembling its 3D structure.

3D Data Representation

3D models converted to volumes usually do not carry color information [119] and the intensity of voxels can be interpreted as how likely it is for them to exist in that specific location. In the 3D domain colors are represented with a separate data structure with respect to volumetric occupation. In order to model the different pieces of information jointly (spatial distribution and color), our approach consists of representing the shape information and color with a 4-dimensional volume, where one channel (binary) informs about the existence of a certain

voxel location; the remaining channels model instead the color appearance. The size of the converted volumes may vary depending on the proportions of the original 3D model; consequently during the development of our approach we choose to work with $80x80x80$ voxel models. The choice was motivated by the need of achieving an acceptable level of detail, yet allowing sustainable training load. This solution proved to be effective, returning qualitatively good results.

Periodic Activation Function

The extension of the SinGAN model to the third dimension requires an additional tool; to extensively capture the structure of 3D voxel models, we propose to leverage the periodic activation functions, presented in [99]. Authors of [99] demonstrated that ReLU-based architectures lack in fine details; furthermore they do not effectively represent the derivatives of a target signal, turning into non-optimal results when encoding complex or large scenes with fine details. On the other hand, periodic activation functions demonstrated to rapidly converge to an accurate fit in complicated scenarios with high frequency details.
The authors of [132] propose a novel activation function, named "Snake": they claim that the proposed activation function is capable to learn periodic functions while still being able to optimize loss functions with competitive results as conventional activation functions. The snake activation function is defined as: $Snake = x + sin^2(x)$. The applicability of this activation function to generative tasks has been proved in [4]; they trained a deep convolutional GAN (DCGAN) [84] to generate samples of the MNIST dataset. The network trained with Snake activation generates realistic samples, which are qualitatively indistinguishable from those generated by a conventional activation function.

Training Procedure

The network is trained sequentially from the coarser scale to the finest one; once a generator is trained, it is kept fixed until the next generator is optimized. At

each scale the generator is trained using adversarial loss from its corresponding discriminator. The implemented loss is calculated in a similar manner to the original SinGAN paper but extended to the third dimension:

$$\min_{G_n} \max_{D_n} L_{adv}(G_n, D_n) + \alpha L_{rec}(G_n) + \beta L_{color}(G_n) \tag{3.1}$$

In Equation 3.1, G_n and D_n denote the Generator and the Discriminator for the scale n, respectively. L_{adv} denotes the adversarial loss, defined to penalize the distance between the patches in the input image x_n and the patches in the generated samples $\hat{x_n}$; L_{rec} and L_{color} define the reconstruction and color losses respectively.

Color Loss

The original implementation of SinGAN employs only the adversarial and reconstruction losses; however, for 3D models using only these losses the color distribution of the generated samples differ significantly from the one of the training models. Authors of [128] also reported the same issue, and resolved it by matching the mean and covariance of the colors from the generated images at progressive resolutions. In our framework, we match the mean and covariance of the the three color-channels at each spatial resolution by means of a color loss, defined as:

$$L_{color} = ||\mu_{x_i} - \mu_{\dot{x}_i}||_2 + ||Cov(x_i) - Cov(\dot{x}_i)||_2 \tag{3.2}$$

Where *cov* is the covariance, μ is the mean, x is the set of training frames, and $\dot{x}$ denotes the generated frames.

3.1.2 Results

The proposed SinGAN-3D has been trained with a variety of 3D voxel models with a resolution ranging from $40x40x40$ to $80x80x80$ voxels. We selected

	Abstract obj. Samples		City Samples		Crystal Samples		Simple Forest Samples		Single Tree Samples	
	Snake	ReLU	Snake	ReLU	Snake	ReLU	Snake	ReLU	Snake	ReLU
Abstract Obj	0,646	0,573	0,162	0,204	0,279	0,319	0,185	0,302	0,033	0,086
City	0,110	0,166	0,897	0,843	0,201	0,216	0,141	0,244	0,008	0,000
Crystal	0,278	0,330	0,194	0,228	0,985	0,852	0,283	0,307	0,035	0,043
SimpleForest	0,139	0,154	0,131	0,168	0,254	0,241	0,516	0,414	0,074	0,053
Single Tree	0,069	0,081	0,065	0,096	0,079	0,073	0,097	0,050	0,228	0,064
Trees	0,182	0,140	0,052	0,096	0,134	0,140	0,133	0,164	0,130	0,098
Rocks	0,231	0,325	0,202	0,249	0,300	0,268	0,355	0,368	0,061	0,081
Group of palms	0,052	0,060	0,151	0,172	0,181	0,278	0,151	0,105	0,097	0,025
Phormium	0,042	0,049	0,047	0,060	0,000	0,097	0,000	0,031	0,000	0,000
Waterfall	0,000	0,049	0,000	0,088	0,000	0,000	0,000	0,000	0,000	0,000

	Trees Samples		Rocks Samples		G. of Palms Samples		Phormium Samples		Waterfall Samples	
	Snake	ReLU	Snake	ReLU	Snake	ReLU	Snake	ReLU	Snake	ReLU
Abstract Obj	0,024	0,117	0,274	0,314	0,103	0,123	0,101	0,110	0,029	0,036
City	0,147	0,151	0,203	0,185	0,156	0,179	0,043	0,027	0,000	0,000
Crystal	0,167	0,162	0,332	0,336	0,169	0,240	0,208	0,198	0,000	0,000
SimpleForest	0,150	0,151	0,351	0,342	0,182	0,149	0,246	0,297	0,007	0,024
Spyrals	0,328	0,168	0,098	0,063	0,124	0,074	0,329	0,344	0,007	0,039
Trees	0,552	0,498	0,144	0,163	0,145	0,227	0,278	0,283	0,004	0,041
Rocks	0,093	0,159	0,676	0,644	0,259	0,223	0,158	0,204	0,000	0,046
Group of palms	0,000	0,012	0,248	0,318	0,752	0,697	0,110	0,161	0,000	0,073
Phormium	0,147	0,183	0,000	0,044	0,000	0,005	0,444	0,375	0,000	0,054
Waterfall	0,000	0,008	0,047	0,081	0,000	0,000	0,023	0,008	0,920	0,875

Table 3.1: The following table shows MSSIM scores of all real training voxel volumes compared against 50 generated samples of each class. Results highlight how Snake activation function better preserve the latent structure of the training image. For space reasons we divided the table in two parts.

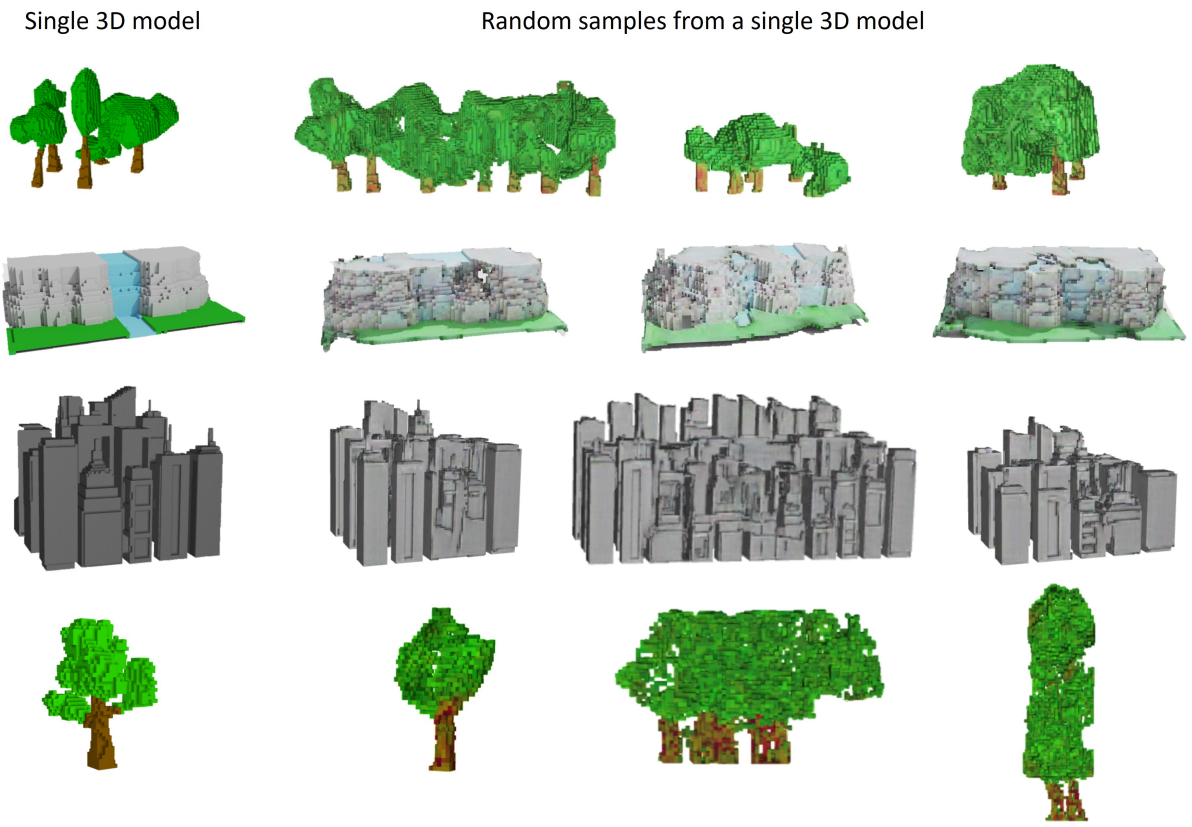


Figure 3.4: First column: original input voxelized models. Columns 2-4: Generated samples from the input training model. After training SinGAN-3D on a single voxelized model it can generate random samples, with arbitrary aspect ratios, characterized by a similar latent structure with respect to the training voxelized models.

this range of resolutions as it provides an acceptable level of detail yet allowing a reasonable computational load in training the network. The models we used contain objects and scenes spanning from urban and nature scenery. Some qualitative examples of the randomly generated 3D voxelized samples are shown in Figure 3.4. We demonstrate the capability of our model to capture the latent structure of the 3D shapes and the ability to generate new realistic volumes. In order to quantitatively evaluate the generated 3D voxelized models we calculate the 3D Single Image Fréchet Inception Distance (3D SiFID) score, which is an extension of the Single Image FID score for 3D models. To compute the 3D SiFID score, we extract the activations from the last convolutional layer of

the reference Convolutional Neural Network; for this experiment we use Voxnet [69], as it is explicitly designed for voxel models. The formulation of FID, computed between the original model and the generated one, is defined as follows:

$$d^2 = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2 * \sqrt{C_1 * C_2}) \tag{3.3}$$

where μ_1, μ_2, C_1 and C_2, are the mean values and the covariances of the activation functions of the last convolutional layer of generated and original samples respectively. We further evaluate the performances of our model by computing the Multi Scale Structural Similarity index between samples generated from different voxelized models. This experiment has been performed to highlight that samples generated from different training 3D shapes are qualitative different from each other, while being similar to the training input. Results show a significant value of similarity between samples and the reference voxelized model as highlighted in Table 3.1. Lastly, we performed a perceptual user study using Amazon Mechanical Turk. Images were visible to workers for 3 seconds; they were asked to decide whether the image was obtained from a real scene or generated by an artificial intelligence algorithm. We generated 10 samples for each of the 10 training voxelized models, having 3 votes for each sample. To measure the confidence, we used a simple yet effective majority voting. Out of 100 generated samples evaluated from the *turkers*, 39 were identified as being obtained from real scenes. This is an encouraging number, especially considering the low resolution of the generated volumes, confirming the viability of the proposed solution.

Ablation Study

We further examine the performances of the application of periodic activation functions in our framework by comparing our scores against SinGAN-3D trained with a ReLU activation function. For a fair comparison between the two networks we compared samples generated with the same Gaussian noise seed.

As shown in Table 3.2, our samples have lower 3D SiFID scores; this indicates that our approach models the latent structure of the training 3D shapes better than the trivial 3D extension of SinGAN.

Table 3.2: 3D SiFID scores averaged over 450 voxelized models (50 samples per each training input).

Model	3D SiFID Score
SinGAN-3D ReLU	0.025
SinGAN-3D Snake	0.019

3.1.3 Applications

We show the application of SinGAN-3D for two important editing applications: super resolution and change of the aspect ratio. To perform both operations the original training pipeline is kept unaltered, without further tuning.

Super Resolution

The aim of this task is to increase the resolution of an input 3D model by a factor k. Authors of [26] prove that patches in a given image tend to recur over different scales of the image. Thanks to this property we exploit the capability of our network to perform super resolution by iteratively adding finer details to the input data. This is done by injecting at each layer of the generators pyramid the up-sampled low resolution model. To perform this operation we follow a similar pipeline to the original SinGAN. Consequently, to upsample the 3D input by a factor k, we set the 3D pyramid scale factor to $\sqrt[n]{k}$ for k $\in \mathbb{N}$. A sample result is shown in Figure 3.5.

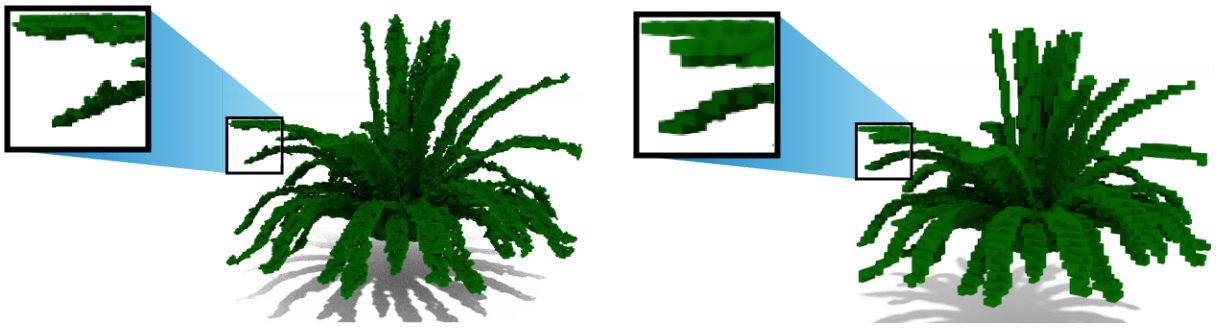


Figure 3.5: Left 3x up-sampling using SinGAN-3D. Right: 3x up-sampling using trilinear interpolation. The source model has a resolution of 80^3 voxels. Compared to the trilinear interpolation, which appears to flatten out the plant leaves, the proposed model preserves a better texture and a more realistic appearance.

