UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

THIAGO SOTORIVA LERMEN

Radiance-Field Equirectangular Projection of 3D Gaussian Splats for Object Classification with Distortion-Correcting CNNs

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Claudio Rosito Jung

Porto Alegre
July 2026

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profª. Marcia Cristina Bernardes Barbosa
Vice-Reitor: Prof. Pedro de Almeida Costa
Pró-Reitor(a) de Graduação: Profª. Nádya Pesce da Silveira
Diretor do Instituto de Informática: Prof. Luciano Paschoal Gaspary
Coordenador do Curso de Ciência de Computação: Prof. Álvaro Freitas Moreira
Bibliotecária-chefe do Instituto de Informática: Alexsander Borges Ribeiro

AGRADECIMENTOS

Agradeço, antes de tudo, à minha família, que é a base de tudo o que construí. À minha mãe, Rosemary Sotoriva Machado, e à minha avó, Catarina Maria Sotoriva, sobretudo: vocês duas sustentaram cada etapa desta caminhada, abriram mão de muita coisa para que eu pudesse estudar e nunca me deixaram desistir. Devo a vocês quem eu sou. Agradeço também ao Loreno Lermen, pela presença e pelo apoio ao longo de todos estes anos.

Ao meu orientador, Prof. Dr. Cláudio Rosito Jung, minha gratidão por acreditar neste trabalho e por me guiar com tanto cuidado. Desde a iniciação científica, sua orientação moldou minha forma de pensar pesquisa e foi decisiva para que este TCC existisse.

Agradeço também ao Prof. Dr. Renato Perez Ribas pelo tempo em que trabalhamos juntos durante minha participação no PET Computação, período que contribuiu muito para a minha formação.

Aos meus amigos do conchavo da UFRGS, que tornaram a graduação muito mais leve e divertida: Gabriel Couto Domingues, Gabriel Pedroso, Airton Hoch Junior e Bruno Correa. Obrigado por todas as noites de estudo, de desespero e de risada.

Aos amigos que fiz durante o intercâmbio na ENSEEIHT, Victor João Lourenço e Vinícius Moreno Sanches, que transformaram a experiência de estudar longe de casa em algo inesquecível.

Aos meus colegas de trabalho Zoum Tou e Alae Moulay, pela parceria do dia a dia e por tudo que aprendi com vocês.

À Marta Sanchez Márquez, minha namorada, pelo carinho, pela paciência e por estar ao meu lado mesmo nos momentos mais difíceis desta reta final. Obrigado por tudo.

Por fim, agradeço à Universidade Federal do Rio Grande do Sul e ao seu Instituto de Informática pela formação que recebi, e à ENSEEIHT (Toulouse INP) pela oportunidade de intercâmbio que tanto enriqueceu minha trajetória.

ABSTRACT

3D object classification is a basic computer-vision task with applications in robotics and 3D content retrieval. 3D Gaussian Splatting (3DGS) (KERBL et al., 2023) has quickly become one of the main ways of representing 3D objects: it combines explicit storage of anisotropic Gaussians with the continuous behaviour of a radiance field. In parallel, equirectangular projection (ERP) maps spherical data onto a planar, regular grid that standard image architectures can process. The ERP does introduce a distortion that grows toward the poles, and distortion-aware CNNs such as the HSDC (STRINGHINI; JUNG, 2024a) and SWHDC (STRINGHINI; JUNG, 2024b) blocks address it with horizontally dilated shared-weight convolutions, but were only validated on geometric ERPs ray-cast from triangle meshes; whether they remain effective on an ERP generated from a volumetric radiance field was an open question.

This work answers that question with a pipeline that re-encodes the volumetric 3DGS data in a planar, image-like structure: the PLY file is converted into an $N_{\text{shells}} \times H \times W$ ERP tensor by sampling the radiance field on concentric spherical shells with EgoNeRF-style exponential spacing (CHOI; KIM; KIM, 2023). Two distortion-correction blocks are trained *from scratch* on this tensor, ResNet-34+HSDC (5.5 M parameters) and ResNet-50+SWHDC (23.6 M parameters), each against a plain ResNet of the same depth on the same input; the paired baseline isolates each block's contribution. The models are evaluated on the ModelNet10 and ModelNet40 benchmarks in their 3DGS re-release, ModelSplat (MA et al., 2025).

The best RF-ERP model is the plain ResNet-50, at 91.5% on ModelNet10 and 88.7% on ModelNet40; the correction blocks give no consistent gain: SWHDC, which adds no parameters over its baseline, trails it on both benchmarks, and HSDC beats its baseline only on ModelNet10, with about a quarter of the parameters. The 4.5–7.5 percentage-point gap to the geometric-ERP baselines comes mainly from the information loss of the radiance-field input. Against the ShapeNet-pretrained, Transformer-based Gaussian-MAE (MA et al., 2025), the pipeline trails by four to five points while using up to $4\times$ fewer parameters and no pretraining.

Keywords: 3D object classification. 3D Gaussian Splatting. equirectangular projection. distortion-correcting CNN. HSDC. SWHDC.

Projeção Equirretangular em Campo de Radiância de Gaussian Splats 3D para Classificação de Objetos com CNNs de Correção de Distorção

RESUMO

A classificação de objetos 3D é uma tarefa básica de visão computacional. O *3D Gaussian Splatting* (3DGS) (KERBL et al., 2023) combina Gaussianas anisotrópicas explícitas com a continuidade de um campo de radiância e tornou-se rapidamente uma representação padrão de objetos 3D. Em paralelo, a projeção equirretangular (ERP) mapeia dados esféricos em uma grade planar que arquiteturas de imagens convencionais processam. A ERP introduz uma distorção que cresce em direção aos polos, e CNNs com correção de distorção, como os blocos HSDC (STRINGHINI; JUNG, 2024a) e SWHDC (STRINGHINI; JUNG, 2024b), tratam-na com convoluções dilatadas horizontais de pesos compartilhados, mas validados apenas sobre ERPs geométricas obtidas de malhas por *ray-casting*; resta saber se continuam efetivos quando a ERP vem de um campo de radiância volumétrico.

Este trabalho responde essa pergunta com um *pipeline* que recodifica o 3DGS volumétrico em uma estrutura planar, semelhante a uma imagem: o arquivo PLY é convertido em um tensor ERP de $N_{\text{shells}} \times H \times W$ canais, amostrando o campo de radiância em conchas concêntricas com espaçamento exponencial no estilo EgoNeRF (CHOI; KIM; KIM, 2023). Sobre esse *tensor* treinam-se, *do zero*, ResNet-34+HSDC ($5{,}5$ M parâmetros) e ResNet-50+SWHDC ($23{,}6$ M), cada um comparado a uma ResNet simples de mesma profundidade, isolando o bloco, e avaliados no ModelNet10 e ModelNet40 em sua reedição 3DGS, o ModelSplat (MA et al., 2025).

O melhor modelo sobre a RF-ERP é a ResNet-50 simples, com $91{,}5\%$ no ModelNet10 e $88{,}7\%$ no ModelNet40; os blocos não trazem ganho consistente: a SWHDC, sem parâmetros extras sobre sua *baseline*, fica abaixo dela nos dois conjuntos, e a HSDC só supera a sua no ModelNet10, com cerca de um quarto dos parâmetros. A queda de $4{,}5$ a $7{,}5$ pontos frente às ERPs geométricas vem sobretudo da perda de informação do campo de radiância; frente ao Gaussian-MAE (MA et al., 2025), pré-treinado no ShapeNet, o *pipeline* fica quatro a cinco pontos abaixo com até $4\times$ menos parâmetros e sem pré-treino.

Palavras-chave: Classificação de objetos 3D. Gaussian Splatting 3D. Projeção equirretangular. CNN de correção de distorção. HSDC. SWHDC.

LIST OF FIGURES

Figure 1.1 Common 3D object representations, all illustrating the same chair in side view. *Top:* explicit forms: point cloud (unordered coordinates), triangle mesh (vertices and faces), and voxel grid (regular lattice). *Bottom:* implicit and hybrid forms: a signed distance function (surface at the zero level set), a neural radiance field (learned volumetric density), and 3DGS (explicit anisotropic Gaussians defining a continuous radiance field). 13

Figure 2.1 A single 3D Gaussian primitive. *Left:* the anisotropic kernel at $\boldsymbol{\mu}$, with the 1/2/3σ Mahalanobis level sets shaded and the covariance principal directions $\mathbf{v}_k$ scaled by $\mathbf{s}$. *Right:* the kernel contribution versus Mahalanobis distance D, capped by the opacity α and negligible beyond the 3σ support ($\approx 99.7\%$ of the mass). 18

Figure 2.2 Forward map from a sphere to an $H \times W$ ERP image. A ray $\mathbf{d}(\theta, \varphi)$ from the spherical camera at the centroid (left) maps to pixel (u, v) via Equations (2.4)–(2.5) (right). The equator ($\varphi = 0$) maps to the horizontal midline and the prime meridian ($\theta = 0$) to the vertical midline. 20

Figure 2.3 Latitude distortion: a fixed-size receptive field covers a smaller geodesic region near the poles than near the equator, because the horizontal pixel density grows as $1/\cos\varphi$ (STRINGHINI; JUNG, 2024a). 20

Figure 2.4 The two distortion-correction blocks. Both apply four shared-weight 3×3 convolutions with horizontal dilations $d \in \{1, 2, 3, 4\}$ and circular padding. *Top (HSDC):* concatenation, giving $4\,C_{\text{in}}$ output channels. *Bottom (SWHDC):* latitude-dependent weighted sum (weights from $R_\varphi = \min(d_{\max}, 1/\cos\varphi)$), keeping C_{in} channels with zero added parameters. 22

Figure 4.1 Pipeline overview, from 3DGS PLY input through centroid, EgoNeRF concentric shells, N_{shells}-channel RF-ERP, input transforms, HSDC/SWHDC backbone, and classification head. Stages 1–4 correspond to the RF-ERP generation of Section 4.2, stage 5 to the input transforms of Section 4.3, and stages 6–7 to the networks and classification head of Section 4.4. 28

Figure 4.2 Three ModelSplat objects (`airplane`, `car`, `flower_pot`) traced through the RF-ERP pipeline. (a) Raw 3DGS PLY as a coloured point cloud. (b) Opacity-weighted centroid $\mathbf{C}$ and the eight EgoNeRF sampling shells on the xz-projection. (c) The resulting $8 \times 256 \times 512$ RF-ERP, with the eight channels laid out as a 2×4 grid: shell 1 (innermost) at the top left through shell 8 (outermost) at the bottom right. Each panel is one channel of the tensor; brighter pixels mean higher log-compressed density integrated over that shell's radial interval, so the object's surface lights up in the shells whose interval crosses it and fades in the others, encoding a coarse near-to-far radial decomposition of the object. 30