Change of the Aspect Ratio

The objective of this application is to produce voxelized models that maintain the visual aspect of the original one but vary in spatial resolution. Since our network is fully convolutional, samples with different spatial resolutions can be easily generated by changing the dimensions of the Gaussian noise map at the base of the pyramid. Some real world uses involve the generation of similar looking 3D images to fit different screen sizes, or data augmentation to obtain a collection of samples from a single voxelized model. Variations of the original voxelized models with different spatial resolutions are shown in Figure 3.4.

3.1.4 Discussion

This section of the thesis presents SinGAN-3D the first work conducted as an exploration of the Computer Graphics field. The developed model is capable of learning from a single voxelized model and generate samples resembling its structure. We expand the capabilities of the SinGAN architecture to work with 3D inputs. We demonstrate the applicability of periodic activation functions for this specific task, achieving promising results. We further propose to model the 3D inputs as 4-dimensional models to add the RGB color information. An

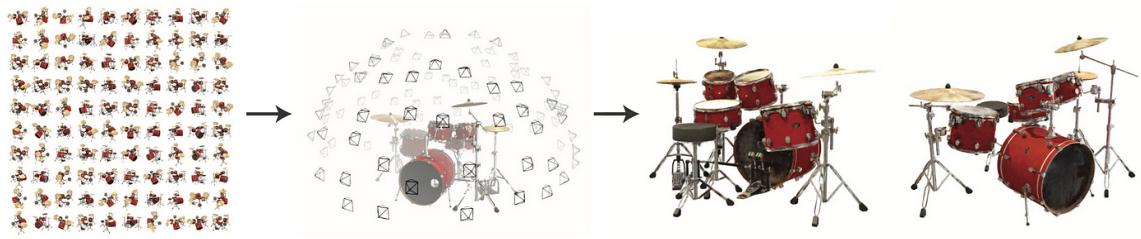


Figure 3.6: Neural Radiance fields enables representing the scene with a 5D continuous representation, from a set of input images. Volume rendering techniques are used to accumulate samples along rays inside the 3D scene to render images from every view-point. Image taken from [71].

existing limitation consists of the limited capability of learning the semantic meaning of an object, hindering the possibility of generating complex and rich structures. Integrating the architecture with attention, as authors of [15] propose for the original SinGAN, is a possible way to overcome this limitation. Overall, the achieved results are promising, showing the potential of generating arbitrary shapes starting from single 3D representations.

3.2 Neural Radiance Fields

Neural Radiance Fields (NeRF) have transformed the landscape of view synthesis and scene reconstruction, providing a novel way to capture and render complex three-dimensional environments. This method differs from conventional approaches since it utilizes a coordinate-based neural network to model scenes as continuous fields of radiance. Which is optimized directly without the need for pre-existing training datasets beyond the scene images themselves, as shown in image 3.6. Unlike traditional local interpolation techniques that once dominated the field, NeRF's neural framework requires no additional external data for training and achieves a representation that is significantly more compact (over a hundredfold) compared to equivalently resolved dense sampling arrays.

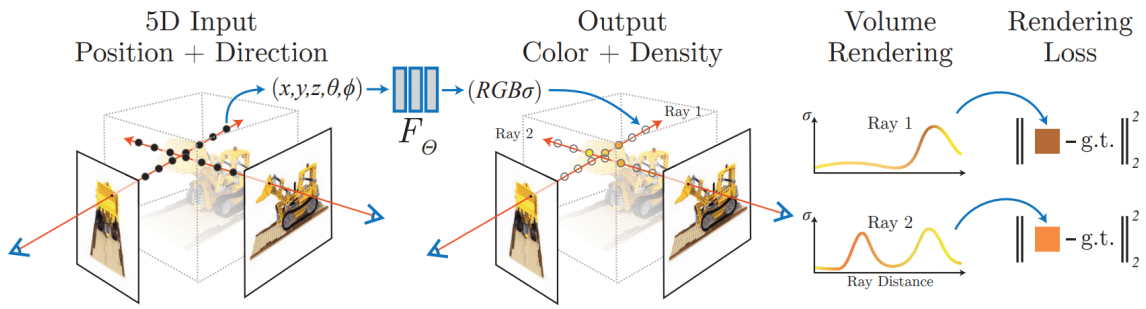


Figure 3.7: An overview of the neural radiance field scene representation and differentiable rendering procedure. The output images are synthesized by sampling 3D coordiantes and 2D input directions along camera rays (a). These locations and directions are fed into an MLP to produce a color and a volume density (b). Volume rendering is used to produce an image with those by blending these values. The rendering is differentiable, so it enables a training routine with automatic differentiation, minimizing the residuals between the produced and the groud-truth image.(d) Image taken from [71]

At the heart of NeRF's approach is the concept of representing a scene in a high-dimensional space, where each location $x = (x, y, z)$ and viewing direction (θ, ϕ), or equivalently, a 3D Cartesian unit vector d, is associated with specific radiance (color) and density values. This association is captured by an MLP network, $F_\Theta : (x, d) \rightarrow (c, \sigma)$, which is optimized to map these 5D coordinates to their corresponding output values.

To ensure the network accurately represents multi-view consistency—crucial for realistic rendering—the volume density σ is predicted based on the location alone, while the RGB color c is determined by both the location and viewing direction. This differentiation allows NeRF to capture complex view-dependent effects, such as specular highlights, that are essential for photo-realism.

Rendering from a NeRF involves simulating how light travels through and interacts with the scene. This process utilizes classical volume rendering techniques, interpreting the volume density $\sigma(x)$ as a measure of the likelihood that light is absorbed or scattered at a point in space. The expected color $C(r)$ of a camera ray $r(t) = o + td$ is calculated through an integral that accumulates

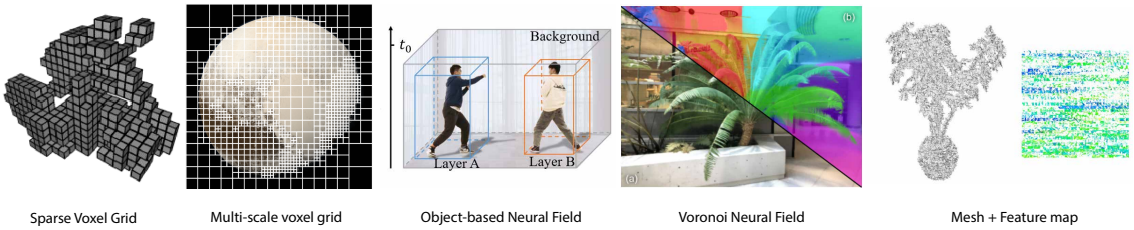


Figure 3.8: Examples of hybrid representations: starting from the leftmost Neural Sparse Voxel Grid [65], multi scale voxel grid with neural 2D image compression [68], object bounding boxes with neural radiance field [81], Voronoi decomposition with neural radiance fields [86], triangle mesh and by deep features [16].

radiance along the ray, modulated by the transmittance $T(t)$, which represents the likelihood of the ray traveling unobstructed through the scene.

A key challenge in rendering NeRF scenes is efficiently estimating the integral over continuous space, which is addressed through a stratified sampling strategy. This approach ensures that the MLP is queried at a diverse set of points within the scene, allowing for a continuous representation despite the discrete nature of numerical integration.

This rendering process is inherently differentiable, allowing the use of gradient descent to optimize the model to represent complex scenes. The optimization aims to minimize the discrepancy between observed images and views rendered from the neural radiance field, thus encouraging the network to construct a coherent model of the scene.

3.3 Hybrid NeRF approaches

Recent advancements in neural fields have seen the rise of hybrid neural field models, which represent combination of classical data structures and neural network methodologies. These models utilize auxiliary grids to arrange additional trainable parameters, achieving a unique synthesis of global continuity and local detail-oriented neural field capabilities, thereby enhancing the accuracy and

richness of high-frequency detail representation.

Discrete data structures—ranging from regular and adaptive grids to point clouds and meshes—are staples in visual computing for their ability to efficiently organize and process spatial information. When these structures are combined with neural fields, several advantages are realized:

- Computational Efficiency: Utilizing structures like bounding volume hierarchies enables rapid queries, significantly reducing the computational overhead.
- Network Capacity Optimization: Large, monolithic MLP networks often suffer from diminishing returns regarding their representation capacity. Discrete structures help allocate network capacity more effectively.
- Enhanced Rendering: The use of discrete structures facilitates techniques like empty space skipping, which can substantially accelerate the rendering process.
- Versatility for Simulation and Editing: Discrete structures are inherently suitable for simulations (e.g., finite element methods) and provide a solid basis for manipulation and editing tasks.

The integration of neural fields with discrete data structures typically involves storing neural field parameters within a spatially organized framework, allowing for efficient parameter retrieval based on input coordinates. This approach manifests in two primary methodologies: network tiling and embedding. Network tiling involves employing a collection of distinct neural fields, each covering a specific region of the input space. Although these networks share an architectural blueprint, they maintain separate parameters for their assigned regions. A coordinate lookup retrieves the relevant network parameters, enabling localized processing. Alternatively for embedding, latent variables are stored within the discrete structure, serving as compact, local representations of the

neural field parameters. Through a mapping function, these embeddings are translated into parameters that inform the neural field's output.

As defined in the survey "Neural Fields in Visual Computing" [121] the discrete datastructure g can be defined as vector field that maps coordinates to quantities using a sum of Dirac delta functions δ:

$$g(x) = \alpha_0\delta(x_0 - x) + \alpha_1\delta(x_1 - x) + \ldots + \alpha_n\delta(x_n - x) \tag{3.4}$$

where coefficients α_i (scalar or vector) are the quantities stored at coordinates x_i. For uniform voxels, the coordinates x_i are distributed on a regular grid, whereas for a point cloud the coordinates x_i are distributed arbitrarily. For network tiling, α_i is an entire set of network parameters Θ_i; for embedding, α_i is latent variable z_i.

The discrete data structure g may use an interpolation scheme to define g outside the coordinates x_i of the Dirac deltas, such as nearest neighbor, linear, or cubic interpolation. If so, instead of Dirac deltas, function δ are basis functions of non-zero, compact, and local support. For instance, voxel grids are regularly combined with nearest-neighbor interpolation, where $g(x)$ is defined as α_i at the x_i closest to x.

3.3.1 Regular Grid

The most common for of data-structure used in the context of hybrid representation to enhance NeRF's representation is the regular grid, defined as 2D pixels for images and voxels in 3D modeling. This grid form partition the coordinate domain uniformly, simplifying indexing and processing. However, their scalability is challenged by dimensional and frequency demands. Addressing this, adaptive and sparse grids concentrate capacity on regions of interest, benefiting from hierarchical and texture-based implementations. Grid tiling, by deploying smaller neural networks across segmented spaces, and embedding grids, by

localizing model parameters or latent variables, present effective strategies to manage large-scale, high-frequency signals. Following the framework of grid tiling the coordinate domain is discretized with a grid and each location is defined with a small neural network [89]. In this way this class of methods enable the representation of larger signals [47] or fasten their inference [87]. An known limitation of network tiling is the increased risk of creation of visual artifacts caused by overfitting to sparse data. This limitation is relaxed by the usage of parameter interpolation [19, 76].

The use of regular embedding grids, which are organized in a consistent pattern, requires interpolation methods to maintain smooth transitions across boundaries. This class of methods is particularly suitable for modeling large-scale signals [11]. However, this approach comes with a trade-off: the high quantity of parameters stored within these grids may restrict their application for extremely large signals due to memory constraints. Despite this, the advantage of having parameters directly embedded in the grid is the possibility of the deployment of smaller, more efficient networks, enhancing speed during inference [25, 65, 107]. Moreover, the structure of embedding grids proves to be beneficial for generative models [46, 94, 112] focused on both imagery and three-dimensional content, leveraging the detailed storage capacity of the grids for high-quality content generation.

3.3.2 Irregular Grids

Irregular grids partition the coordinate space in a non-uniform manner, relaxing the constraints of regular sampling patterns like the Nyquist-Shannon limit. This flexibility allows them to be dynamically reshaped, enhancing their capacity to accurately represent areas of data complexity. Connectivity among points can be established either explicitly through structures like meshes [16] or implicitly via Voronoi diagrams [86]. Additionally, these grids can be structured into hierarchical frameworks, including bounding volume hierarchies (BVH) or

scene graphs, to further optimize data representation and processing efficiency [29, 81].

Object-Centric Representations

Object-centric representations refine the granularity of scene understanding by focusing on individual objects rather than the scene as a whole. Unlike traditional point-based representations, each point in an object-centric framework is equipped with an orientation and a bounding box or volume, enabling detailed spatial reasoning and manipulation at the object level. This approach is exemplified in works such as [116] which introduces a method for estimating objects' poses and sizes across various categories using a normalized coordinate space. Neural field parameters, critical to defining the appearance and geometry of objects within a scene. These parameters can represent embeddings or neural networks themselves, facilitating a wide range of manipulations and interactions with the scene. Authors of [81] explores the use of neural scene graphs to manage dynamic scenes in an interpretable and manipulable format, allowing for individual object adjustments without compromising the scene's overall integrity. Further extending the utility of object-centric approaches, [32] focuses on rendering scenes by prioritizing individual objects. This prioritization enables precise control over object appearances and spatial relations, thereby enhancing the quality and flexibility of scene composition.

Mesh representations

Meshes serve as a fundamental data structure in computer graphics, knwon for their well-understood properties and a broad range of processing operations. Particularly in the realm of triangle meshes, the utility of embeddings stored on vertices is highlighted in applications such as novel view synthesis of dynamic human figures, as explored in [83]. In this context, barycentric interpolation emerges as a critical technique for achieving smooth and continuous represen-

tations across the mesh surface, enabling the precise manipulation and rendering of complex shapes and movements. Beyond triangle meshes, the challenge of representing and processing more complex polygons necessitates the adoption of more sophisticated coordinate systems. Mean-value coordinates and harmonic coordinates represent two such systems, offering enhanced flexibility and accuracy in the handling of complex polygonal shapes. A standout example of innovation in this area is Mobile NeRF by Chen et al.[16], which trains a 3D neural representation based on a polygonal mesh with color, feature and opacity connected to each mesh point. By discretizing alpha values and enhancing features for anti-aliasing, this method allows for the mesh's rasterization contingent on the viewing angle, utilizing a small MLP for pixel shading. This method proved to be around 200 times faster and nerf.