Figure 4.3 Exponential shell spacing of Choi, Kim and Kim (2023) used by the RF-ERP pipeline. The $N_{\text{shells}} = 8$ shells are centred on the opacity-weighted centroid $\mathbf{C}$; the schedule concentrates shells in the dense near field and stretches the spacing in the far field. 32

Figure 4.4 Empirical justification for the log compression. (a) The raw ModelNet10 shell density is heavy-tailed: a spike of empty pixels and a long tail whose maximum is about $8\times$ the 99th percentile (log-scaled counts). (b) After $\log(1+\rho)$ the populated densities occupy a bounded band. (c) The transfer curve is steepest near $\rho = 0$, amplifying low-density boundaries and compressing the tail. 35
Figure 4.5 HSDCNet (ResNet-34 + HSDC) for 10-channel RF-ERP input: one HSDC block in the stem, two HSDC blocks per residual stage, GAP+FC head; ~ 5.5 M parameters. 38
Figure 4.6 SWHDCResNet (ResNet-50 + SWHDC) for 10-channel RF-ERP input: every 3×3 convolution inside each bottleneck is replaced by an SWHDC block; head matches HSDCNet; ~ 23.6 M parameters. 39

Figure 5.1 Per-class training-sample count on the official ModelNet10 (3,991 objects) and ModelNet40 (9,843 objects) splits used by ModelSplat. Both are moderately imbalanced: the largest class (`chair`) has roughly an order of magnitude more samples than the smallest. 44
Figure 5.2 Training and validation curves for the eight runs, grouped by benchmark (top: ModelNet10, bottom: ModelNet40; left: loss, right: accuracy). Within each panel the four runs of that benchmark are overlaid; validation curves are solid and training curves are faded. 48
Figure 5.3 Per-class recall of each block against its plain-ResNet baseline on the same RF-ERP input. Top: ModelNet10; bottom: ModelNet40. Left panels: ResNet-34 + HSDC vs. plain ResNet-34; right panels: ResNet-50 + SWHDC vs. plain ResNet-50. Classes are sorted by the block's recall. 51
Figure 5.4 Parameter–accuracy Pareto plot on ModelNet10 (left) and ModelNet40 (right): parameter count (log scale) versus test overall accuracy; the RF-ERP variants of this work occupy the low-parameter region of the front. 53
Figure 5.5 Per-class RF-ERP gallery on ModelNet10. One representative object per category; the first eight columns show the log-compressed density shells produced by the pipeline of Section 4.2, from the innermost (shell 1, left) to the outermost radius (shell 8), and the two rightmost columns show the derived pseudo-depth and MIP channels of Section 4.3.2; brighter pixels indicate higher values. 54
Figure 5.6 Test-set confusion matrices on ModelNet10. Left: HSDC. Right: SWHDC. The matrices are row-normalised so that each row sums to one. 55
Figure 5.7 Test-set confusion matrices on ModelNet40. Left: HSDC. Right: SWHDC.55

LIST OF TABLES

Table 3.1 ERP-based 3D object classification baselines, reported on geometric (mesh ray-cast) ERP. Accuracy figures are taken directly from the cited works.....25
Table 3.2 Selected results from Ma et al. (2025), reported with 1024 input splats per object and full encoder fine-tuning. C, O, S, R, SH denote centroid, opacity, scale, rotation and spherical-harmonic colour coefficients respectively.....26
Table 3.3 Taxonomy of 3D object classification methods on ModelNet by classifier input modality, architectural family and pretraining regime. "3DGS-RF-ERP" denotes the radiance-field equirectangular projection of this work; "SSL" stands for self-supervised learning (pretraining without labels).....27

Table 4.1 Single-sample augmentation primitives applied during training. Each primitive fires independently with the indicated probability. 37

Table 5.1 Headline statistics of the ModelSplat dataset used in this work. 42
Table 5.2 ModelNet10 and ModelNet40 sample counts. The training directory is split internally into a training and a validation set with a class-stratified $80:20$ hold-out partition seeded by 42.....43
Table 5.3 Hyperparameters shared by all four experimental configurations.....45
Table 5.4 Hyperparameters that vary across the four block-variant configurations. Each also has a plain-ResNet ablation baseline with identical settings but standard convolutions in place of the block (≈ 21.3 M for ResNet-34, ≈ 23.6 M for ResNet-50). 45
Table 5.5 Test-set results on ModelNet10. All four backbones consume the 10-channel RF-ERP and are trained from scratch (Section 4.5). Each block variant is paired with a plain-ResNet baseline of the same depth; SWHDC is capacity-matched to its baseline. Evaluation uses the highest-validation-accuracy checkpoint. 46
Table 5.6 Test-set classification results on ModelNet40. Same setup as Table 5.5. 47
Table 5.7 Convergence speed of the eight runs: epoch at which the EMA-smoothed validation curve first reaches the given fraction of that run's peak validation accuracy. "Peak val. OA" and "Peak epoch" are the raw best-checkpoint values used for evaluation. 49
Table 5.8 Comparison with the geometric-ERP baselines of the original HSDC (STRINGHINI; JUNG, 2024a) and SWHDC (STRINGHINI; JUNG, 2024b) papers. "Geometric ERP" is the ray-cast ERP of those papers, "RF-ERP" the radiance-field ERP of this work; Δ_{OA} is the block's accuracy difference versus its paper baseline.50
Table 5.9 The RF-ERP pipeline of this work against representative ModelNet classifiers from the four main representation families ("SSL" = self-supervised pretraining on ShapeNet). Reference accuracies are the single-model figures reported by the cited works; "—" marks a value not reported for that benchmark. 52

LIST OF ABBREVIATIONS AND ACRONYMS

3DGS	3D Gaussian Splatting
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
ERP	Equirectangular Projection
GAP	Global Average Pooling
HSDC	Horizontally Stacked Dilated Convolutions
MAE	Masked Autoencoder
MIP	Maximum Intensity Projection
MLP	Multilayer Perceptron
MN10	ModelNet10
MN40	ModelNet40
NeRF	Neural Radiance Field
OA	Overall Accuracy
PLY	Polygon File Format
RF	Radiance Field
RF-ERP	Radiance-Field Equirectangular Projection
RQ	Research Question
SH	Spherical Harmonics
SSL	Self-Supervised Learning
SWHDC	Spherically-Weighted Horizontally Dilated Convolutions
TTA	Test-Time Augmentation
ViT	Vision Transformer
YAML	YAML Ain't Markup Language

CONTENTS

1 INTRODUCTION **12**
1.1 Motivation **12**
1.2 Goals and Contributions **14**
1.3 Research Questions **16**
1.4 Document Organisation **16**
2 THEORETICAL BACKGROUND **17**
2.1 Gaussian Splatting **17**
2.2 Spherical Images **18**
2.2.1 Equirectangular Projection 19
2.2.2 CNNs for ERPs 21
3 RELATED WORK **23**
3.1 Point-Based Methods **23**
3.2 Voxel-Based Methods **23**
3.3 View-Based Methods **24**
3.4 3DGS-Based Classification **25**
3.5 Positioning of This Work **26**
4 METHODOLOGY **28**
4.1 Pipeline Overview **28**
4.2 Radiance-Field ERP Generation **29**
4.2.1 Opacity-Weighted Centroid 31
4.2.2 ERP Ray Directions 31
4.2.3 Shell Radii via EgoNeRF Spacing 31
4.2.4 Per-Shell Radial Integration 32
4.2.5 3DGS Density Evaluation and Spatial Culling 33
4.3 Classifier Input Transforms **34**
4.3.1 Log Compression 35
4.3.2 Derived Channels: Pseudo-Depth and MIP 35
4.3.3 Single-Sample Augmentation 36
4.3.4 Sample-Pair Augmentation 37
4.4 Network Architectures **37**
4.4.1 ResNet-34 + HSDC 38
4.4.2 ResNet-50 + SWHDC 39
4.4.3 Classification Head 39
4.5 Training Protocol **39**
4.6 Evaluation Protocol **40**
5 RESULTS **42**
5.1 Experimental Setup **42**
5.1.1 Data Source: ModelSplat 42
5.1.2 Hardware, Software, and Preprocessing Cost 43
5.1.3 Hyperparameter Configurations 44
5.2 Classification Accuracy **46**
5.3 Training Dynamics **47**
5.4 Comparison with Prior Work **49**
5.4.1 Geometric-ERP Baselines 49
5.4.2 The Broader ModelNet Literature 50
5.5 Parameter Efficiency Analysis **52**
5.6 Qualitative Analysis **53**
5.6.1 Confusion Matrices 54

5.6.2 Failure Cases 55
6 CONCLUSION 57
6.1 Summary of Contributions 57
6.2 Limitations 58
6.3 Future Work 59
REFERENCES 61

1 INTRODUCTION

3D object classification is a core task in computer vision, with applications in robotics, augmented reality, and 3D content retrieval (GUO et al., 2021). The choice of 3D representation, however, has changed substantially in recent years: meshes and point clouds from ModelNet (WU et al., 2015) and ShapeNet (CHANG et al., 2015) dominated the field for nearly a decade, but neural radiance fields (MILDENHALL et al., 2020) and, more recently, 3D Gaussian Splatting (3DGS) (KERBL et al., 2023) have become standard ways of representing 3D objects. ModelNet, in particular, is the reference benchmark of this task: a repository of $12{,}311$ CAD meshes of everyday objects released by Wu et al. (2015), organised into 40 categories (ModelNet40) with a 10-category subset (ModelNet10) and an official train/test partition, against which virtually every classifier surveyed in Chapter 3 reports its accuracy. This chapter motivates the work, states its goals and contributions, formulates the research questions, and outlines the organisation of the document.

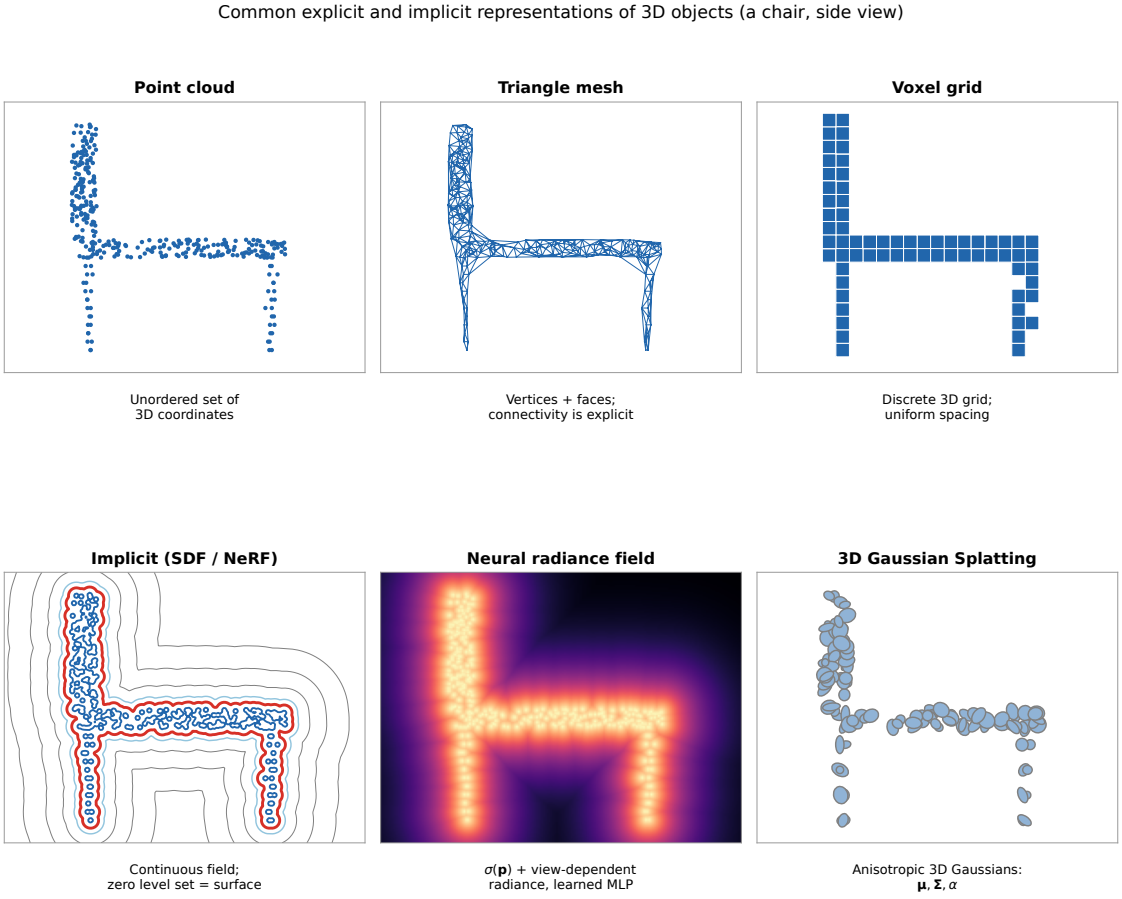
1.1 Motivation

3D object classification maps a geometric representation of an object to a semantic category. It underpins downstream systems such as category-specific robotic grasping, AR scene understanding, and retrieval over CAD libraries (GUO et al., 2021).

How the object is represented largely determines which network families can process it. Figure 1.1 illustrates the main options on the same simple object, a chair seen from the side. The classical *explicit* representations are the point cloud, an unordered set of surface samples consumed by permutation-invariant networks such as PointNet (QI et al., 2017a; QI et al., 2017b); the triangle mesh, which adds surface topology and is the interchange format of CAD repositories such as ModelNet (WU et al., 2015); and the voxel grid, a regular 3D lattice that 3D CNNs such as VoxNet (MATURANA; SCHERER, 2015) can convolve at the cost of cubic memory growth. A more recent family encodes the object *implicitly*, as a continuous function of space: signed distance functions (PARK et al., 2019), occupancy fields (MESCHEDER et al., 2019), and neural radiance fields (NeRF) (MILDENHALL et al., 2020).

3DGS (KERBL et al., 2023) sits between the two families: it stores an explicit, enumerable set of primitives, but each primitive is a continuous anisotropic 3D Gaussian,

Figure 1.1: Common 3D object representations, all illustrating the same chair in side view. *Top:* explicit forms: point cloud (unordered coordinates), triangle mesh (vertices and faces), and voxel grid (regular lattice). *Bottom:* implicit and hybrid forms: a signed distance function (surface at the zero level set), a neural radiance field (learned volumetric density), and 3DGS (explicit anisotropic Gaussians defining a continuous radiance field).



Source: the author.

so the collection defines a continuous radiance field (RF) similar to that of a NeRF. Fast differentiable rendering, small storage footprint, and photo-realistic novel-view synthesis have made 3DGS the default output of many reconstruction pipelines, and the ShapeSplats benchmark (MA et al., 2025) has reissued the whole ModelNet hierarchy (WU et al., 2015) in 3DGS form. 3DGS is therefore likely to join point clouds and meshes as a standard input modality for 3D classifiers.

In parallel, 2D convolutional networks have been adapted to 3D inputs by projecting the object onto an *equirectangular projection* (ERP), a 360° panoramic image obtained by sampling the unit sphere at a regular grid in the spherical coordinates (θ, φ) (SHI et al., 2015; CAO; HUANG; KARTHIK, 2017; STRINGHINI; JUNG, 2024a). The ERP lets any off-the-shelf 2D CNN process a full view of the object in a single image, but it is periodic along the horizontal axis and its horizontal pixel density grows towards the poles, which distorts the receptive field of ordinary convolutions. The HSDC (STRINGHINI; JUNG, 2024a) and SWHDC (STRINGHINI; JUNG, 2024b) blocks correct these effects with shared-weight horizontally dilated convolutions and circular padding: built on top of ResNet-34 and ResNet-50 respectively, they reach 97.1% / 93.9% and 94.1% / 91.9% on ModelNet10 / ModelNet40, trained from scratch. Both blocks, however, were validated only on *geometric* ERPs synthesised by ray-casting a triangle mesh from an internal viewpoint.

These two trends, the rise of 3DGS and the success of distortion-correcting CNNs on geometric ERP, meet naturally if the 3DGS representation can be converted into an ERP image that exposes its volumetric structure. This is the central theme of this work: *representing volumetric 3DGS data in a planar, image-structured form* so that standard 2D CNNs can process it directly. The empirical question of this work is whether the existing distortion-correcting backbones still work on such an input, and how the resulting classifier compares with recent Transformer-based 3DGS classifiers.

1.2 Goals and Contributions

The general goal of this work is to evaluate whether the HSDC and SWHDC distortion-correction blocks, originally validated on geometric ERP, remain effective on a radiance-field ERP (RF-ERP) generated from a 3DGS representation. This goal breaks into four specific objectives:

1. **Build a preprocessing pipeline** that converts pre-trained 3DGS PLY files into multi-channel RF-ERP tensors by sampling the RF on concentric spheres with EgoNeRF-style exponential spacing (CHOI; KIM; KIM, 2023).
2. **Evaluate planar and distortion-aware CNNs** trained from scratch on this input, with no ImageNet or ShapeNet pretraining: the distortion-aware ResNet-34 + HSDC (STRINGHINI; JUNG, 2024a) and ResNet-50 + SWHDC (STRINGHINI; JUNG, 2024b), each paired with a plain (planar) ResNet baseline of the same depth trained identically. Training from scratch isolates the effect of the input modality from pretraining, and the planar baselines isolate the contribution of each distortion-correction block.
3. **Benchmark** the resulting classifiers on the ModelNet10 and ModelNet40 subsets of ModelSplat (MA et al., 2025), using the official train/test partition of Wu et al. (2015).
4. **Compare** the resulting accuracies with the geometric-ERP baselines of the original HSDC and SWHDC papers (same backbone, different input modality) and with the Transformer-based 3DGS classifiers of Ma et al. (2025), Yu et al. (2022), Pang et al. (2022) (same input domain, different architectural family).

These objectives translate into the main contributions of this work. First, to the best of our knowledge, the RF-ERP pipeline of Chapter 4 is the first to convert a 3DGS representation into an equirectangular projection by sampling the underlying RF on exponentially spaced concentric shells. Second, Chapter 5 delivers the first controlled evaluation of these blocks on a 3DGS-derived input, each against a plain-ResNet baseline trained from scratch on the same RF-ERP. The blocks train stably and reach competitive accuracy, but the ablation shows that their distortion-correction bias does not consistently beat a plain CNN on this noisier input: SWHDC trails its capacity-matched baseline on both benchmarks, and HSDC improves on its baseline only on ModelNet10. Third, the HSDC variant is the parameter-efficient corner of the RF-ERP front: it stays within about 5 percentage points of Gaussian-MAE (MA et al., 2025) on ModelNet10 while using roughly $4\times$ fewer parameters and no pretraining. Finally, the preprocessing, training, and evaluation code, the experiment configurations, and the analysis notebooks are released as open source.

1.3 Research Questions

The HSDC and SWHDC blocks were validated on geometric ERPs, where each pixel encodes a surface property at the first ray–mesh intersection. A 3DGS file has no surface, only a continuous volumetric density, so the natural encoding samples that density at several radial offsets per ray and yields a multi-channel RF-ERP. This input replaces the sharp surface boundary with a soft density signal and carries the floater and reconstruction noise of the optimised Gaussians (MA et al., 2025), so it is not obvious that blocks designed for sharp-surface depth ERPs transfer to it. This work raises three research questions (RQ):

- **RQ 1:** Can a 3DGS representation be summarised into a single multi-channel ERP image that exposes its volumetric structure to a planar CNN?
- **RQ 2:** When trained from scratch on such an RF-ERP, do the HSDC and SWHDC distortion-correction blocks remain effective, and do they still improve on a plain CNN of the same depth?
- **RQ 3:** How does the resulting classifier compare with the geometric-ERP baselines of the original papers and with current Transformer-based 3DGS classifiers?

If HSDC and SWHDC generalise, the parameter-efficient ERP-based CNNs can be reused on 3DGS inputs with little extra effort; if not, the case for Transformers over Gaussian parameters, as in Gaussian-MAE (MA et al., 2025), becomes stronger.

1.4 Document Organisation

This document is organised as follows. Chapter 2 reviews the theoretical background: the 3DGS representation and spherical images, including the ERP and the HSDC and SWHDC convolutions. Chapter 3 surveys 3D classification methods by input type and positions this work against the emerging 3DGS classifiers. Chapter 4 describes the RF-ERP pipeline, the architectures, and the training and evaluation protocols; Chapter 5 reports the results against geometric-ERP and 3DGS baselines; and Chapter 6 closes with contributions, limitations, and future work.

2 THEORETICAL BACKGROUND

This chapter reviews the two building blocks of the proposed pipeline. Section 2.1 introduces the 3D Gaussian Splatting representation and the volumetric density that the pipeline samples. Section 2.2 introduces spherical images: the equirectangular projection, its latitude distortion, and the HSDC and SWHDC convolutions that adapt standard CNNs to this projection.