Point-based representations

Point-based representations have been widely explored as a representation in computer graphics[31, 52, 102]. However, rendering point clouds through ray-tracing presents a challenge due to the lack of a defined surface, complicating ray and point cloud interactions. Early solutions employed splatting techniques using disks [62] or ellipsoids [10, 88] to address this issue. Recent methods have also been used in reconstruction settings in conjunction with neural fields, but they frequently either require careful initialization, such as points from depth [2, 53, 80, 85, 124], COLMAP point cloud [48] and LiDAR [12, 105].

Recently, point-based neural rendering [48, 57, 129, 130] enabled the rendering of 3D point clouds onto images via differentiable rasterization. Particularly, Gaussian Splatting [48] and its follow-ups [118, 125, 126, 127] have achieved impressive results in rendering quality and test-time efficiency. Despite their quick adoption, these methods still rely on COLMAP for point initialization, and carefully tuned heuristics to grow and prune points. More importantly sufferfrom gradient vanishing when points are far from target pixels, thus these

methods require a large number of points to accurately model scene details typically requiring *large storage*, with millions of points needed to accurately represent a scene. A critical issue remains the diminishing gradients when points are far from target pixels, forcing reliance on a large number of to accurately model scene details, thus requiring considerable storage requirements.

3.4 Lagrangian Hashing for Compressed Neural Field Representations

As immersive mixed reality interfaces and volumetric content capture systems become popular, there is an increasing demand for a multimedia format that can compactly represent and transmit various types of multimedia. The diversity of multimedia formats (images, videos, volumetric 3D, radiance fields, etc) in mixed reality systems necessitates the codecs themselves to also be flexible to handle different types of formats.

Neural fields [121] have emerged as a general data format that can represent different multimedia formats with a unified codec based on model fitting. They have been popular in recent literature for representing all sorts of data, but in particular have been widely used for radiance field reconstruction [71]. In contrast to traditional multimedia formats that convert data into alternate formats through transformations, neural fields convert data by fitting a model to the data via optimization. This adds to the flexibility of the transformation by allowing the integration of additional objective functions and scenarios, like 3D reconstruction in an inverse problem setting.

A class of neural field models that have been particularly successful are *feature grids*, which uses a differentiable data structure holding features and a small multi-layer perceptron (MLP) as the model. These models inherit the strengths of highly performant data structure (such as an octree [107] or hash grid [78]) which allow the neural fields to fit complex data with large spatial extent with-

Figure 3.9: We introduce a hybrid representation that is simultaneously Eulerian (grids) and Lagrangian (points), which realizes high-quality novel view synthesis as shown above, while at the same time being more memory efficient.

out sacrificing performance. These feature grids methods, however, typically come at the cost of a larger memory footprint.

Although many data structure tricks like sparsity [68, 107], low-rank factorization [13], linear transforms [90], and hash probing [108] have been proposed to improve the *memory-quality tradeoff curve* of feature grids, these works do not fundamentally address the spatially non-homogeneous structure of 3D data: more features should be allocated for parts of the data with higher complexity. Achieving this objective would let the representation use the available memory footprint more efficiently.

These feature grid-based neural fields typically represent data in an *Eulerian* way, as a vector field over some coordinate system. Even if they use sparse or factorized data structures, they generally use a grid where vertices are laid out in uniform intervals which allows for simple implementations of indexing algorithms.

Lagrangian ways of representing data, on the other hand, would allow the representation to flexibly allocate the grid points in space, but may suffer from more complex indexing algorithms that may involve techniques like approxi-

mate nearest neighbour searches. [1]

Our work, Lagrangian Hashing (LagHash), marries the simplicity and performance of *Eulerian* representations where features are laid out on uniform grids with a *Lagrangian* representation that employs a point-based representation in which features can freely move around in space. In more details, our representation builds upon hierarchical hashes introduced by InstantNGP [78], but, in each hash bucket, rather than just storing a feature, we store a small set of points, each equipped with a feature. [2] While other point-based representations require acceleration data structures to access the points, our representation *reuses* the hash implementation. Most importantly, as point-clouds can adaptively allocate representation budget by increasing the density of the point cloud where needed, they are a more effective representation for storing high-frequency information, which results in smaller models that achieve similar visual performance.

3.4.1 Method

We introduce a new representation that combines the Eulerian nature of fast-training NeRF methods [78] to the Lagrangian nature of emergent 3D Gaussian Splatting (3DGS) representations [48]. Building directly on top of the hierarchical Eulerian representation introduced by InstantNGP [78], we achieve this by incorporating a point-based representation in the high-resolution layers of its hash tables. We select the high-resolution layers as to let the model focus its representation power to precise locations in space, akin to how 3DGS [48] employs small Gaussians to capture fine-grain details of the scene. We (implicitly) equip each point with a standard deviation proportional to the grid resolution, hence our representation can be interpreted as a mixture of Gaussian with non-

[1] We refer to Eulerian and Lagrangian representations in numerical physics, where one can store the state of the system (as an example, velocity of a fluid) on either grids (Eulerian) or on particles (Lagrangian).

[2] Note this is done on a selection of levels, and therefore our representation is an Eulerian-Lagrangian hybrid.

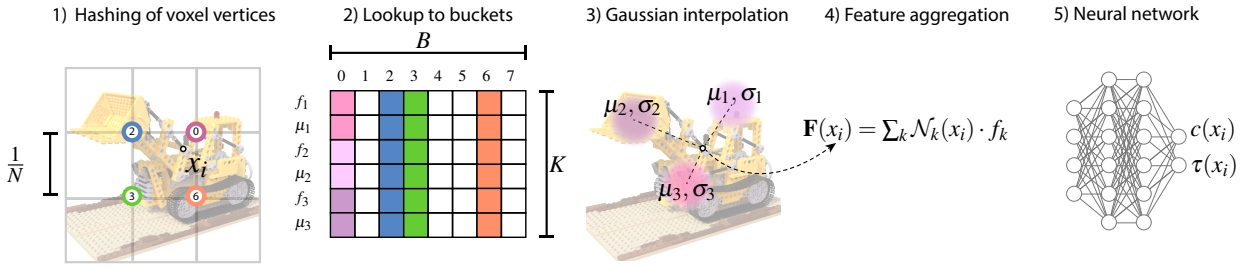


Figure 3.10: *(1) Hashing of voxel vertices:* For any given input coordinate x_i, our method identifies surrounding voxels across L Levels of detail (Lods) (Only one Lod is showed for convenience). Indices are then assigned to the vertices of these voxels, through hashing procedure. *(2) Lookup to buckets:* for all resulting corner indices, we look up the corresponding B buckets, containing K feature vector and their corresponding μ_k position. *(3) Gaussian interpolation:* We compute Gaussian weights with respect to the input position for every feature vector in the bucket. *(4) Feature aggregation:* We multiply the Gaussian weights for the feature corresponding to the feature vector and aggregate them from every level of detail. *(5) Neural Network*: the resulting concatenated features are mapped to the input domain by the Neural Network.

trainable standard deviation and mixture weights. Differently from 3DGS, we employ *isotropic* Gaussians, and the standard deviation associated with each point describes the portion of space that the feature stored alongside the point position is meant to represent.

Overview A visual overview of our representation can be found in Figure 3.10. We reviewing the hashed multi-scale representation in section 3.4.1, how features are interpolated within each level in section 3.4.1, and then detail how to augment hash buckets with a mixture-of-Gaussians representation in section 3.4.1. We discuss our training methodology in section 3.4.2, including the introduction of a loss that is critical to guide the MoG towards regions of space that require additional representation power.

Multi-scale representation

Analogous to InstantNGP [78], our architecture generates a field value $\mathcal{F}(\mathbf{x})$ at any arbitrary position $\mathbf{x}$ in space by interpolating feature vectors at the vertices of a stack $l = 1, \ldots, L$ of regular grids. The resolution of these grids follows a geometric progression: $N_l = N_{\min} \cdot b^l$, where L, $N_{\min}$, and b are hyper-parameters. This progression ensures that each level has a distinct resolution, enabling the capture of multi-scale features.

To compute the field value, we concatenate ($\oplus$) the features across all levels and pass this combined vector through a shared decoder parameterized by $\boldsymbol{\theta}$:

$$\mathcal{F}(\mathbf{x}) = \text{MLP}(\mathbf{f}_1(\mathbf{x}) \oplus \mathbf{f}_2(\mathbf{x}) \oplus \ldots \oplus \mathbf{f}_L(\mathbf{x}); \boldsymbol{\theta}), \qquad \mathcal{F}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^F \tag{3.5}$$

Per-level feature – $\mathbf{f}_l(\mathbf{x})$

Each feature $\mathbf{f}_l(\mathbf{x})$ is computed as follows: Let $\mathbf{F}$ be a tensor of features in memory, and let $\mathcal{H}(\mathbf{x})$ be an indexing function that retrieves the *indices* of $\mathbf{F}$ corresponding to the grid corners $\mathbf{v}_v$ of the field query position $\mathbf{x}$. The corresponding grid interpolation weights are denoted as α_v, where $v = 1, \ldots, V{=}2^D$.

We interpolate features at position $\mathbf{x}$ as:

$$\mathbf{f}_l(\mathbf{x}) = \sum_{v \in \mathcal{H}_l(\mathbf{x})} \alpha_v \cdot \mathbf{F}_l[v](\mathbf{x}) \tag{3.6}$$

Similarly to InstantNGP [78], we implement $\mathcal{H}_l$ as an injective map whenever $N_l < B$, and as a hash function otherwise. In other words, $\mathcal{H}_l : \mathbb{R}^D \to [1, N_l - 1]$ if $N_l < B$, and $\mathcal{H}_l : \mathbb{R}^D \to [0, B-1]$ otherwise, and respectively the feature tensor $\mathbf{F} \in \mathbb{R}^{N_l \times F}$ or $\mathbf{F} \in \mathbb{R}^{B \times F}$. As we closely follow InstantNGP [78] to enable fair comparisons, we refer the reader to this paper for additional details regarding the implementation of $\mathcal{H}$. As at finer levels v indexes the buckets of a hashing operation, we refer to $\mathbf{F}_l[v]$ as a "per-bucket" feature.

Eulerian vs. Lagrangian features. Note that, differently from InstantNGP [78], the notation in eq. (3.6) hints at the fact that the feature grid is *evaluated* at the position $\mathbf{x}$. The feature evaluation depends on the depth of the corresponding level. Consider a hierarchical hash of L levels where the last $\tilde{L}$ are Lagrangian. At shallow levels, that is $l<(\mathrm{L}-\tilde{L})$, the feature $\mathbf{F}$ follows InstantNGP [78], that is, the feature is retrieved from the hashed Eulerian grid. In other words, $\mathbf{F}_l[v](\mathbf{x}) \equiv \mathbf{F}_l[v]$. At deeper levels, we retrieve the features from a Lagrangian representation that takes the form of a *Gaussian mixture*, which we will detail next in section 3.4.1.

Per-bucket feature (Lagrangian) – $\mathbf{F}_l[v](\mathbf{x})$

To simplify notation, and without loss of generality, let us drop the bucket index $[v]$ and the level index l. Within each bucket, we store an isotropic Gaussian mixture consisting of $k = \{1,\dots,K\}$ elements, parameterized by mean $\boldsymbol{\mu}_k$, standard deviation σ_k^2, and corresponding feature vector $\mathbf{f}_k$. The feature is computed by evaluating the Gaussian mixture at position $\mathbf{x}$:

$$\mathbf{F}(\mathbf{x}) = \sum_k \mathcal{N}_k(\mathbf{x}) \cdot \mathbf{f}_k, \qquad \mathcal{N}_k(\mathbf{x}) = \frac{1}{(2\pi)^{1/2}\sigma_k} \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_k\|_2^2}{2\sigma_k^2}\right). \tag{3.7}$$

3.4.2 Training

During training, we jointly optimize the shared decoder parameters, the Eulerian representation for levels $l<(\mathrm{L}-\tilde{L})$, and the Lagrangian representation for the last $\tilde{L}$ levels. The latter includes the mean and feature, and (optionally) the standard deviation of each Gaussian. To train our representation, we optimize the loss:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}} + \lambda_{\text{guide}}\mathcal{L}_{\text{guide}} \tag{3.8}$$

where $\mathcal{L}_{\text{recon}}$ is the pixel reconstruction loss computed by volume rendering [71], evaluated via a Huber loss analogously to InstantNGP [78], $\mathcal{L}_{\text{dist}}$ is the distortion

loss proposed in [6] to promote the formation of surfaces within the volume, and $\mathcal{L}_{\text{guide}}$ avoids vanishing gradients in the optimization of the Lagrangian portion of our representation by guiding the movement of Gaussians towards surfaces.