2.1 Gaussian Splatting

3DGS (KERBL et al., 2023) represents a scene as a set of N continuous, anisotropic Gaussian primitives. Each primitive is parameterised by its centre $\boldsymbol{\mu} \in \mathbb{R}^3$; an opacity $\alpha \in [0, 1]$ that bounds its contribution to the volumetric density; a per-axis scale $\mathbf{s} \in \mathbb{R}^3_{>0}$ and a unit quaternion $\mathbf{q}$ that stretch and orient the kernel; and a set of spherical-harmonic (SH) coefficients that encode its view-dependent colour (RAMAMOORTHI; HANRAHAN, 2001). The geometric parameters combine into the 3×3 covariance matrix

$$\boldsymbol{\Sigma} \;=\; \mathbf{R}\,\mathrm{diag}(\mathbf{s})^2\,\mathbf{R}^\top, \tag{2.1}$$

where $\mathbf{R} \in \mathrm{SO}(3)$ is the rotation matrix derived from $\mathbf{q}$ (KERBL et al., 2023, Eq. 6). The unnormalised density of one primitive at a query point $\mathbf{p}$ decays with the Mahalanobis distance from $\mathbf{p}$ to $\boldsymbol{\mu}$,

$$g(\mathbf{p};\,\boldsymbol{\mu},\boldsymbol{\Sigma}) \;=\; \exp\!\left(-\tfrac{1}{2}\,(\mathbf{p}-\boldsymbol{\mu})^\top\,\boldsymbol{\Sigma}^{-1}\,(\mathbf{p}-\boldsymbol{\mu})\right), \tag{2.2}$$

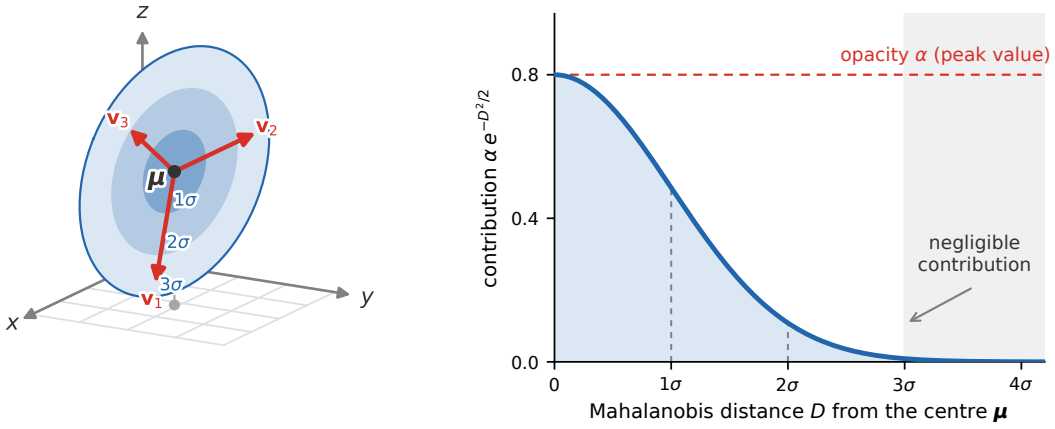
and the volumetric density of the whole scene is the opacity-weighted superposition of all primitives,

$$\rho(\mathbf{p}) \;=\; \sum_{i=1}^{N} \alpha_i\, g(\mathbf{p};\,\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i). \tag{2.3}$$

Figure 2.1 illustrates the anatomy of a single primitive. The left panel shows the anisotropic kernel in space: the shaded regions are the level sets of the Mahalanobis distance $D = 1$, 2, and 3 around the centre $\boldsymbol{\mu}$, and the arrows are the principal directions $\mathbf{v}_k$ of $\boldsymbol{\Sigma}$ (the columns of $\mathbf{R}$), each stretched by its per-axis scale s_k, which is what makes the kernel anisotropic. The right panel plots the corresponding contribution profile $\alpha\, e^{-D^2/2}$ along

the Mahalanobis distance: the opacity α caps the peak value at the centre, the contribution decays smoothly with D, and beyond the 3σ level set, which encloses $\approx 99.7\%$ of the kernel mass, it is negligible. This 3σ support is the same cutoff used by the 3DGS rasteriser of Kerbl et al. (2023) and is reused by the spatial-culling step of Section 4.2.5.

Figure 2.1: A single 3D Gaussian primitive. *Left:* the anisotropic kernel at $\boldsymbol{\mu}$, with the $1/2/3\sigma$ Mahalanobis level sets shaded and the covariance principal directions $\mathbf{v}_k$ scaled by $\mathbf{s}$. *Right:* the kernel contribution versus Mahalanobis distance D, capped by the opacity α and negligible beyond the 3σ support ($\approx 99.7\%$ of the mass).



Source: the author.

For rendering, Kerbl et al. (2023) project each Gaussian onto the image plane and accumulate the contributions along each ray with classical α-compositing, evaluated in real time by a differentiable rasteriser. The pipeline of Chapter 4 does not use the rendering equation: it samples the volumetric density of Equation (2.3) directly and ignores the view-dependent SH colour coefficients.

A trained 3DGS scene is stored as a Polygon File Format (PLY) file in which each Gaussian is a vertex carrying the raw parameters optimised during training, kept in unbounded form so that gradient updates remain unconstrained: the opacity is stored in logit space and recovered by a sigmoid, the per-axis scale is stored in log space and recovered by an exponential, and the rotation quaternion is renormalised to unit length on loading (KERBL et al., 2023).

2.2 Spherical Images

A spherical (omnidirectional) image assigns a value to every direction around an observation point, covering the full $360° \times 180°$ field of view (SILVEIRA et al., 2022). To

be processed by planar networks, the sphere must first be mapped onto a 2D pixel grid (SILVEIRA et al., 2022). The equirectangular projection described next is the canonical mapping and the input format used by HSDC, SWHDC, and this work.

2.2.1 Equirectangular Projection

An ERP image of resolution $H \times W$ samples the unit sphere S^2 on a regular grid in spherical coordinates: the azimuth $\theta \in [-\pi, \pi)$ and the elevation $\varphi \in [-\pi/2, \pi/2]$. The pixel-to-angle map is linear,

$$\theta(u) \;=\; \frac{u}{W}\,2\pi - \pi, \qquad \varphi(v) \;=\; \frac{\pi}{2} - \frac{v}{H}\,\pi, \tag{2.4}$$

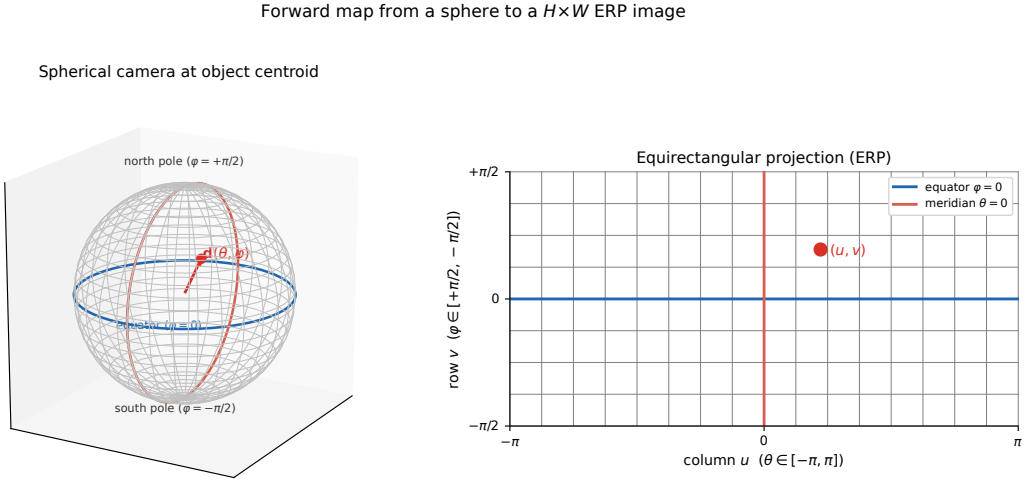
where $u \in \{0, \dots, W-1\}$ and $v \in \{0, \dots, H-1\}$ are the horizontal and vertical pixel indices, and the corresponding 3D unit direction is

$$\mathbf{d}(\theta, \varphi) \;=\; (\cos\varphi\cos\theta,\; \cos\varphi\sin\theta,\; \sin\varphi)^{\top}, \tag{2.5}$$

following the ERP convention adopted by the HSDC and SWHDC papers (STRINGHINI; JUNG, 2024a; STRINGHINI; JUNG, 2024b, Eq. 1–2). Both maps are well defined everywhere except at the poles, where the azimuth is undefined. Throughout this work, θ always denotes the azimuth and φ the *elevation* (latitude), so the equator lies at $\varphi = 0$ and the poles at $\varphi = \pm\pi/2$. Part of the spherical-image literature instead measures the polar angle (colatitude) $\pi/2 - \varphi$ down from the north pole; formulas quoted from those works are converted to the elevation convention here. Equations (2.4)–(2.5) together form what this text calls the *ERP camera model*: the grid acts as an omnidirectional camera at the observation point, each pixel (u, v) owning the direction $\mathbf{d}(\theta(u), \varphi(v))$. Figure 2.2 illustrates the forward map.

The ERP is not an isometric parameterisation of the sphere, and two geometric peculiarities follow. First, a small horizontal step in the image covers a smaller arc on the sphere near the poles than near the equator: the parallel at elevation φ has circumference proportional to $\cos\varphi$, yet every ERP row spans the same W pixels, so the horizontal pixel density per unit arc length grows as $1/\cos\varphi$. A point near the pole therefore occupies many more pixels than an equivalent point near the equator, and a fixed-size convolution kernel covers a smaller geodesic area at the poles than at the equator (Figure 2.3).

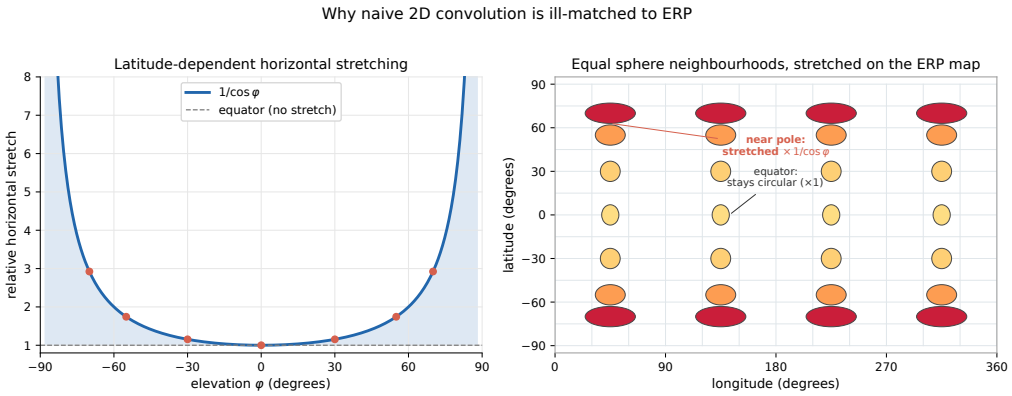
Figure 2.2: Forward map from a sphere to an $H \times W$ ERP image. A ray $\mathbf{d}(\theta, \varphi)$ from the spherical camera at the centroid (left) maps to pixel (u, v) via Equations (2.4)–(2.5) (right). The equator ($\varphi = 0$) maps to the horizontal midline and the prime meridian ($\theta = 0$) to the vertical midline.



Source: the author.

Second, the ERP is periodic on its horizontal axis: the left and right edges of the image are identified along the prime meridian, so a horizontal shift by W pixels is the identity. A standard 2D convolution with zero padding ignores this periodicity and produces boundary artefacts.

Figure 2.3: Latitude distortion: a fixed-size receptive field covers a smaller geodesic region near the poles than near the equator, because the horizontal pixel density grows as $1/\cos\varphi$ (STRINGHINI; JUNG, 2024a).



Source: the author.

2.2.2 CNNs for ERPs

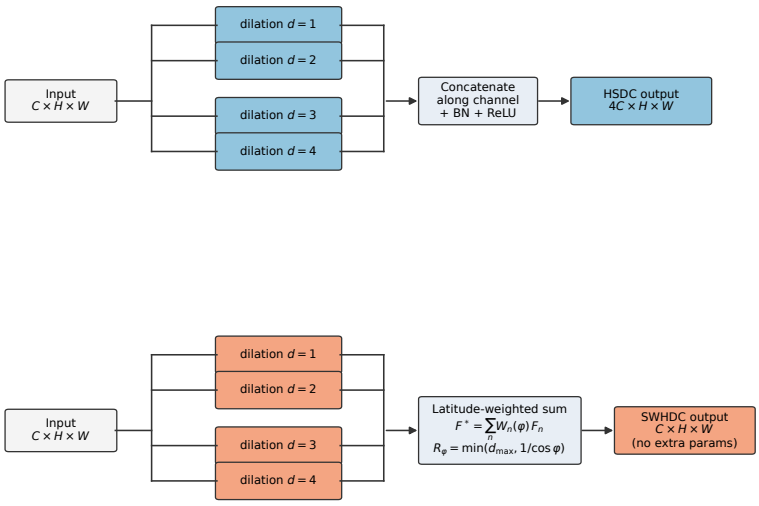
The two peculiarities above make a direct application of a standard CNN to an ERP suboptimal: a fixed 3×3 kernel sees a different geodesic neighbourhood at each image row, and zero padding splits features that cross the prime meridian between the left and right edges of the image. One family of solutions redefines the convolution operator on the sphere, through rotation-group spherical correlations (COHEN et al., 2018), distortion-aware kernel sampling (COORS; CONDURACHE; GEIGER, 2018), or learned row-dependent kernels (SU; GRAUMAN, 2017); these operators gain spherical awareness at a significant computational cost. A second family keeps the standard 2D convolution and compensates the distortion architecturally, through multi-scale horizontal receptive fields. The HSDC (STRINGHINI; JUNG, 2024a) and SWHDC (STRINGHINI; JUNG, 2024b) blocks adopted by this work belong to this second family.

Both blocks are inserted into residual networks (HE et al., 2016), the standard CNN backbone family in which each layer learns a residual function on top of an identity shortcut. ResNet-34 stacks *basic* blocks (two 3×3 convolutions), while ResNet-50 stacks *bottleneck* blocks (1×1, 3×3, 1×1 convolutions); this work uses one backbone per variant, matching the original papers. Both blocks are also built on the *dilated convolution* of Yu and Koltun (2016): a convolution whose kernel taps are spaced by a dilation rate $d > 1$, covering a larger receptive field with the same number of learnable parameters.

Both blocks share the same core idea: apply four convolutions to the same feature map, each with a different *horizontal* dilation rate $d \in \{1, 2, 3, 4\}$ and a vertical dilation of one, so the four branches cover horizontal receptive fields of increasing width. The kernel weights are shared across the branches and the horizontal axis uses *circular* padding to respect the ERP periodicity. Because the dilation-d branch covers an arc that grows towards the poles as $d/\cos\varphi$, the four branches give the network a choice of receptive field that can match the local geodesic scale at each latitude. Sharing the kernel means the four branches together cost no more parameters than a single convolution. In implementation terms, one single 3×3 weight tensor is allocated and passed to four invocations of the standard dilated convolution operator. Branch d places the nine kernel taps at horizontal offsets $\{-d, 0, +d\}$ and vertical offsets $\{-1, 0, +1\}$ around each output pixel, so every branch computes a weighted sum over a *different set* of input pixels using the *same* nine weights. During backpropagation, the gradients of the four branches accumulate into the one shared tensor.

The two blocks differ only in how they combine the branches (Figure 2.4). HSDC (STRINGHINI; JUNG, 2024a) concatenates them along the channel axis, so a block with C_{in} inputs produces $4\,C_{\text{in}}$ output channels and a downstream 1×1 convolution restores the count. SWHDC (STRINGHINI; JUNG, 2024b) instead takes a weighted sum of the four branches, with different weights at each image row. The weights come from the distortion factor $R_\varphi = \min(d_{\max},\, 1/\cos\varphi)$, where $d_{\max} = 4$ is the largest dilation rate (the symbol N is reserved in this text for the number of Gaussian primitives): the closer a row is to a pole, the more weight goes to the branches with higher dilation. These weights are fixed rather than learned, so SWHDC keeps the channel count at C_{in} and adds zero parameters, which makes it a drop-in replacement for the 3×3 convolution inside a ResNet-50 bottleneck.

Figure 2.4: The two distortion-correction blocks. Both apply four shared-weight 3×3 convolutions with horizontal dilations $d \in \{1, 2, 3, 4\}$ and circular padding. *Top (HSDC):* concatenation, giving $4\,C_{\text{in}}$ output channels. *Bottom (SWHDC):* latitude-dependent weighted sum (weights from $R_\varphi = \min(d_{\max}, 1/\cos\varphi)$), keeping C_{in} channels with zero added parameters.



Source: the author, after Stringhini and Jung (2024a), Stringhini and Jung (2024b).

3 RELATED WORK

Methods for 3D object classification are commonly grouped by the input type consumed by the network: point-based, voxel-based, and view-based strategies (STRINGHINI, 2025). This chapter follows that taxonomy. Sections 3.1 to 3.3 briefly survey each group, with emphasis on the view-based line that this work extends. Section 3.4 reviews the emerging family of classifiers that operate on 3DGS data and closes the chapter by positioning this work against Gaussian-MAE (MA et al., 2025), its most directly related predecessor.

3.1 Point-Based Methods

Point-based methods extract features directly from unordered point clouds. PointNet (QI et al., 2017a) introduced permutation-invariant feature learning through shared multilayer perceptrons (MLPs) and max pooling, and PointNet++ (QI et al., 2017b) extended it with hierarchical local aggregation. Later architectures improved the same paradigm: PointMLP (MA et al., 2022) with a pure residual-MLP design, and PointNeXt (QIAN et al., 2022) with revisited training and scaling strategies for PointNet++. Attention-based variants tokenise the point cloud and pretrain a Transformer encoder with masked-prediction objectives on ShapeNet, as in Point-BERT (YU et al., 2022) and Point-MAE (PANG et al., 2022). On ModelNet40 these methods range from 89.2% (PointNet) to about 94% (PointMLP) overall accuracy. When the source data are meshes, as in ModelNet, point-based methods depend on a sampling strategy to convert the surface into points, which can affect the classification result (STRINGHINI, 2025).

3.2 Voxel-Based Methods

Voxel-based methods discretise the object into a regular 3D grid and apply volumetric networks. 3DShapeNets (WU et al., 2015), which also introduced the ModelNet benchmark, models voxel occupancy with a convolutional deep belief network, and VoxNet (MATURANA; SCHERER, 2015) applies 3D convolutions to occupancy grids. Voxel approaches capture volumetric structure but scale cubically in memory with grid resolution, and meshes, being 2D manifolds embedded in 3D space, produce mostly

empty voxels (STRINGHINI, 2025). They have largely been superseded by point- and view-based methods on ModelNet.

3.3 View-Based Methods

View-based methods project the 3D object onto one or more 2D images and reuse established planar CNNs, typically at lower computational cost than point- and voxel-based networks (STRINGHINI, 2025). *Multi-view* methods render the object from several viewpoints and aggregate the per-view features: MVCNN (SU et al., 2015) established this paradigm, and RotationNet (KANEZAKI; MATSUSHITA; NISHIDA, 2018) extended it with joint pose estimation over unsupervised viewpoints. Their main cost is the need to render, process, and aggregate many views per object.

Single-view methods summarise the object in one image. The use of panoramic projections for this purpose predates the blocks adopted in this work by nearly a decade: DeepPano (SHI et al., 2015) classifies a depth panorama obtained by cylindrical projection of the shape; Sfikas, Theoharis and Pratikakis (2017), Sfikas, Pratikakis and Theoharis (2018) project depth and surface-orientation features onto the lateral surface of a cylinder and classify the resulting PANORAMA representations, named after the panoramic shape descriptor of Papadakis et al. (2010), with an ensemble of CNNs; and Cao, Huang and Karthik (2017) introduced spherical projections centred at the object's barycenter, recording ray distances on the sphere. These early methods established panoramas and spherical images as a competitive input for 3D shape classification, but they either miss the polar regions (cylindrical projections) or leave the latitude distortion of the spherical projection uncorrected.

The HSDC (STRINGHINI; JUNG, 2024a) and SWHDC (STRINGHINI; JUNG, 2024b) works extend this single-view line by combining a multi-channel ERP of the mesh, ray-cast from the area-weighted centroid (depth, normal, curvature features at the first ray–mesh intersection), with the distortion-correcting convolutions described in Section 2.2.2. ResNet-34 with HSDC blocks attains 97.1% overall accuracy on ModelNet10 and 93.9% on ModelNet40 with only 5.3 million parameters; ResNet-50 with SWHDC blocks attains 94.11% and 91.9% while adding zero parameters over the plain backbone. The SWHDC paper also evaluated a from-scratch Transformer baseline on the same input, the encoder of PanoFormer (SHEN et al., 2022) followed by a fully-connected classification head, which reached only 85.7% / 79.7%, suggesting that without large-scale

pretraining the inductive biases of a distortion-aware CNN match the ERP modality better than a ViT does. Table 3.1 summarises these baselines, which are the reference targets for the RF-ERP variants reported in Chapter 5.

Table 3.1: ERP-based 3D object classification baselines, reported on geometric (mesh ray-cast) ERP. Accuracy figures are taken directly from the cited works.

Method	Backbone	MN10 OA	MN40 OA	Params (M)
HSDC (STRINGHINI; JUNG, 2024a)	ResNet-34 + HSDC	97.1%	93.9%	5.3
SWHDC (STRINGHINI; JUNG, 2024b)	ResNet-50 + SWHDC	94.1%	91.9%	25.5
PanoFormer encoder + FC (from scratch) (STRINGHINI; JUNG, 2024b)	ViT	85.7%	79.7%	—

Source: the author, compiled from Stringhini and Jung (2024a), Stringhini and Jung (2024b).

3.4 3DGS-Based Classification

The release of 3DGS (KERBL et al., 2023), together with large-scale 3DGS datasets such as ShapeSplat (MA et al., 2025), has motivated a small but fast-growing literature on classifying objects represented as Gaussian splats. Most of these works, however, use 3DGS only indirectly. GS-PT (LUO et al., 2025) embeds a 3DGS renderer inside a self-supervised pretraining pipeline for a point-cloud Transformer; at inference the model still consumes point clouds. UniGS (LI et al., 2025) learns a language-aligned 3DGS embedding for zero-shot classification and does not evaluate on ModelNet. 3D Gaussian Point Encoders (JAMES, 2025) use fixed 3D Gaussians to speed up PointNet-style point-cloud encoders, and the BSc thesis of Berg (2024) feeds raw Gaussian parameters to PointNet++, concluding that the non-uniform centroid distribution of optimised Gaussians limits accuracy.