Guidance loss. During training, whenever we back-propagate a position $\mathbf{x}$ with respect to a Gaussian whose mean $\boldsymbol{\mu}$ that is too far from $\mathbf{x}$ (scaled by the standard deviation σ^2), the computed gradients become very small, which interferes with effective optimization of our representation. Note that a very similar problem is found in the training of Gaussian Mixture Models, for which Expectation-Maximization (EM) is typically employed to address this issue [103]. In defining our loss, we take *inspiration* from EM training of GMMs. For the moment being, consider having Gaussians at a *single* level, as we will extend this later to the multi-level setting. In the *E-step*, given a query point $\mathbf{x}$ we identify the Gaussian (across buckets *and* Gaussians therein) whose PDF (scaled by the mixture weights) is maximal:

$$G(\mathbf{x}) = \alpha_{v^*} \cdot \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{k^*,v^*}, \sigma^2_{k^*,v^*}), \quad k^*, v^* = max_{k,v}\ \alpha_v \cdot \mathcal{N}_{k,v}(\mathbf{x}) \tag{3.9}$$

Then, in the *M-step*, we optimize the parameters so to minimize the discrepancy between $G(\mathbf{x})$ and the NeRF integration weights $W(\mathbf{x}) = T(\mathbf{x}) \cdot \tau(\mathbf{x})$, which is a PDF along the ray, as derived in [106, Eq. (29)]. [3] We measure the discrepancy between the two PDFs via the KL-divergence, and after dropping the subscripts (k^*, v^*) to simplify notation we note that the equality:

$$max_{\boldsymbol{\mu},\sigma^2}\ \text{KL}(W(\mathbf{x})||G(\mathbf{x})) \equiv max_{\boldsymbol{\mu},\sigma^2}\ -W(\mathbf{x}) \cdot \log(G(\mathbf{x})) \tag{3.10}$$

holds due to the definition of KL divergence, and that the term $W(\mathbf{x}) \cdot \log(W(\mathbf{x}))$ is constant with respect to the Gaussian means, and hence can be dropped. Similarly to the EM algorithm, our approach involves alternating between two key steps: 1) determining the posterior distribution of latent variables by assigning

[3] We denote density with $\tau(\mathbf{x})$ to avoid confusion with the Gaussian's variance σ^2.

each point to a Gaussian, and 2) maximizing the KL divergence based on the defined correspondence. This two-step process can be written as a loss: [4]

$$\mathcal{L}^{l}_{\text{guide}}(\mathbf{x}) = -W(\mathbf{x}) \cdot \log\left(\max_{k,v} \alpha_{v,l} \cdot \mathcal{N}_{k,v,l}(\mathbf{x})\right) \tag{3.11}$$

and as the max commutes with the (monotonic) log operator, after some straightforward algebraic manipulations, we define our loss to its final form:

$$\mathcal{L}_{\text{guide}}(\mathbf{x}) = \sum_{l} \mathcal{L}^{l}_{\text{guide}}(\mathbf{x}), \quad \mathcal{L}^{l}_{\text{guide}}(\mathbf{x}) = W(\mathbf{x}) \left(\min_{k,v} \alpha_{v,l} \cdot \frac{\|\mathbf{x} - \boldsymbol{\mu}_{k,v,l}\|_2^2}{2\sigma^2_{k,v,l}}\right) \tag{3.12}$$

Note that this loss has a very intuitive interpretation. It states that, while integrating along a ray, if we find a position $\mathbf{x}$ that is likely to lie on a surface (i.e. $W(\mathbf{x}) \approx 1$), then there should be *one* Gaussian nearby (i.e. $\boldsymbol{\mu} \approx \mathbf{x}$). Further, if we assume constant α and σ^2, note that amongst all possible Gaussians the loss will select the *closest* Gaussian. In other words, our loss is a (scaled) one-sided *Chamfer* loss between Eulerian and Lagrangian representations.

3.4.3 Implementation

We now provide an overview of the key implementation details of our method. [5] We based our NeRF implementation on the *nerfacc* framework [63] and our 2D image fitting on the Kaolin-wisp library [109]. For our experiments, we chose a feature dimension, F=2, and L=16 levels of detail (LoDs) for all the experiments in the paper as proposed in InstantNGP. As the decoder in our architecture, we employ a Multi-layer Perceptron (MLP) with one hidden layer, containing 64 neurons for all the tasks.

For initialization of our decoder, we employ a classical Xavier uniform distribution [27]. The hash table features are initialized using a normal distribution

[4] We draw *inspiration* from EM algorithm, but there are distinctions. Our implementation does not incorporate optimizable mixture weights, and we employ a *hard* assignment between points and Gaussians. Finally, each point in our model is weighted, contrasting with the EM algorithm in GMM where all points have equal weights.

[5] We provide a copy of our source code in the supplementary materials.

with zero mean and standard deviation of $1e^{-3}$. We parameterize the Gaussians with a learnable mean and a *fixed* standard deviation. The initialization of Gaussian means for the image tasks is randomized within the image space, while for Neural Radiance Field (NeRF) reconstruction, they are uniformly distributed within a sphere of radius 0.75. In our framework, the standard deviation of each Gaussian defines a field of influence for each feature vector. The standard deviations are directly related to the spatial domain of the grid vertices. We initialize the standard deviations as $50\times$ the measure of the grid cells for each LoD, and then exponentially decay their size during training to $5\times$ the measure of the grid cell for each LoD. This approach proved to be beneficial, particularly at the initial stages of training; allowing for smooth convergence of the Gaussians to regions with high surface density weights.

3.4.4 Experiments

We demonstrate the efficiency of our method in building compact representations with two distinct applications: 2D image fitting (section 3.4.4), and 3D radiance reconstruction from inverse rendering (section 3.4.4, section 3.4.4). We evaluate image reconstruction on four complex high-resolution images, and NeRF reconstruction on the Synthetic blender dataset [71] and the Tanks & Temples dataset [51]. We conclude by ablating our design choices (section 3.4.5).

2D image fitting (gigapixel)

The image fitting task involves learning a mapping between 2D coordinates and image colors, and is a popular benchmark for evaluating neural field methods capabilities in representing high-frequency signals. We train with an L2 reconstruction loss, and parameterize our models with codebook containing $B = 2^{17}$ and $B = 2^{18}$ buckets, utilize a maximum of K=4 Gaussian feature per bucket, and set the maximum resolution N_{max} to half the width of the image. For this

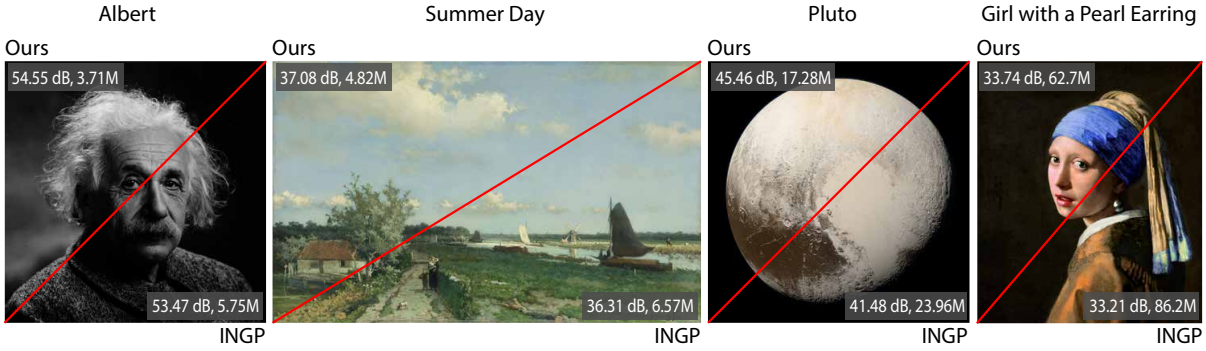


Figure 3.11: Qualitative comparisons on the giga-pixel images. On each image, we show the reconstruction quality (PSNR) together with the number of parameters.

Method	# Params	Girl	Pluto	Summer	Albert	Avg.
InstantNGP($B = 2^{19}$)	11.91M	28.73	38.59	37.26	53.81	39.60
Ours($B = 2^{17}$)	4.56M	27.60	37.53	37.08	54.55	39.19
Ours($B = 2^{18}$)	8.41M	28.83	39.72	38.93	55.35	40.71

Table 3.3: We quantitatively compare our method with InstantNGP [78] on four giga-pixel images: Girl with a Pearl Earring (Girl), Pluto, Summer Day(Summer), and Albert. We compare average PSNR↑ and average # paramaters↓.

task, the spatial gradient norm of pixel values $\|\nabla I(x_i)\|$ is employed for $W(\mathbf{x})$ in $\mathcal{L}_{\text{guide}}$ eq. (3.12), hence prioritizing representation of areas within the image containing high-frequency details.

Dataset and baselines. We evaluate our method qualitatively and quantitatively on publicly available gigapixel images, where the total number of pixels ranges from 4 M to 213 M, with InstantNGP as a representative baseline. We train both our network and InstantNGP with Adam [50] for 350 epochs with a learning rate of $1e^{-2}$ and parameters β_1=0.9, β_2=0.99, ε=10^{-15}, while the learning rate for Gaussian positions is set to $1e^{-3}$. We use a batch size of 2^{16}.

Discussion. As shown in Figure 3.15 and Table 3.3 our method reconstructs high-fidelity gigapixel images in comparison to InstantNGP while also being a 2.6× more compact representation. Note the superior performance on images characterized by localized high-frequency features (e.g. Pluto) This aspect of our method is also highlighted in the pareto plot in Figure 3.14, where our method consistently outperforms InstantNGP on different parameters counts.

Novel view synthesis

We now demonstrate the applicability of our method in a NeRF [71] setup. Differently from the 2D image fitting task, in the NeRF setting, a volumetric shape is parameterized by a spatial (3D) density function and a spatio-directional (5D) radiance. We demonstrate that our method is capable of solving this problem better than baselines while requiring fewer parameters.

Dataset and baselines. We evaluate our method qualitatively and quantitatively with InstantNGP on two widely used benchmarks: the NeRF synthetic dataset [71] and the Tanks & Temples real-world dataset [51]. For Tanks & Temples, for both our method and InstantNGP, we use mask supervision and train without background modeling. We evaluate each method both qualitatively (Figure 3.12), and quantitatively (Tables 3.4 and 3.5), using peak signal-to-noise ratio (PSNR) as the metric.

Implementation. For this task, we train our models with codebook containing $B{=}2^{17}$ and $B{=}2^{17.9}$ (i.e. we match the number of parameters of InstantNGP) buckets. We employ $K{=}4$ Gaussian features per bucket for both datasets. We evaluate our method's performance with a maximum grid resolution $N_{max}{=}1024$. We fix the weights of distortion loss (λ_{dist}) to $1e^{-3}$ for the NeRF Synthetic dataset and $1e^{-}2$ for the Tanks & Temples dataset, for both InstantNGP and our model. Additionally, we train our network with a guidance loss weight (λ_{guide})

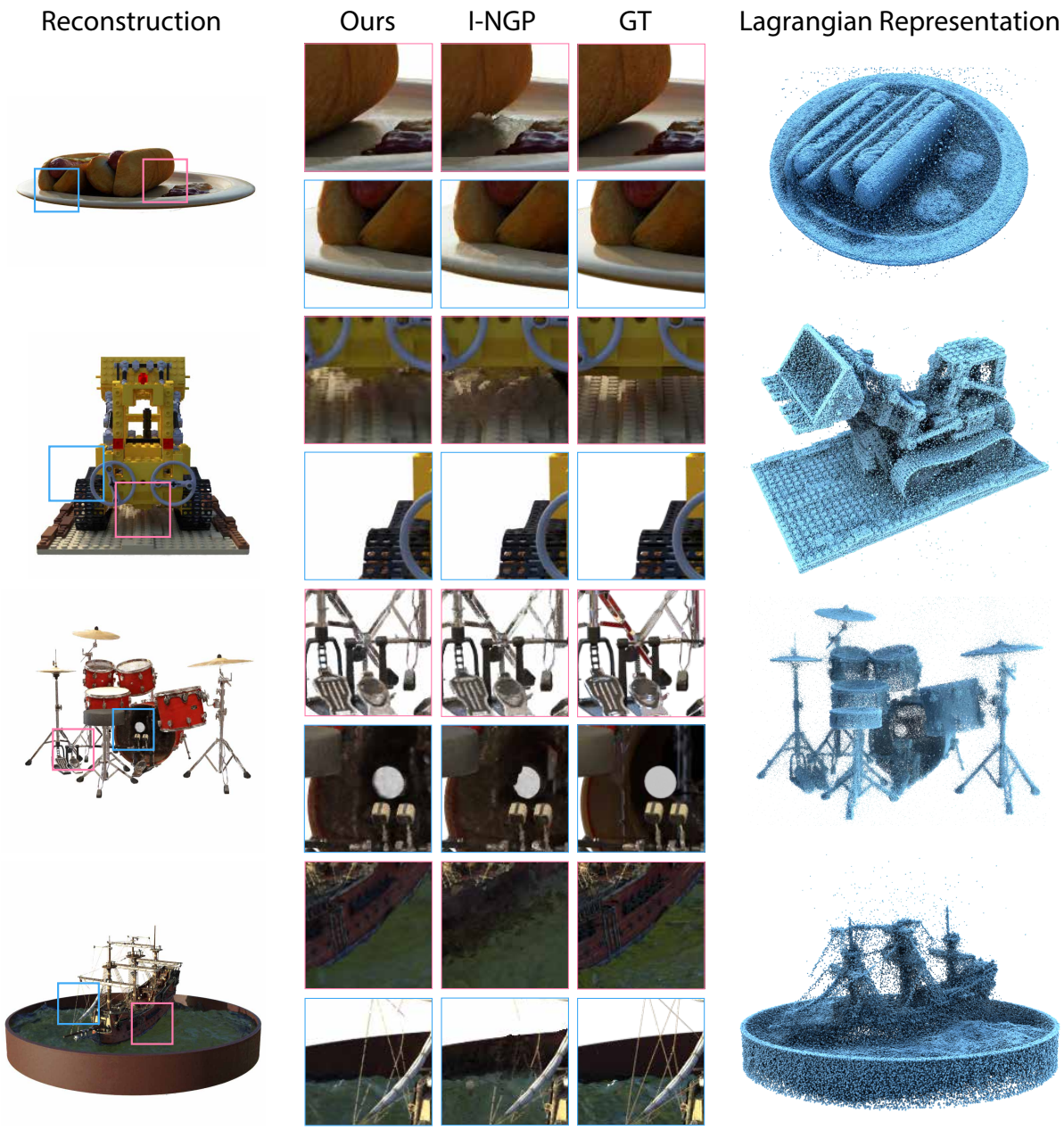


Figure 3.12: Qualitative comparisons on the Synthetic NeRF Dataset [71]. The leftmost column (reconstruction) shows the full-image results of our method and the rightmost column shows the Lagrangian Representation which is learned by our model for the particular scene.