The only published work that operates directly on the 3DGS parameters for a downstream classification task is Gaussian-MAE, introduced by Ma et al. (2025) together with the ShapeSplat dataset, whose ModelSplat subset is the data source of this work. Gaussian-MAE treats the Gaussian primitives as an unordered set in a high-dimensional feature space and trains a masked autoencoder (PANG et al., 2022) on it, using a 12-layer Transformer encoder, a Gaussian feature-grouping step, and a splat-pooling layer. The authors ablate the input features: with centroids only, the model is slightly worse than the point-cloud baseline (93.72% vs. 94.93% for Point-MAE on ModelNet10), which they attribute to the non-uniform centroid distribution produced by 3DGS optimisation; with all Gaussian attributes it surpasses the point-cloud baselines, reaching 95.37% on ModelNet10 and 93.35% on ModelNet40 (Table 3.2). These results use 1024 input splats, 300 epochs of self-supervised pretraining on a 52 K-object subset of ShapeNet, and 300

epochs of fine-tuning.

Table 3.2: Selected results from Ma et al. (2025), reported with 1024 input splats per object and full encoder fine-tuning. C, O, S, R, SH denote centroid, opacity, scale, rotation and spherical-harmonic colour coefficients respectively.

Method	**Input features**	**MN10 OA**	**MN40 OA**
Point-BERT (YU et al., 2022)	Point cloud	94.82%	93.20%
Point-MAE (PANG et al., 2022)	Point cloud	94.93%	93.20%
Gaussian-MAE (MA et al., 2025)	$E(C)$	93.72%	91.77%
Gaussian-MAE	$E(C, S, R)$	94.27%	93.19%
Gaussian-MAE	$E(O, C, S, R)$	95.48%	92.42%
Gaussian-MAE	$E(\mathrm{All})$	$\mathbf{95.37}\%$	$\mathbf{93.35}\%$

Source: Ma et al. (2025), Tables 2–3.

3.5 Positioning of This Work

Table 3.3 organises the surveyed methods by the input modality consumed by the classifier, the architectural family, and the pretraining regime. Three gaps stand out. First, no published work converts a 3DGS representation into a structured 2D image and classifies it with a CNN trained from scratch. Second, from-scratch training is rare in the 3DGS literature: Gaussian-MAE and GS-PT both depend on large-scale pretraining, which confounds the contribution of the input modality with that of the pretraining data. Third, the ERP-based CNNs reach their accuracies with 5–25 million parameters (Table 3.1), whereas the Transformer-based methods need about 22 million in the encoder alone (YU et al., 2022; PANG et al., 2022; MA et al., 2025), plus a pretraining budget; the parameter counts of all families are compared side by side in Chapter 5.

This work therefore sits immediately after Gaussian-MAE in the 3DGS-classification literature: both consume the same ModelSplat objects, but where Gaussian-MAE processes the Gaussian parameters as an unordered set with a pretrained Transformer, this work converts the splats into an RF-ERP image and classifies it with a distortion-correcting CNN trained from scratch. The conversion keeps the parameter efficiency of the view-based family and opens that family to the 3DGS modality. Chapter 5 quantifies the resulting accuracy–parameter trade-off.

Table 3.3: Taxonomy of 3D object classification methods on ModelNet by classifier input modality, architectural family and pretraining regime. "3DGS-RF-ERP" denotes the radiance-field equirectangular projection of this work; "SSL" stands for self-supervised learning (pretraining without labels).

Method	**Input modality**	**Architecture**	**Pretraining**
PointNet (QI et al., 2017a)	Point cloud	MLP	None
PointNet++ (QI et al., 2017b)	Point cloud	Hierarchical MLP	None
PointMLP (MA et al., 2022)	Point cloud	Residual MLP	None
PointNeXt (QIAN et al., 2022)	Point cloud	Improved PointNet++	None
Point-BERT (YU et al., 2022)	Point cloud	Transformer	SSL on ShapeNet
Point-MAE (PANG et al., 2022)	Point cloud	Transformer	SSL on ShapeNet
Gaussian-MAE (MA et al., 2025)	3DGS parameters	Transformer	SSL on ShapeNet
GS-PT (LUO et al., 2025)	Point cloud (3DGS-augmented)	Transformer	SSL on ShapeNet
UniGS (LI et al., 2025)	3DGS parameters	Transformer	Language-Image-3D (CLIP)
van den Berg (BERG, 2024)	3DGS parameters	PointNet++	None
DeepPano (SHI et al., 2015)	Cylindrical panorama	Shallow CNN	None
HSDC (STRINGHINI; JUNG, 2024a)	Geometric ERP	ResNet-34 + HSDC	None
SWHDC (STRINGHINI; JUNG, 2024b)	Geometric ERP	ResNet-50 + SWHDC	None
This work	**3DGS-RF-ERP**	**ResNet + HSDC/SWHDC**	**None**

Source: the author.

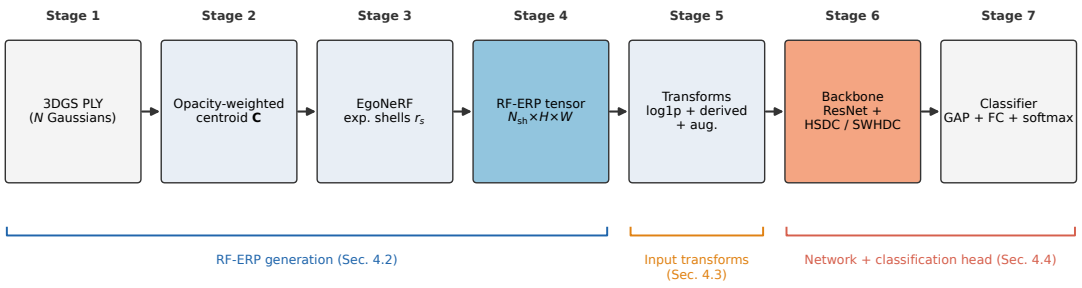
4 METHODOLOGY

This chapter describes the full pipeline, from a pre-trained 3D Gaussian Splat (3DGS) file to a class prediction produced by a distortion-correcting CNN. Section 4.1 gives an overview and fixes the notation. Section 4.2 derives the radiance-field equirectangular projection (RF-ERP), the central technical contribution of this work. Section 4.3 covers the transforms applied to the RF-ERP before training, Section 4.4 describes the two backbones, and Sections 4.5–4.6 detail the training and evaluation protocols.

4.1 Pipeline Overview

The proposed pipeline transfers the single-view ERP representation of 3D shapes developed by Stringhini (2025) to the 3DGS modality. In the original, geometric pipeline (STRINGHINI; JUNG, 2024a; STRINGHINI; JUNG, 2024b; STRINGHINI, 2025), rays are cast omnidirectionally from the centroid of a triangle mesh, and each ERP pixel stores geometric features (depth, normal, curvature) computed at the ray–mesh intersection points. This work replaces the mesh by the continuous RF encoded by a 3DGS representation (KERBL et al., 2023) and keeps the rest of the classification pipeline as close as possible to the original, so that the effect of the input modality can be isolated. Figure 4.1 shows the seven stages. Stages 1–4 produce the RF-ERP from the input PLY file (Section 4.2); stages 5–7 are the input-side transforms (Section 4.3), the convolutional backbone (Section 4.4), and the classification head.

Figure 4.1: Pipeline overview, from 3DGS PLY input through centroid, EgoNeRF concentric shells, N_{shells}-channel RF-ERP, input transforms, HSDC/SWHDC backbone, and classification head. Stages 1–4 correspond to the RF-ERP generation of Section 4.2, stage 5 to the input transforms of Section 4.3, and stages 6–7 to the networks and classification head of Section 4.4.



Source: the author.

The following notation is used throughout the chapter. N is the number of 3D Gaussian primitives in the input object; $\boldsymbol{\mu}_i, \alpha_i, \mathbf{s}_i, \mathbf{q}_i$ are the centre, opacity, per-axis scale, and quaternion rotation of the i-th Gaussian, decoded from the raw PLY values as described in Section 2.1; $\boldsymbol{\Sigma}_i$ is the corresponding 3×3 covariance matrix; $\mathbf{C}$ is the opacity-weighted centroid of the object; N_{shells} is the number of concentric sampling shells and r_s the radius of the s-th shell; H and W are the height and width of the ERP in pixels; (θ, φ) are the azimuth and elevation on the unit sphere; and $\rho(\mathbf{p})$ is the volumetric density of the RF at the point $\mathbf{p} \in \mathbb{R}^3$.

4.2 Radiance-Field ERP Generation

The core idea of this work is to apply the ERP projection of Stringhini (2025) to a *continuous RF* rather than to a discrete geometric mesh. This changes the nature of the per-pixel measurement. A ray cast through a mesh produces a discrete set of intersection points, at which surface features are well defined. An RF, in contrast, is a continuous volumetric density $\rho(\mathbf{p})$ (Equation (2.3)): there are *no intersection points* for the rays to report, only a density value at every point along the ray. The per-pixel feature must therefore be built by *sampling* the field along the ray instead of intersecting a surface.