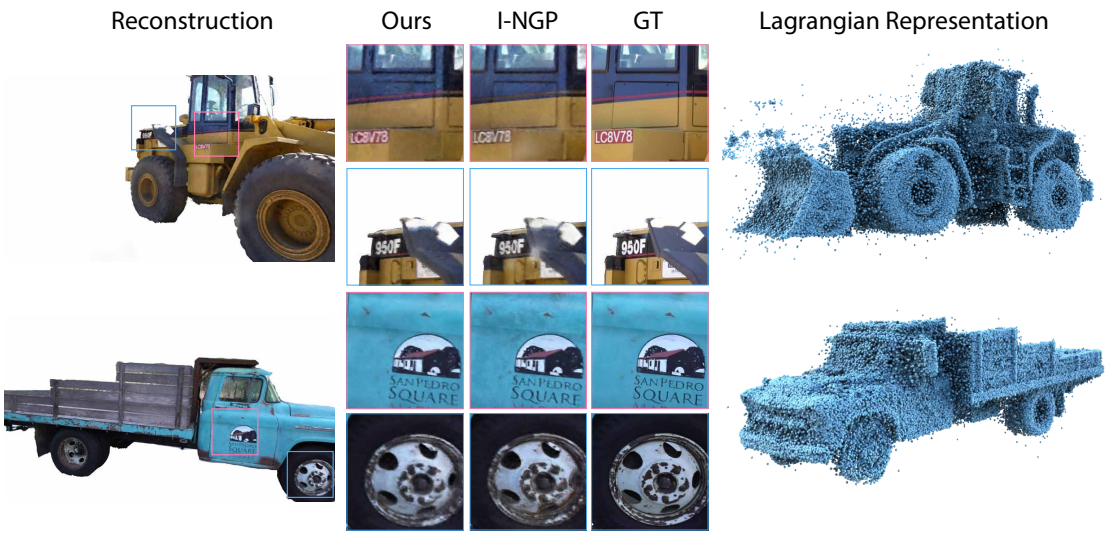


Figure 3.13: Qualitative comparisons on the Tanks and Temples dataset [51]. The leftmost column (reconstruction) shows the full-image results of our method and the Rightmost column shows the Lagrangian Representation which is learned by our model.

Method	# Params	Lego	Mic	Materials	Chair	Hotdog	Ficus	Drums	Ship	Avg.
InstantNGP($B = 2^{19}$)	12.10M	35.67	36.85	29.60	35.71	37.37	33.95	25.44	30.29	33.11
Ours($B = 2^{17}$)	6.68M	35.60	36.45	29.63	35.61	37.23	33.89	25.67	30.84	33.12
Ours($B = 2^{17.9}$)	12.13M	35.74	36.78	29.66	35.76	37.30	34.02	25.75	31.01	33.25

Table 3.4: We compare our method with InstantNGP on the Nerf Synthetic dataset [71]. We report PSNR↑ and the parameter count ↓. Our method matches the performances of this state-of-the-art method with considerably fewer parameters.

of $1e^{-1}$ and a warm-up schedule to allow for the coarse structure of the scene to be learnt first. We train both the models with an Adam optimizer for $20K$ iterations with a learning rate of $1e^{-2}$ and parameters β_1=0.9, β_2=0.99, ε=10^{-15}. The learning rate of Gaussian positions is set to $1e^{-3}$.

Discussion. As shown in Tables 3.4 and 3.5 our method matches the quality of reconstruction of InstantNGP while achieving a 1.8× more compact representation. In Figure 3.13, we qualitatively evaluate our method on real data

Method	# Params	Truck	Barn	Family	Caterpillar	Avg.
InstantNGP($B = 2^{19}$)	12.10M	27.42	27.10	33.23	26.27	28.51
Ours($B = 2^{17}$)	6.68M	27.31	27.36	33.22	26.31	28.55
Ours($B = 2^{17.9}$)	12.13M	27.38	27.66	33.36	26.33	28.68

Table 3.5: We quantitatively compare our method with InstantNGP on the Tanks and Temple dataset [51]. We compare the PSNR↑ and the parameter count ↓. Notably, we are able to match the performances of this state-of-the-art method with considerably fewer parameters.

containing complex structures and reflections. These evaluations highlight our method's superior performance in handling complex scenes. Notably, our approach demonstrates its capability to resolve collisions that, within the Instant-NGP framework, lead to the formation of micro-structures on smooth surfaces (as observed on the truck model, and wheels of the truck). In Figure 3.12, we present a visual comparison that highlights the improved rendering of thin structures, such as the legs of a drum instrument, the mast of a ship, and the structure of a hotdog. Figure 3.12 also illustrates our Lagrangian representation, where we demonstrate how the Gaussian in our model converge to regions of high surface density in 3D space.

Compact representation

We now demonstrate the efficacy of our model in developing compact representation in both Image fitting and NeRF reconstruction applications. In this experiment, we study decreasing the number of parameters to demonstrate the capability of our method to focus the limited capacity of the codebook towards capturing the high-frequency details of the signal.

Dataset and baselines. We evaluate our method quantitatively with Compact-NGP [108], a compressed variant of InstantNGP, on the NeRF synthetic dataset. Since the source code for CompactNGP is not publicly available, we compare

Method	# Params	Lego	Mic	Materials	Chair	Hotdog	Ficus	Drums	Ship	Avg.
InstantNGP($B = 2^{14}$)	0.50M	32.03	35.08	28.73	32.59	34.99	30.99	25.36	27.71	30.94
CompactNGP(from [108])	0.18M	32.31	33.88	28.32	32.05	34.26	32.05	24.71	27.71	30.66
Ours($B = 2^{11}$)	0.18M	31.15	32.65	28.52	32.44	35.67	31.98	25.07	28.26	30.72

Table 3.6: We quantitatively compare our method with CompactNGP [108] on the Nerf Synthetic Dataset [71]. We compare the PSNR↑ and the parameter count ↓. We outperform the recently published CompactNGP while matching in parameter count.

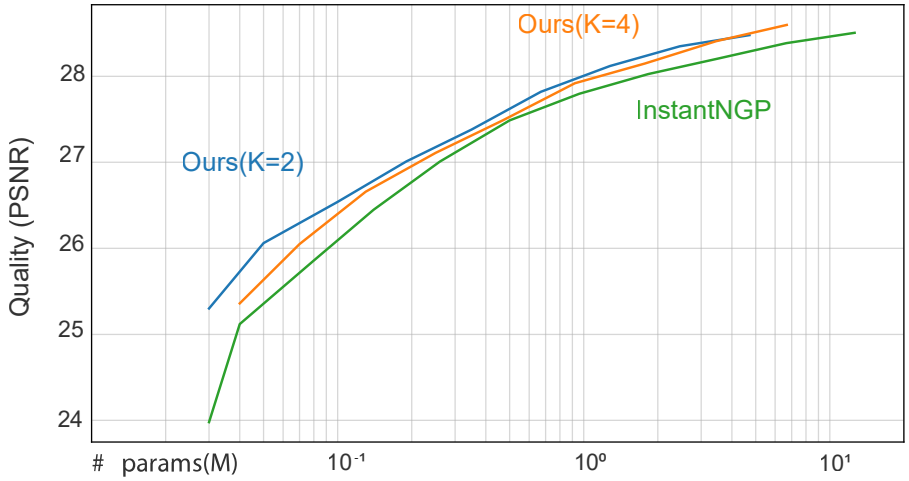


Figure 3.14: Pareto plot: Tanks and temples. We demonstrate that our method consistently outperforms InstantNGP in terms of quality vs number of parameters.

our framework with the scores directly taken from the paper. We also include in our analysis a comparison with InstantNGP, focusing on the quality of the reconstruction and compactness of the representation, across a range of hyperparameter configurations on NeRF reconstruction and 2D image fitting. For plotting our compression graph, we evaluate both our model and InstantNGP on Tanks & Temples for NeRF reconstruction and gigapixel images fitting.

Implementation. For these experiments, we train our models with codebook containing $B = 2^{11}$ buckets, and we employ a maximum of K=6 Gaussian features per bucket and evaluate the model at a maximum grid resolution N_{max}=256. We disable the distortion loss (λ_{dist}=0) for our network to enable a fair com-

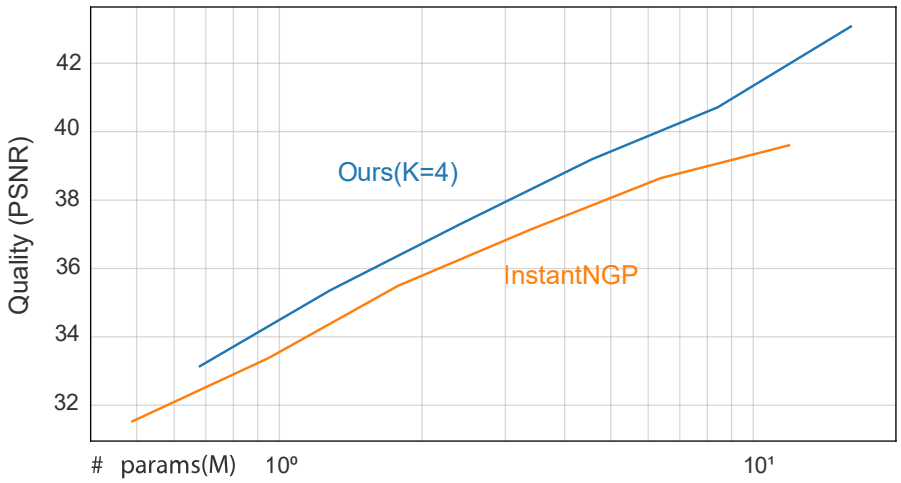


Figure 3.15: Pareto plot: Gigappixel Images. We demonstrate that our method consistently outperforms InstantNGP in terms of quality vs number of parameters.

parison with CompactNGP, that did not use this loss. Additionally, we train our network with a guidance loss weight (λ_{guide}) of $1e^{-1}$ and a warm-up scheduler. Following CompactNGP, we train our model with an Adam optimizer for $35K$ iterations with a learning rate of $1e^{-2}$ and parameters β_1=0.9, β_2=0.99, ε=10^{-15}. The learning rate of Gaussians is set to $1e^{-3}$.

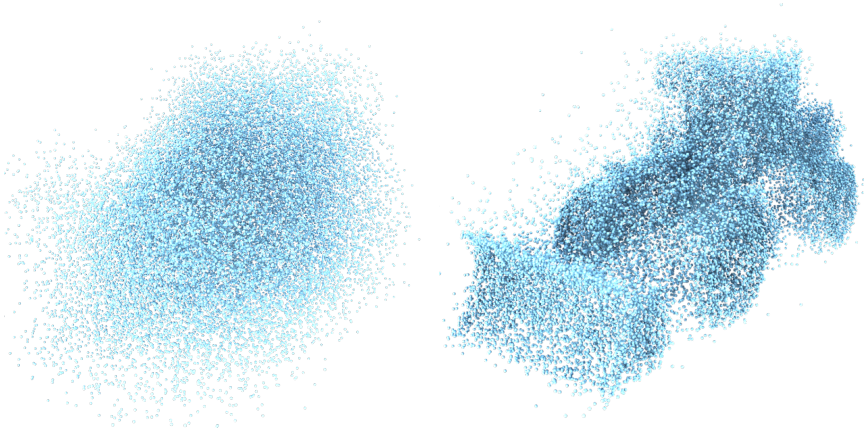
Discussion. As shown in table 3.6, our method quantitatively outperforms the recently proposed CompactNGP method on its primary task, while achieving the same level of compression, a 2.8× more compact representation when compared to InstantNGP. In Figure 3.14, we show that our pareto front lies ahead of InstantNGP in both NeRF application and image fitting, meaning that our model *consistently* outperforms InstantNGP in terms of quality vs. number of parameters.

3.4.5 Ablations

We validate our method in the NeRF reconstruction on the real-world scenes from Tanks & Temples. We investigate the importance of the losses, the number

	PSNR
Full	**27.94**
w/o $\mathcal{L}_{\text{dist}}$	27.70
w/o $\mathcal{L}_{\text{guide}}$	27.75

(a) Impact on PSNR



(b) w/o $\mathcal{L}_{\text{guide}}$ (c) w/ $\mathcal{L}_{\text{guide}}$

Figure 3.16: **Impact of losses.** We show the proposed losses are essential to have the optimal PSNR. We also show that $\mathcal{L}_{\text{guide}}$ is critical to place points onto the surface (i.e., locations in space that need more capacity to be represented), which is the key to achieve a good compression rate. The quality of the results without the guidance Loss since we fall in the Instant-NGP framework.

of Gaussian features per codebook, and the number of Lagrangian levels.

Loss function – fig. 3.16. We analyze the importance of each loss term except for $\mathcal{L}_{\text{recon}}$. Note both losses beneficially contribute to the final performance. We also qualitatively examine how losses impact point learning, which is a key to learning a compact representation – by design, we achieve a high compression rate by allocating more points near the object's surface. We observe that both losses are essential for points, and therefore features, to focus on these surfaces. Particularly, $\mathcal{L}_{\text{guide}}$ as shown in fig. 3.16, guide random points towards the surface. We also observe that $\mathcal{L}_{\text{dist}}$, as discussed in MipNeRF360 [6], eliminates floaters.

Number of Gaussian features per bucket (K) – table 3.7. We study the impact of the number of Gaussians in each bucket on reconstruction performance. And we show that 4 Gaussians achieve a good trade-off between the number of pa-

# of Gauss.	No mixture	2	**4**	8
# Params. (M)	0.50	0.67	**0.92**	1.41
PSNR	27.49	27.82	**27.94**	27.99

Table 3.7: **Num. of Gaussian (K).** We report the storage (the number of parameters) and PSNR with the different number of Gaussian in each bucket. We observe that 4 Gaussians achieve a better trade-off between performance and storage.

# of levels	0	**2**	4	8
# Params. (M)	0.5	**0.92**	1.34	2.2
PSNR	27.49	**27.94**	28.00	28.03

Table 3.8: **Num. of levels (L).** We report the storage (the number of parameters) and PSNR with the different number of Lagrangian levels. We observe that the model of 2 Lagrangian levels at the finest level Gaussians achieves the better trade-off between performance and storage.

rameters and performance. As shown in table 3.7, 8 or even more Gaussians in each bucket, while leading to better performance, start to introduce redundant parameters since the spatial collision is already well addressed with only 4 Gaussians.

Number of Lagrangian levels ($\tilde{L}$). We further investigate the impact of the number of Lagrangian levels. We observe that two Lagrangian levels in the finest levels where the majority of spatial collisions happen achieve the best trade-off between reconstruction and storage.