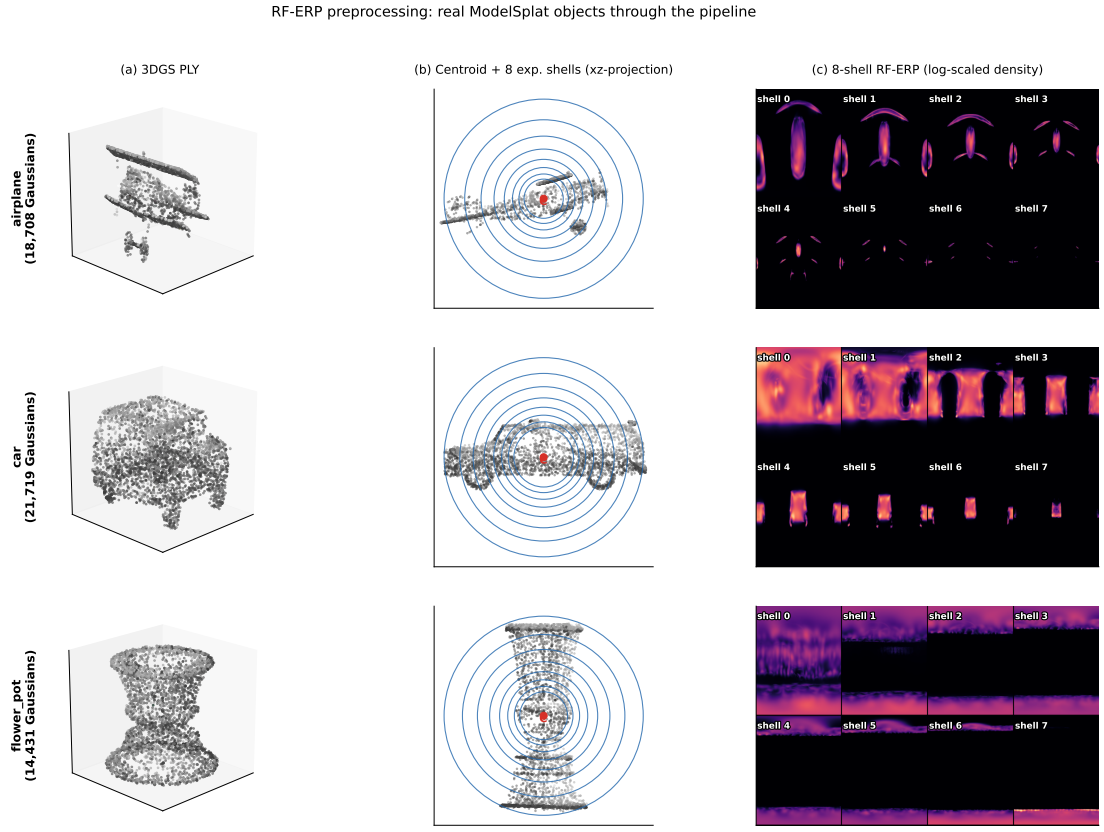
To structure this sampling, the pipeline accumulates the density on N_{shells} concentric spherical shells centred at the object, with exponentially spaced radii inspired by the balanced spherical grid of EgoNeRF (CHOI; KIM; KIM, 2023). EgoNeRF showed that, for egocentric RFs, an exponential radial schedule concentrates samples in the near field, where the radiance content is densest, and stretches the spacing in the far field, where the field is nearly empty; this work borrows only that sampling concept, not the EgoNeRF rendering machinery. Each shell becomes one channel of the output ERP, so the resulting $N_{\text{shells}} \times H \times W$ tensor encodes a coarse radial decomposition of the object's density. The generation proceeds in five steps:

1. compute the opacity-weighted centroid $\mathbf{C}$ of the object (Section 4.2.1);
2. build the per-pixel unit ray directions $\mathbf{d}(u, v)$ of the spherical camera centred at $\mathbf{C}$ (Section 4.2.2);
3. select N_{shells} concentric sampling radii $\{r_s\}$ adaptively from the distribution of Gaussian centres (Section 4.2.3);
4. integrate the RF density radially within each shell (Section 4.2.4); and

5. evaluate the 3DGS density at each sample point with spatial culling for efficiency (Section 4.2.5).

Figure 4.2 traces three ModelSplat objects through these steps, from the raw Gaussian cloud, through the centroid and shell placement, to the final multi-channel ERP tensor, and makes the intermediate products of the pipeline concrete before each step is detailed below.

Figure 4.2: Three ModelSplat objects (`airplane`, `car`, `flower_pot`) traced through the RF-ERP pipeline. (a) Raw 3DGS PLY as a coloured point cloud. (b) Opacity-weighted centroid $\mathbf{C}$ and the eight EgoNeRF sampling shells on the xz-projection. (c) The resulting $8\times256\times512$ RF-ERP, with the eight channels laid out as a 2×4 grid: shell 1 (innermost) at the top left through shell 8 (outermost) at the bottom right. Each panel is one channel of the tensor; brighter pixels mean higher log-compressed density integrated over that shell's radial interval, so the object's surface lights up in the shells whose interval crosses it and fades in the others, encoding a coarse near-to-far radial decomposition of the object.



Source: the author.

4.2.1 Opacity-Weighted Centroid

Every ERP camera model needs an optical centre. This work uses the *opacity-weighted centroid* of the Gaussian cloud,

$$\mathbf{C} = \frac{\sum_{i=1}^{N} \alpha_i \, \boldsymbol{\mu}_i}{\sum_{i=1}^{N} \alpha_i}, \tag{4.1}$$

which is the direct analogue of the area-weighted mesh centroid used by Stringhini and Jung (2024a, §II-A). The opacity α_i plays the role of a perceptual mass: low-opacity Gaussians contribute little to the rendered RF. Because the sigmoid-decoded opacities are strictly positive and a ModelSplat object carries 10^4–10^5 Gaussians (Section 5.1.1), the denominator $\sum_i \alpha_i$ is in practice far from zero; the implementation keeps a defensive fallback to the arithmetic mean of $\boldsymbol{\mu}_i$ for degenerate inputs (an empty or fully transparent file), but this guard is never triggered on ModelSplat. For a typical ModelSplat object, $\mathbf{C}$ lies inside the convex hull of the surface Gaussians, so a single spherical panorama can observe the entire object.

4.2.2 ERP Ray Directions

The output ERP has resolution $H \times W$ (default 256×512). The spherical coordinates $(\theta(u), \varphi(v))$ of each pixel and the corresponding unit ray direction $\mathbf{d}(u, v)$ follow the ERP camera model of Equations (2.4)–(2.5): the top row of the image is the north pole ($\varphi = +\pi/2$), the bottom row is the south pole, and the centre column corresponds to $\theta = 0$. The set $\{\mathbf{d}(u, v)\}$ is precomputed once per resolution and reused for every shell.

4.2.3 Shell Radii via EgoNeRF Spacing

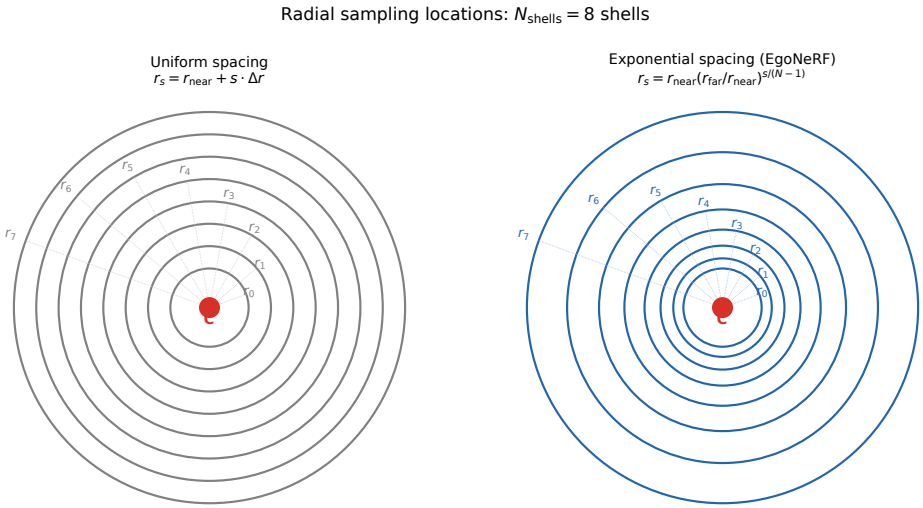
Following the balanced spherical grid of Choi, Kim and Kim (2023, §3.2), the density is sampled on N_{shells} concentric spheres whose radii grow exponentially from an

inner radius r_{near} to an outer radius r_{far},

$$r_s \;=\; r_{\text{near}} \cdot \left(\frac{r_{\text{far}}}{r_{\text{near}}} \right)^{s/(N_{\text{shells}}-1)}, \qquad s = 0, 1, \ldots, N_{\text{shells}} - 1. \tag{4.2}$$

This packs the shells densely in the near field, where the object surface usually lies, and spreads them out in the far field (Figure 4.3). The two bounds are set per object from the Euclidean distances $\|\boldsymbol{\mu}_i - \mathbf{C}\|_2$ between the Gaussian centres and the centroid $\mathbf{C}$: r_{near} is the 5th percentile and r_{far} the 95th. Percentiles rather than the minimum and maximum guard against *floater* Gaussians, low-opacity primitives that RF optimisation leaves drifting far from the surface (PHILIP; DESCHAINTRE, 2023) and that would otherwise inflate r_{far}. If the two bounds coincide, the grid falls back to linear spacing around the single radius.

Figure 4.3: Exponential shell spacing of Choi, Kim and Kim (2023) used by the RF-ERP pipeline. The $N_{\text{shells}} = 8$ shells are centred on the opacity-weighted centroid $\mathbf{C}$; the schedule concentrates shells in the dense near field and stretches the spacing in the far field.



Source: the author, after Choi, Kim and Kim (2023).

4.2.4 Per-Shell Radial Integration

A pure point sample of the density at r_s would miss the density between adjacent shells. Each shell s is therefore associated with a radial interval $[r_s^-, r_s^+]$ bounded by the midpoints between consecutive shell centres, $r_s^{\pm} = \frac{1}{2}(r_s + r_{s\pm1})$, with the first and last endpoints extended symmetrically so that the shells cover the full radial range. Within

$[r_s^-, r_s^+]$, the pipeline ray-marches at K uniformly spaced radii and writes the arithmetic mean of the sampled densities into channel s of the ERP. With $K = 1$ this is a point sample at the shell centre; with larger K it approximates a Riemann sum over the radial extent of the shell, so that each ERP channel represents an integral of the RF density over its interval rather than a single slice. Every experiment in this work uses $K = 4$, the default of the released configuration files (Table 5.3).

4.2.5 3DGS Density Evaluation and Spatial Culling

At every sample point along every ray, the pipeline evaluates the volumetric density of the RF: the opacity-weighted sum of all Gaussian kernels defined in Equation (2.3) (Section 2.1), following Kerbl et al. (2023). The orientation of each primitive is recovered from its stored quaternion (SHOEMAKE, 1985), and the per-Gaussian geometry is precomputed once per object, so that evaluating one whole shell becomes a single batched vectorised operation over all surviving Gaussians and all ERP pixels. Evaluating every Gaussian at every sample point would still be prohibitively expensive (on the order of 10^5 Gaussians times 10^7 sample points per object). But a Gaussian kernel contributes nothing of practical relevance beyond its 3σ support (Figure 2.1, right), the same cutoff used by the original 3DGS rasteriser. The σ here refers to the spread of the *Gaussian*, not to the thickness of the shell. For the anisotropic kernel of Equation (2.2), whose covariance is $\mathbf{\Sigma}_i = \mathbf{R}_i \,\mathrm{diag}(\mathbf{s}_i)^2\, \mathbf{R}_i^\top$, the standard deviation along principal axis k is the per-axis scale $s_{i,k}$. The 3σ support is therefore the ellipsoid of Mahalanobis distance $D \leq 3$, which fits inside a ball of radius $3\, s_i^{\max}$ around $\boldsymbol{\mu}_i$, where $s_i^{\max} = \max_k s_{i,k}$ is the largest of the three scales. The shell, in turn, has a radial thickness of its own: all its sample points lie at Euclidean distances within $[r_s^-, r_s^+]$ from the centroid (Section 4.2.4). Putting the two together, a Gaussian can influence shell s only if the distance $\|\boldsymbol{\mu}_i - \mathbf{C}\|_2$ from its centre to $\mathbf{C}$ falls inside that interval padded by $3\, s_i^{\max}$ on both sides. The pipeline checks this condition first and discards every Gaussian that fails it, before any density is evaluated. This *spatial culling* eliminates more than 95% of the Gaussians per shell and yields an approximate $20\times$ speed-up with no perceptible change in the output density.