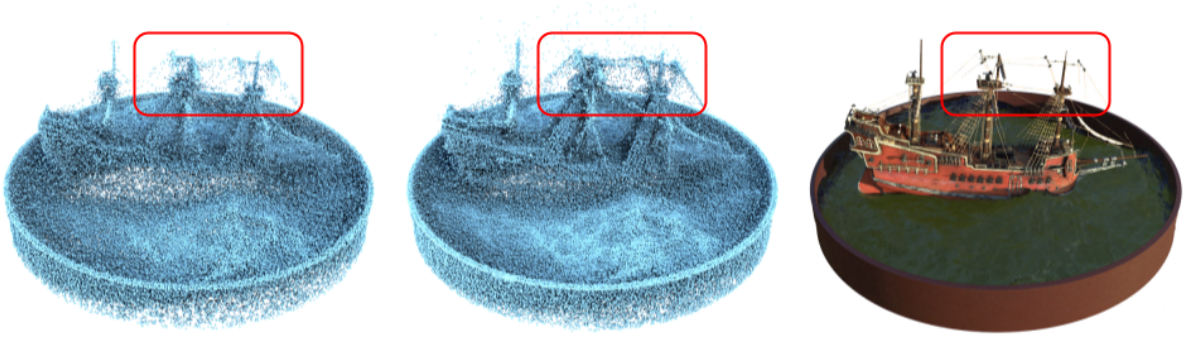
3.4.6 Discussion

We introduce a neural representation that unifies Eulerian with Lagrangian schools of thought. We take advantage of the Eulerian grids that allow the use of hash tables, which we extend with the Lagrangian point clouds, imbuing them with

the flexibility of point representations. Taking the best of both worlds, we outperform the state of the art, and provide a better PSNR - parameter count ratio than InstantNGP across various neural field workloads, including real-world scenes.

Our method is currently restricted to object-centric scenes. Our approach does not achieve an effective Eulerian-Lagrangian hybrid representation for unbounded scenes when trained with the scene contraction function. Due to our emphasis on learning a compact representation of 3D objects, we left further exploration of Lagrangian representations for unbounded scenes to future work.

We show our lagrangian representation across $\tilde{L}$ LoDs. As shown in fig. 3.17, both scales learn the complete point representation and the fine LoD scale tends to capture more high-frequency details.



Pre-final LoD Final LoD Rendered RGB image

Figure 3.17: Notice the mast of the ship(highlighted region), where we see that the final LoD represents high-frequency details better than the pre-final LoD.

Chapter 4

Comparative study for 3D Radiance Fields based methods

In recent years, the advancement of 3D reconstruction technologies has revolutionized the documentation and analysis of industrial and archaeological sites. Among these, photogrammetry has long been established as the baseline for precise, high-resolution mapping and modeling. However, the recent advent of Artificial Intelligence (AI) in the 3D field, thanks to the introduction of Neural Radiance Fields (NeRF) and more recently 3D Gaussian splatting techniques presents a novel paradigm. These powerful technologies show a potential for overcoming some of the inherent limitations faced by traditional methods. This paper provide a comprehensive comparison between these cutting-edge techniques, focusing on their application in the industrial context of excavation sites, a domain where precision in the reconstruction play a pivotal role.

Excavation sites properties introduce unique challenges for 3D reconstruction due to their dynamic nature, intricate details, and the need for precise georeferencing. The acquisition of data, requires methods that can adapt to these challenges, ensuring a high level of fidelity in the reconstruction of occluded regions. For utility companies and other stakeholders, the integration of geospatial data with 3D reconstructions facilitates a deeper understanding of the surface infrastructure, allowing for the precise location of pipes and other assets

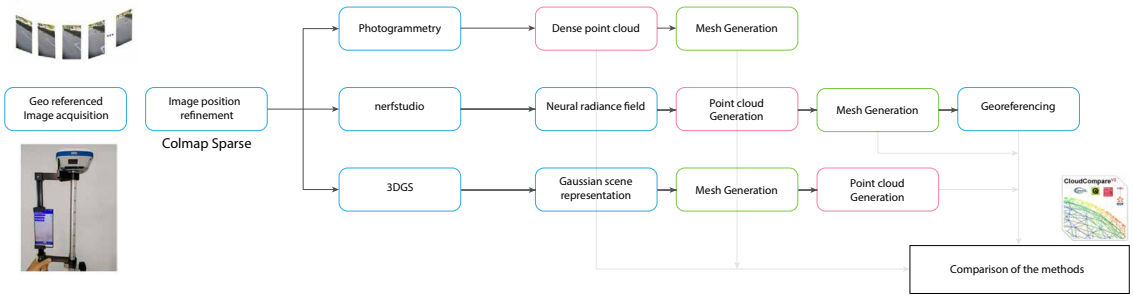


Figure 4.1: Overview of the proposed methodology

with centimeter accuracy. This capability not only aids in the efficient planning and management of utilities but also significantly reduces the risk of accidental damage during excavation activities. Moreover, the usage of georeferenced data in the 3D reconstruction enable the usage of augmented reality (AR) technologies to offer an additional layer of information, enhancing operational safety and efficiency. By overlaying digital models onto the physical world, operators can gain real-time insights, further preventing the accidental severing of critical infrastructure.

4.1 3D reconstruction methods in industrial settings

This section explores the comparative advantages and potential limitations of photogrammetry, NeRF, and Gaussian splatting in the field of excavation sites reconstruction equipped with geographical positioning information. By leveraging datasets composed of images and precise geographic coordinates, this study aims to highlight the strengths and weaknesses of each method, especially in scenarios where traditional photogrammetry may not be enough. Further, the study extends the evaluation to playground scenarios, which resemble excavation sites in terms of their unbounded spatial characteristics and presence of detailed features.

Motivation

Neural Radiance Fields based methods have recently emerged as an alternative to traditional photogrammetry in the field of image-based 3D reconstruction. This innovation is especially significant for the challenging scenarios of excavation sites, which are often characterized by their unbounded and outdoor settings, making them susceptible to complex view-dependent effects This research is motivated by the potential of NeRF to enhance the compression and precision of reconstructions in such complex settings. By comparing NeRF with traditional photogrammetry across varied scenes, this study aims to precisely assess their efficacy in capturing intricate details, surface textures, and overall geometric accuracy. Our goal is to evaluate the suitability of NeRF and related techniques to be adopted in real-world applications.

4.1.1 Background

3D reconstruction techniques are becoming increasingly important across construction, excavation, and broader worksite management fields. These methods not only facilitate comprehensive tracking of the state of the work, but also, through georeferencing and virtual reality, offer the capability for operators to virtually navigate sites, both during and after project completion. In such contexts, capturing the scene using 2D images from multiple viewpoints is a practical approach, falling under the category of multi-view reconstruction.

Among photogrammetric solutions for view synthesis, we focus on Colmap [95] for its open-access policy and continual improvements. This method enables the conversion of 2D images into comprehensive 3D models, including point clouds and textured meshes, enabling advanced spatial analyses.

However, the application of photogrammetric reconstruction encounters several challenges, particularly when dealing with objects that possess complex optical properties such as high absorbency, reflectivity, or scattering. These

methods can also be hindered by varying lighting conditions, including shadows, glare, or inconsistent illumination, as well as by surfaces with uniform or repetitive textures and complex shapes or geometries.

In this context Radiance Fields based technologies emerge as powerful solution that show the capabilities necessary for addressing some of these inherent limitations. They rely on an innovative scene representation containing particles, each of them characterized by density and color. In this study we will compare with the photogrammetry two radiance fields based techniques: Nerfacto, a variations of InstantNGP available in Nerfstudio and SuGAR a variation of 3D Gaussian Splatting.

Neural Radiance Fields Neural Radiance Fields (NeRF) have emerged as a significant advancement in the field of 3D scene reconstruction. The scene is represented with a novel 5D function. This function correlates each spatial point (x, y, z) with the radiance emitted in any direction, defined by azimuthal and polar angles (θ, ϕ). The outcome, characterized by volume density σ and RGB color values c, varies with the viewing direction.

Building this scene representation, nerfstudio [110] introduces an innovative platform designed to streamline and enhance the creation and manipulation of NeRF-based models. At the heart of NeRFStudio's methods is the Nerfacto model, which incorporate insights from a state-of-the-art research, including works such as MipNeRF-360 [6], Instant-NGP [78], and Ref- [115]. Nerfacto' focus is in optimizing camera views and sampling processes, utilizing a piecewise sampler and a proposal network sampler for efficient and effective scene reconstruction. Additionally, it employs a novel scene contraction technique and a refined NeRF field that incorporates appearance embeddings and normal predictions, all implemented in PyTorch for ease of use and customization. This method not only accelerates the reconstruction process but also improves the fi-

delity and versatility of the generated scenes.

3D Gaussian Splatting for Real-Time Radiance Field Rendering 3D Gaussian Splatting, a novel approach to scene representation, contrasts with neural fields by optimizing an explicit point-based scene model. Each point in this representation is associated with various attributes: a position $p \in \mathbb{R}^3$, opacity $o \in [0,1]$, third-degree spherical harmonics (SH) coefficients $k \in \mathbb{R}^{16}$, 3D scale $s \in \mathbb{R}^3$, and 3D rotation $R \in SO(3)$ represented by 4D quaternions $q \in \mathbb{R}^4$. Rendering to the image plane involves accumulating the color c_{GS} from correctly-sorted points using the equation:

$$c_{GS} = \sum_{j=1}^{N_p} c_j \alpha_j \tau_i \quad \text{where} \quad \tau_i = \prod_{i=1}^{j-1}(1-\alpha_i) \tag{4.1}$$

with c_j determined by SH coefficients k and α_j calculated from the projected 2D Gaussian with covariance $\Sigma' = JM\Sigma M^T J^T$, incorporating per-point opacity o, viewing transformation M, and Jacobian J of the affine approximation of the projective transformation. The 3D covariance matrix Σ ensures positive semi-definiteness through the scale matrix $S = \text{diag}(s_1, s_2, s_3)$ and rotation R, following $\Sigma = RSS^T R^T$.

Building upon the principles of 3D Gaussian Splatting, Surface Gaussian Approximation for Rendering (SuGAR) leverages Gaussian functions to model object surfaces within a scene, achieving precision in handling occlusions and detailed surface texturing through Gaussian ”splats” projected onto a volume grid. Each splat influences the volume’s density and color based on its spatial location and Gaussian distribution, described mathematically as:

$$G(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{3}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu) \right) \tag{4.2}$$

where $\mathbf{x}$ denotes a point in space, μ the mean location (center of the splat), and Σ the covariance matrix shaping the Gaussian distribution. SuGAR's method for accumulating multiple splats across a scene constructs a volumetric representation capturing density and color information, enabling precise shading and depth rendering. This approach is beneficial for visualizing medical data, archaeological site reconstructions, and more, where precision and detail are critical.

4.1.2 Methodology

This study aims to evaluate the effectiveness and potential benefits of Neural Radiance Fields (NeRF) and 3D Gaussian Splatting against traditional image-based reconstruction techniques, in the context of augmented/virtual reality applications. Our focus is on scenarios with specific challenges, including excavation sites and playground objects, which are characterized by unbounded environments and non-Lambertian surfaces or composed of fine materials.

To facilitate a direct comparison, the same dataset of images, captured with georeferencing, is utilized across all reconstruction methods. This standardized approach ensures that differences in the reconstruction quality and efficiency can be attributed solely to the methodologies rather than errors that can be attributed to the alignment of the results.

Dataset acquisition

The datasets are collected using a system comprised of two devices: a smartphone and an RTK-GNSS spatially calibrated as we can see from the figure 4.2. These devices ensure highly accurate pose information for all the images we have collected. The study aims to analyze utility case studies, therefore, as scenarios, we have selected some simple playground games mixed with real excavation scenarios where the reconstruction is more challenging. Our datasets

consist of 7 playground scenarios and 3 excavation scenarios.

Figure 4.2: System acquisition composed by Stonex and Smartphone for obtaining high accuracy geo-referenced images

Acquisition Process: Our analysis focuses on handheld acquisitions performed by operators in the field. The trajectory involves rotating around the object when it is possible, maintaining a capture at eye level with the camera facing downward. The angle varies depending on whether the scenario is an excavation or a playground game. During acquisition, the frame rate is fixed at 5 frames per second with a resolution of 1280 x 720. The accuracy of the geopose data is approximately 3 cm and less than 1 degree for each acquired image. We maintain a uniform velocity during acquisition, so the number of images for each scenario depends on the length of the trajectory. In table 4.1, we provide the main information about all the acquisitions scenarios.

Methodologies Employed

Three distinct reconstruction methodologies were applied to the captured datasets, an overview of the methodology is shown in Figure 4.1:

	Number Images	Geo tras. acc.	Resolution
Parcogiochi1	153	<2cm	1280x720
Parcogiochi2	218	<2cm	1280x720
Parcogiochi3	159	<2cm	1280x720
Parcogiochi4	149	<3cm	1280x720
Parcogiochi5	151	<3cm	1280x720
Parcogiochi6	152	<2cm	1280x720
Parcogiochi7	245	<3cm	1280x720
Scavo5	565	<3cm	1280x720
Scavo7	468	<3cm	1280x720
Scavo13	745	<3cm	1280x720

Table 4.1: From this table, we describe the number of images and their resolution for all acquisitions, then we also include, for each scene, the accuracy of translation with respect to the Geographical Coordinate System (GCS) of the images.

1. **Photogrammetry:** The classical photogrammetric procedure involves estimating camera orientation parameters for sparse point cloud construction, generating a dense point cloud. Then followed by mesh creation and texture extraction. For this purpose we used the Colmap software with all phases conducted in high-quality mode to ensure maximum detail and accuracy.

2. **NeRF-Based Reconstruction:** The training of Neural Radiance Field reconstruction requires known camera poses as input. As the software for this methodology we used *nerfstudio* [110], inside this framework we used ”nerfacto” a model strongly based on InstantNGP [78] used for its fast training and inference. We then extracted the dense point clouds and textured mesh from nerfstudio’s API, in particular for mesh extraction we expoited Poisson reconstruction.

3. **Gaussian Splatting (SuGaR):** Similarly to NeRF this methods requires know camera poses as input. This explicit model is then trained to approx-

imate the radiance field of the scene. The training of SuGAR involves more than one step. The training starts with 7k iterations of normal 3D gaussian Splatting and 7k iterations of SuGAR fine-tuning to extract a more precise geometry.

The acquisition of our dataset incorporated georeferencing, simplifying the alignment process for the reconstructions. The only exception is NeRF, which as an implicit framework normalizes its coordinates between -1 and 1. This aspect of NeRF requires an additional step to calibrate the model, to incorporate scale and translation derived from the georeferenced input to ensure accurate alignment.

For the dataset to be used in training, we first need to estimate the camera parameters from the input images. This estimation is necessary because the NeRF based techniques requires knowledge of both the camera's positions and the corresponding images to accurately generate the scene's representation. To achieve this, we utilized COLMAP, a known software for its application of Structure from Motion (SfM) techniques [104], for estimating three-dimensional structures from two-dimensional image sequences.

To facilitate comparison, given that outputs from photogrammetry are not directly comparable with those from neural fields or Gaussian splatting, we incorporate an additional conversion phase. Nerfstudio provides functionality to convert NeRF outputs into point clouds and meshes. The point clouds are easily exported since the neural representation can be inspected at any 3D point. For the meshes this conversion employs the marching cubes algorithm and the Poisson surface reconstruction method. In the SuGAR framework the mesh extraction phase it also done through marching cubes or Poisson surface reconstruction, in this case the reconstruction is enhanced thanks to the precise estimation of the normals of the sampled points. To obtain a measure of precision we derive a cloud-to-cloud comparison using the CloudCompare software.

Comparative Analysis Framework

The comparative analysis between these methods focuses on the following key metrics: **Accuracy and Detail Resolution:** Evaluating the fidelity of the reconstructed models to the original scenes. **Processing Time:** Assessing the efficiency of each methodology in terms of computational resources and time required for reconstruction. To compare the level of fidelity of the reconstructed models we propose using the point clouds generated by the studied methods. In this way we can obtain a quantitative metric. To be more specific we measure the cloud to cloud deviation of the methods based on radiance fields with respect to the recontruction using classical photogrammetry, in this case Colmap. Since this measure is an absolute value which doesn't tell which method is doing better but just the deviation from one to the other we also show the rendering results in order to see the best performing in Figure 4.4 and 4.5. In addition to this quantitative result we also propose a qualitative comparison of the resulting meshes, comparing the three proposed methodologies in Figure 4.3.

4.1.3 Discussion

We show a comparison of NeRF based techniques against traditional photogrammetry utilizing Colmap software, all models are trained on an NVIDIA RTX 3090 GPU. The assessment focuses on their effectiveness in view synthesis and 3D reconstruction, particularly in expansive, unbounded environments.
The results of our analysis highlights that the three methodologies produce high quality point clouds, with very close results especially in the fine structures of the 3D scene, as illustrated in in Figures 4.4, 4.5. Notably, NeRF's output shows a denser point cloud around high-frequency scene features but has gaps in smoother regions. In the presented comparison, the radiance field rendering results demonstrate the exceptional quality of view synthesis provided by nerfacto and SuGaR in varying scenarios. Both methods display impressive re-

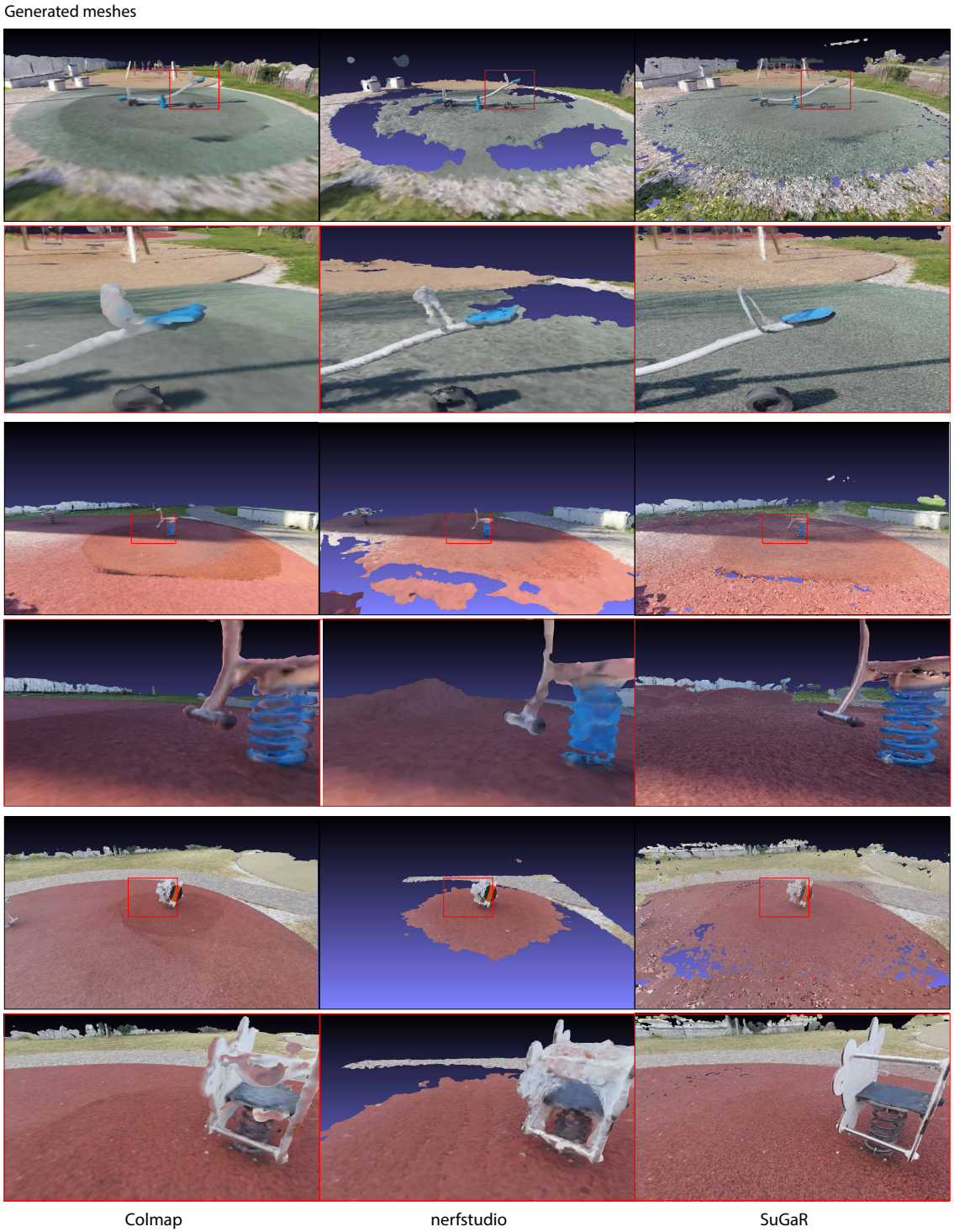


Figure 4.3: Comparison of the mesh obtained with the proposed methodologies on the playgrouds dataset.

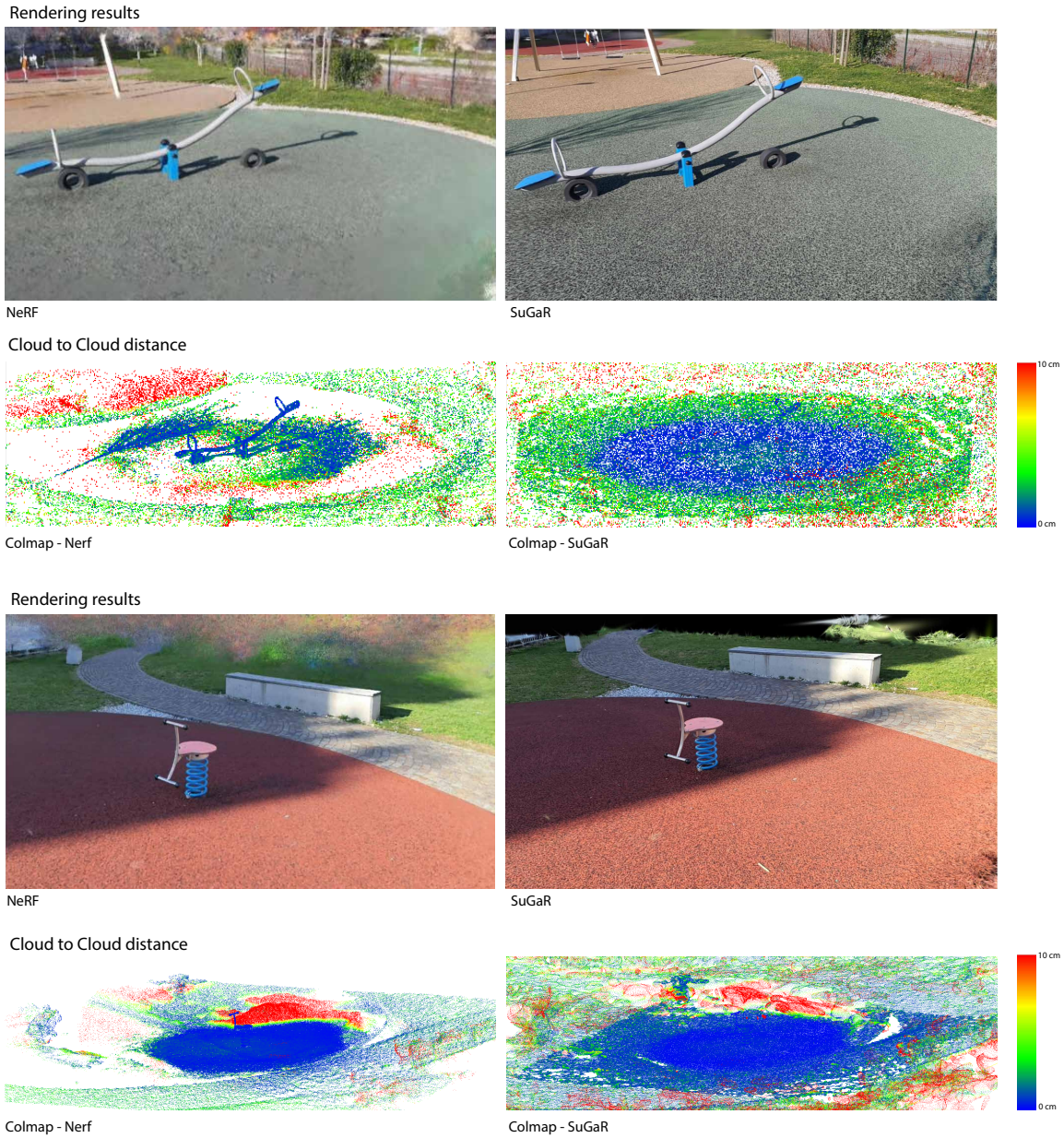


Figure 4.4: Comparison of the cloud to cloud distance of the proposed methodologies on the playground dataset.

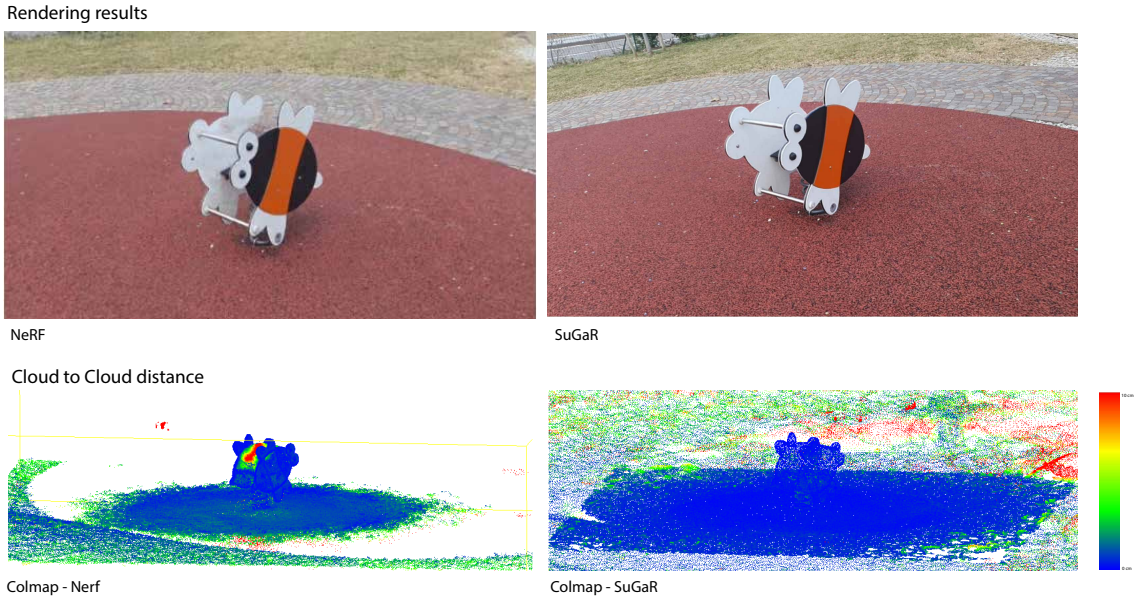


Figure 4.5: Comparison of the cloud to cloud distance of the proposed methodologies on the playground dataset.

alism, yet the results showed in Figure 4.6 point out the higher quality of SuGaR in the reconstruction of the reflections is the puddle. This highlights the capability of an explicit method like 3D Gaussian Splatting to handle view dependent effects. The comparison illustrated in Figure 4.5 highlights a failure case of nerfstudio, with a red area within the scene's object of interest indicating a high cloud-to-cloud distance. This issue not only produce a discrepancy in the point cloud representation but also results in blurring within the targeted region of the neural reconstruction.

Considering the extensive usage of meshes in VR and AR applications, for their simplicity and low memory footprint, we present a comparison of the meshes produced with the three methodologies. In Figure 4.3 we show the obtained meshes also showing a detail of the reconstruction in the region of the 3D scene with finer details. As depicted in Figure 4.7, there's a noticeable variance in detail and texture among the outputs. The colmap mesh, while being consistent, falls short on representing thin structures. In contrast, the NeRF mesh shows

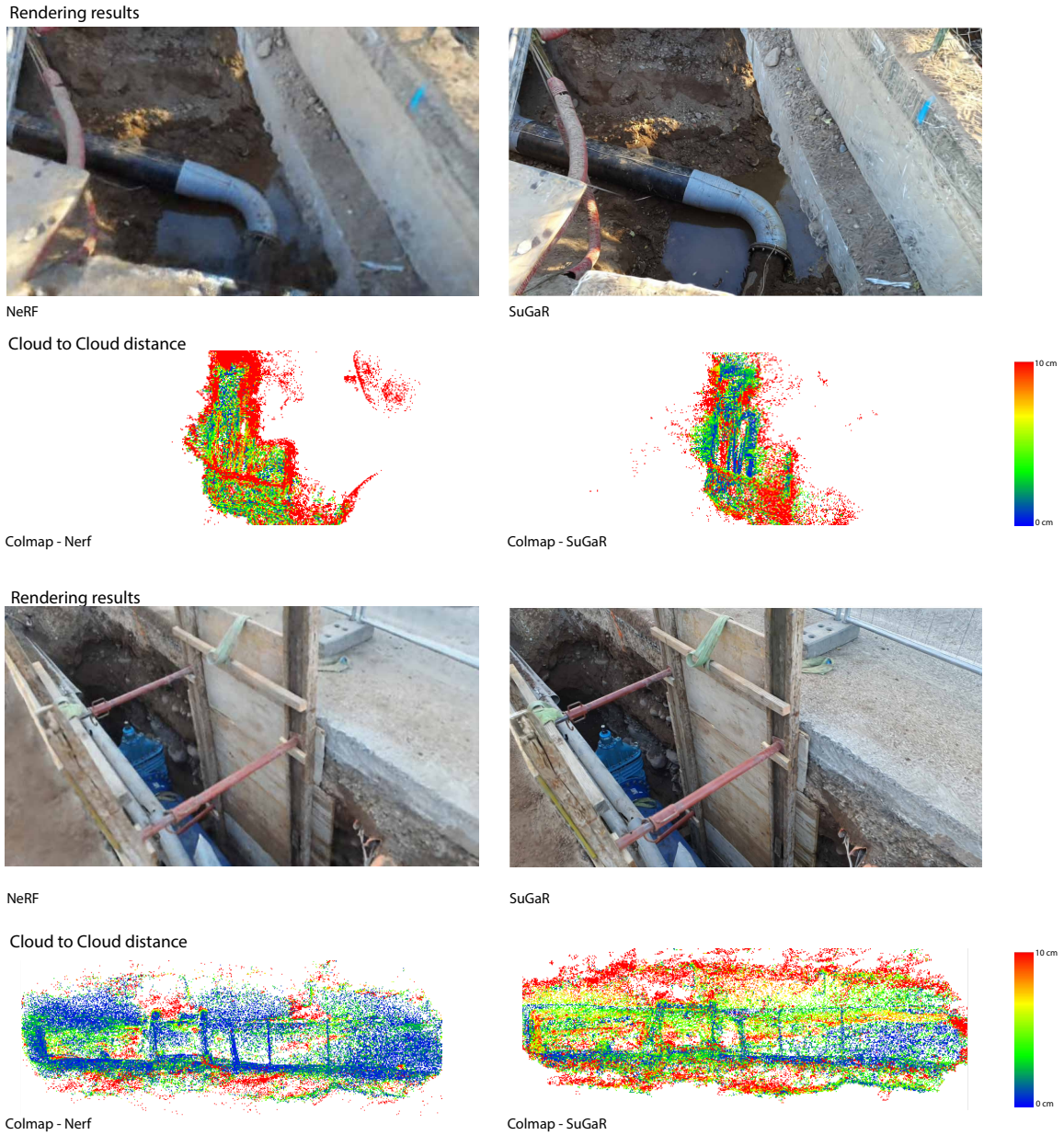


Figure 4.6: Comparison of the mesh obtained with the proposed methodologies on the excavation sites dataset.

Figure 4.7: Comparison of the mesh obtained with the proposed methodologies on the excavation sites dataset.

greater detail but presents some holes. The SuGaR mesh stands out for its superior detail, accurately capturing complex structures where others falter, thanks to its precise normal calculations.

In this paper we provide a comparative analysis of Neural radiance fields based reconstruction methods and classical photogrammetry for unbounded scenarios. We show results in playgrounds and excavations sites, to access the performances in increasingly complex scenarios In our set-up, photogrammetry has provided superior reliability in complex scenes, especially on the excavation sites. Proving also better results in modeling completely flat area which in the NeRF methods presents some artifacts. Although training/reconstruction times are generally not the main concern in the reconstruction of working areas, some application might benefit from fast reconstruction times. In this aspect nerfstudio provided the best speed in the reconstruction, requiring just 15 minutes for the training of a scene. An important aspect that needs to be analyzed is the reliance of the current rendering pipelines for virtual and augmented reality on meshes representations. This advantages the classical photogrammetry which specifically built upon the requirements of this type of output. In contrast, neural rendering technologies focus primarily on view synthesis, offering an alternative that eliminates the need for mesh generation. SuGaR and more in general 3D Gaussian Splatting techniques produce an explicit representation that allow for the splatting of gaussians in the same way traditional methods splat triangle. This feature enable SuGaR and 3D Gaussian Splatting to render the scene in real time, making it possible to use it into existing pipelines. In the future, we see 3D Gaussian Splatting to be a potential replacement for for meshes representations, especially in scenarios requiring the realistic reconstruction of complex environment containing view dependent effects.

Chapter 5

Conclusions and Future Research

In this thesis we focused on the role and the potentialities of explainable and efficient structures in two distinct macro fields: Part-whole hierarchies in neural networks and Lagrangian properties in 3D neural reconstruction. Within the context of 3D reconstruction we also conduct a comparative analysis of the latest implicit reconstruction methodologies against traditional photogrammetry techniques to assess their usability in practical, real-world settings. We introduced ”Agglomerator” and its extension ”Agglomerator++” presenting an innovative learning paradigm that draws inspiration from the brain’s cortical columns, that can learn to build parse trees of part-whole hierarchies in an unsupervised way. We showcase how these approaches enable the dynamic parsing of ”islands of agreement,” showcasing how Agglomerator and Agglomerator++ can maintain the same properties of state-of-the-art capsule networks, such as viewpoint-equivariance, but with enhanced efficiency and simplicity. We further demonstrated that the usage of a more adaptive and interpretable architecture can overcome the limitations of fast training NeRF methods, significantly enhancing the efficiency of neural implicit reconstructions. We compare our results to state-of-the-art models for 3D reconstruction to prove we can achieve a more compact representation. This results are motivated by the ability of our method to adapt to the input signal, while still leveraging the speed of fast

training NeRF methodologies, thanks to our utilization of a hierarchical hashing structure.

We believe that the development of Agglomerator-like models represents a significant step towards neural networks that more closely replicate human brain functions. The ability of parsing articulated objects in an unsupervised way with part-whole hierarchies could drastically simplify the network architecture, as well as reduce the amount of training data, inherently capturing the relationships between viewpoints and the scenes observed. As for possible future research directions, a natural progression of the presented work on 3D representation is the extension of Lagrangian Hashing to work on unbounded scenarios. The application to unbounded scenes, characterized by their significant sparsity, perfectly matches our method's capability to obtain highly compact representations. Additionally, the potential of Lagrangian Hashing to preserve spatial coherence into its representation opens up possibilities for its application in tasks requiring precise manipulation capabilities, like scene editing. We also believe that our hybrid hash architecture could be exploited for different field requiring adaptability to complex signals.

Bibliography

[1] Dolores Albarracin and Robert S Wyer Jr. The cognitive impact of past behavior: influences on beliefs, attitudes, and future behavioral decisions. *Journal of personality and social psychology*, 79(1):5, 2000.

[2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 696–712. Springer, 2020.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[4] Mayur Arvind and Mustansir Mama. Reproducibility report: Neural networks fail to learn periodic functions and how to fix it. In *ML Reproducibility Challenge 2020*, 2021.

[5] Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. Denser: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines*, 20(1):5–35, 2019.

[6] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[7] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12506–12515, 2020.

[8] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[9] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[10] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 17–141. IEEE, 2005.

[11] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 608–625. Springer, 2020.

[12] MingFang Chang, Akash Sharma, Michael Kaess, and Simon Lucey. Neural radiance field with lidar maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17914–17923, 2023.

[13] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In

International conference on machine learning, pages 1597–1607. PMLR, 2020.

[15] Xi Chen, Hongdong Zhao, Dongxu Yang, Yueyuan Li, Qing Kang, and Haiyan Lu. Sa-singan: self-attention for single-image generation adversarial networks. *Machine Vision and Applications*, 32(4):1–14, 2021.

[16] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023.

[17] Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. Sequential attention gan for interactive image editing. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4383–4391, 2020.

[18] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[19] Javier Dehesa, Andrew Vidler, Julian Padget, and Christof Lutteroth. Grid-functioned neural networks. In *International Conference on Machine Learning*, pages 2559–2567. PMLR, 2021.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[21] Zihan Ding, Xiao-Yang Liu, Miao Yin, and Linghe Kong. *Tensor Super-Resolution with Generative Adversarial Nets: A Large Image Generation Approach*, pages 209–223. 11 2019.

[22] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[24] Nicola Garau, Niccolò Bisagno, Zeno Sambugaro, and Nicola Conci. Interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13689–13698, 2022.

[25] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14346–14355, 2021.

[26] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, 2009.

[27] 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*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[28] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[29] Jonathan Granskog, Till N Schnabel, Fabrice Rousselle, and Jan Novák.

Neural scene graph rendering. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021.

[30] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[31] Markus Gross and Hanspeter Pfister. *Point-based graphics*. Elsevier, 2011.

[32] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020.

[33] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T Freeman. Unsupervised semantic segmentation by distilling feature correspondences. *arXiv preprint arXiv:2203.08414*, 2022.

[34] Jeff Hawkins. A thousand brains: A new theory of intelligence, 2021.

[35] Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in neural circuits*, 11:81, 2017.

[36] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[39] Geoffrey Hinton. Some demonstrations of the effects of structural descriptions in mental imagery. *Cognitive Science*, 3(3):231–250, 1979.

[40] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021.

[41] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[42] Geoffrey E Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75, 1990.

[43] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

[44] Kjell Jørgen Hole and Subutai Ahmad. A thousand brains: toward biologically constrained ai. *SN Applied Sciences*, 3(8):1–14, 2021.

[45] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[46] Moritz Ibing, Isaak Lim, and Leif Kobbelt. 3d shape generation with grid-based implicit functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13559–13568, 2021.

[47] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.

[48] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[49] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021.

[50] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[51] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[52] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[53] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, volume 40, pages 29–43. Wiley Online Library, 2021.

[54] Adam R Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. *arXiv preprint arXiv:1906.06818*, 2019.

[55] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[57] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021.

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[59] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004.

[60] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[61] Jiachen Li, Ali Hassani, Steven Walton, and Humphrey Shi. Convmlp: Hierarchical convolutional mlps for vision. *arXiv preprint*, 2021.

[62] P Peggy Li, Scott Whitman, Roberto Mendoza, and James Tsaio. Parvox: a parallel splatting volume rendering system for distributed visualization. In *Proceedings of the IEEE symposium on Parallel rendering*, pages 7–ff, 1997.

[63] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18537–18546, 2023.

[64] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis.

Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2021.

[65] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[66] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[67] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, (2065):20150203, 2016.

[68] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021.

[69] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[70] Vittorio Mazzia, Francesco Salvetti, and Marcello Chiaberge. Efficient-capsnet: Capsule network with self-attention routing. *arXiv preprint arXiv:2101.12491*, 2021.

[71] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[72] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[73] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[74] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[75] Christoph Molnar. *Interpretable Machine Learning*. 2019.

[76] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49(4):62, 2023.

[77] Rinat Mukhometzianov and Juan Carrillo. Capsnet comparative performance evaluation for image classification. *arXiv preprint arXiv:1805.11195*, 2018.

[78] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[79] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[80] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18419–18429, 2022.

[81] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021.

[82] Bariş Özcan, Furkan Kinli, and Furkan Kiraç. Quaternion capsule networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6858–6865. IEEE, 2021.

[83] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9054–9063, 2021.

[84] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[85] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15969–15979, 2022.

[86] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161, 2021.

[87] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021.

[88] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum*, volume 21, pages 461–470. Wiley Online Library, 2002.

[89] Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. Global illumination with radiance regression functions. *ACM Trans. Graph.*, 32(4):130–1, 2013.

[90] Daniel Rho, Byeonghyeon Lee, Seungtae Nam, Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Masked wavelet representation for compact neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20680–20690, 2023.

[91] Fabio De Sousa Ribeiro, Georgios Leontidis, and Stefanos Kollias. Capsule routing via variational bayes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3749–3756, 2020.

[92] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[93] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

[94] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 84–93, 2020.

[95] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[96] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[97] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019.

[98] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[99] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020.

[100] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019.

[101] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[102] Srinath Sridhar, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, and Christian Theobalt. Real-time hand tracking using a sum of anisotropic

gaussians model. In *2014 2nd International Conference on 3D Vision*. IEEE, 2014.

[103] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*, volume 2, pages 246–252. IEEE, 1999.

[104] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[105] Weiwei Sun, Eduard Trulls, Yang-Che Tseng, Sneha Sambandam, Gopal Sharma, Andrea Tagliasacchi, and Kwang Moo Yi. Point-nerf++: A multi-scale, point-based neural radiance field. *arXiv preprint arXiv:2312.02362*, 2023.

[106] Andrea Tagliasacchi and Ben Mildenhall. Volume rendering digest (for nerf). *arXiv preprint arXiv:2209.02417*, 2022.

[107] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021.

[108] Towaki Takikawa, Thomas Müller, Merlin Nimier-David, Alex Evans, Sanja Fidler, Alec Jacobson, and Alexander Keller. Compact neural graphics primitives with learned hash probing. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023.

[109] Towaki Takikawa, Or Perel, Clement Fuji Tsang, Charles Loop, Joey Litalien, Jonathan Tremblay, Sanja Fidler, and Maria Shugrina. Kaolin wisp: A pytorch library and engine for neural fields research. `https://github.com/NVIDIAGameWorks/kaolin-wisp`, 2022.

[110] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023.

[111] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv:2105.01601*, 2021.

[112] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192, 2021.

[113] Toon Van de Maele, Tim Verbelen, Ozan Catal, and Bart Dhoedt. Disentangling what and where for 3d object-centric representations through active inference. *arXiv preprint arXiv:2108.11762*, 2021.

[114] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

[115] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 5481–5490. IEEE, 2022.

[116] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[117] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2:37–52, 1987.

[118] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[119] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes, 2015.

[120] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[121] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[122] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked im-

age modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022.

[123] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[124] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448, 2022.

[125] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. *arXiv preprint arXiv:2311.17089*, 2023.

[126] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023.

[127] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *arXiv preprint arXiv:2311.16493*, 2023.

[128] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[129] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–12, 2022.

[130] Yi Zhang, Xiaoyang Huang, Bingbing Ni, Teng Li, and Wenjun Zhang. Frequency-modulated point cloud rendering with easy editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 119–129, 2023.

[131] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[132] Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. *arXiv preprint arXiv:2006.08195*, 2020.