The density evaluation uses only the geometric parameters of each Gaussian, its centre, covariance, and opacity; the spherical-harmonic colour coefficients stored in the same PLY file are deliberately discarded, for three reasons. First, the focus of this work is geometry: object category is above all a property of *shape*, and the geometric-

ERP baselines likewise consume purely geometric features (depth, normals, curvature) (STRINGHINI; JUNG, 2024a; STRINGHINI; JUNG, 2024b), which keeps the input-modality comparison like-for-like. Second, the feature ablation of Ma et al. (2025) shows their geometry-plus-opacity variant $E(O, C, S, R)$ on par with the full-attribute variant that includes the harmonics (Table 3.2), so most of the discriminative signal lives in the geometric parameters. Third, the choice keeps the channel count independent of the SH degree used during 3DGS optimisation. Its consequences are revisited in Section 6.2.

After all shells have been integrated, the pipeline returns a non-negative `float32` tensor with one channel per shell, $8 \times 256 \times 512$ in the default configuration, whose entry at (s, v, u) is the mean density inside shell s along the ray of pixel (u, v). Note that, unlike the geometric ERP of Stringhini and Jung (2024a), whose pixels store explicit surface features (depth, normal, curvature), no per-pixel geometric feature is saved at this stage: each pixel holds only scalar densities, and the radial position of the mass is encoded implicitly by the channel index. An explicit depth-like cue is reintroduced downstream as the derived pseudo-depth channel of Section 4.3.2. The values are unbounded above and heavily right-skewed, because most pixels look into empty regions of the RF; this skew motivates the log-compression transform of Section 4.3.1. Every computed ERP is cached to disk; the cache directory name encodes all preprocessing hyperparameters, so any change to the pipeline automatically invalidates the cached tensors.

4.3 Classifier Input Transforms

The cached ERP from Section 4.2 is not fed directly to the CNN. Four transforms produce the classifier-ready tensor: log compression (Section 4.3.1), the addition of derived feature channels (Section 4.3.2), per-sample augmentation (Section 4.3.3), and batch-level sample-pair augmentation (Section 4.3.4).

The offline/online split deserves to be explicit. The RF-ERP generation of Section 4.2 runs *offline*, once per object, before any training, with its output cached to disk (Section 4.2.5). The four transforms of this section run *online*, at data-loading time, whenever a cached tensor is read: log compression and the derived channels are deterministic and applied to every split, whereas the two augmentation stages are stochastic, resampled at every epoch, and applied to the training split only. Nothing that involves randomness is precomputed.

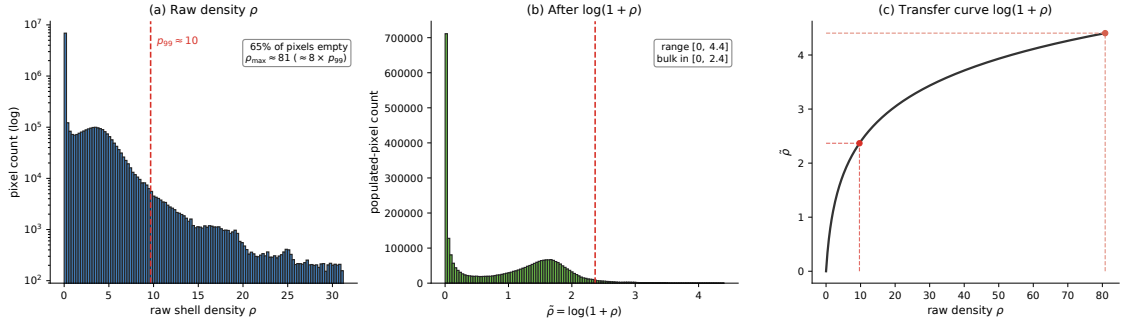
4.3.1 Log Compression

The raw shell density is heavy-tailed (Figure 4.4a): over 150 ModelNet10 objects, about 65% of the ERP pixels are empty and the maximum ($\rho_{\max} \approx 81$) is roughly $8\times$ the 99th percentile ($\rho \approx 10$). On a linear scale a few dense splats would dominate and flatten the low-density object boundaries. The pipeline therefore applies an element-wise log compression,

$$\tilde{\rho} \;=\; \log(1+\rho). \tag{4.3}$$

Its derivative $1/(1+\rho)$ is largest at $\rho = 0$, so it expands the low-density boundaries and compresses the tail (Figure 4.4b,c), mapping the values to $\tilde{\rho} \in [0, \approx 4.4]$. Densities are not standardised across objects, because the absolute magnitude is itself class-discriminative.

Figure 4.4: Empirical justification for the log compression. (a) The raw ModelNet10 shell density is heavy-tailed: a spike of empty pixels and a long tail whose maximum is about $8\times$ the 99th percentile (log-scaled counts). (b) After $\log(1+\rho)$ the populated densities occupy a bounded band. (c) The transfer curve is steepest near $\rho = 0$, amplifying low-density boundaries and compressing the tail.



Source: the author.

4.3.2 Derived Channels: Pseudo-Depth and MIP

After log compression, two derived scalar fields are concatenated as additional channels; they give the backbone explicit depth and silhouette cues that would otherwise have to be learned from the raw shells. The first, a *pseudo-depth* map, is the density-weighted average shell index, normalised to $[0, 1]$,

$$\mathrm{depth}(u,v) \;=\; \frac{1}{N_{\text{shells}}-1} \cdot \frac{\sum_{s=0}^{N_{\text{shells}}-1} s\,\tilde{\rho}_s(u,v)}{\sum_{s=0}^{N_{\text{shells}}-1} \tilde{\rho}_s(u,v) + \epsilon}, \tag{4.4}$$

where $\epsilon > 0$ is a small constant for numerical stability. It encodes which shell along the ray contains the dominant surface; it is the discrete, density-weighted analogue of the *expected depth* rendered from a volumetric RF (MILDENHALL et al., 2020), and it plays the role of the geometric depth channel of Stringhini and Jung (2024a). The second is a *maximum intensity projection* (MIP) map, $\mathrm{mip}(u, v) = \max_s \tilde{\rho}_s(u, v)$, the projection operator long used in volume rendering and medical imaging (WALLIS et al., 1989). It behaves as a soft silhouette mask: pixels whose every shell is empty are zero, while pixels that intersect any part of the object are positive. With the default configuration ($N_{\text{shells}} = 8$ density channels plus pseudo-depth and MIP), the final classifier input has shape

$$\mathrm{ERP}^{\,\mathrm{clf}} \in \mathbb{R}_{\geq 0}^{10 \times 256 \times 512}. \tag{4.5}$$

4.3.3 Single-Sample Augmentation

Every model in this work is trained from scratch, with no ImageNet or any other pretraining. This is a constraint adopted for a fair comparison with the geometric-ERP baselines, which are also trained from scratch. With no pretrained features to fall back on, the pipeline relies more on data augmentation for regularisation than a fine-tuned model would. All augmentations operate on the ERP image itself, the (C, H, W) float32 tensor produced after projection; the Gaussian cloud is never touched again. This holds even for the 3D rotation, which is semantically a rotation of the object but is realised entirely on the ERP grid, as detailed at the end of this section. The primitives are independent of the channel count C and are applied to the training split only. Table 4.1 summarises the augmentation primitives, their probabilities, and their parameter ranges. The values match the recipes of Stringhini and Jung (2024a, §III-A) and Stringhini and Jung (2024b, §IV-A), except that the per-primitive probability is raised from 0.15 to 0.30 to compensate for the smaller effective dataset size of from-scratch training on ModelSplat, and random erasing (ZHONG et al., 2020), which replaces a random rectangular patch of the input by zero, is added because the RF-ERP often contains large low-information regions that the network could otherwise memorise.

The 3D rotation is not a naive 2D image rotation, which would distort the spherical geometry near the poles: it is implemented as a rigid rotation on the sphere, by inversely mapping each output pixel through the ERP camera model and resampling the input with circular wrapping.

Table 4.1: Single-sample augmentation primitives applied during training. Each primitive fires independently with the indicated probability.

Primitive	Probability	Parameters
Horizontal flip	0.50	azimuthal 180° rotation (ERP 2π-periodic)
3D rotation	0.30	$\theta_x, \theta_y \sim \mathcal{U}[0^\circ, 15^\circ]$, $\theta_z \sim \mathcal{U}[0^\circ, 45^\circ]$
Gaussian blur	0.30	$\sigma \sim \mathcal{U}[0.1, 2.0]$, applied channel-wise
Gaussian noise	0.30	$\mu \sim \mathcal{U}[0, 10^{-3}]$, $\sigma \sim \mathcal{U}[0, 0.03]$, additive
Random erasing (ZHONG et al., 2020)	0.30	area $\sim \mathcal{U}[2\%, 33\%]$, aspect $\sim \log \mathcal{U}[0.3, 3.3]$

Source: the author.

4.3.4 Sample-Pair Augmentation

In the training loop, pairs of samples are further combined using *MixUp* (ZHANG et al., 2018) and an ERP-adapted variant of *CutMix* (YUN et al., 2019), with the two primitives alternating $50/50$ per batch. MixUp blends two samples and their labels linearly, with a mixing coefficient drawn from a $\mathrm{Beta}(0.4, 0.4)$ distribution; CutMix replaces a rectangular crop of one sample with the corresponding crop from another and adjusts the labels by the fraction of pixels retained. The CutMix implementation wraps the crop along the horizontal axis, exploiting the 2π-periodicity of the ERP, so that the cropped region is a geometrically valid sub-panorama rather than an arbitrary 2D rectangle.

After all transforms, the per-sample tensor delivered to the loss is

$$\mathbf{x} \in \mathbb{R}^{C \times H \times W}, \qquad C = N_{\text{shells}} + n_{\text{derived}} = 10, \qquad H = 256,\ W = 512, \tag{4.6}$$

with corresponding label vector $\mathbf{y} \in \Delta^{K-1}$ (the K-simplex) and $K = 10$ for ModelNet10 or $K = 40$ for ModelNet40. This is the contract that every network in Section 4.4 consumes.

4.4 Network Architectures

Two distortion-aware backbones are described here, each pairing a residual network (HE et al., 2016) with one of the two distortion-correction blocks of Section 2.2.2: ResNet-34 + HSDC and ResNet-50 + SWHDC. Each is also evaluated as a plain (planar) ResNet of the same depth, with the block replaced by standard convolutions, so the experiments cover four configurations in total: two distortion-aware and two planar. All four accept the $10 \times 256 \times 512$ tensor of Equation (4.6) and produce a class-probability distribution over the K ModelNet categories. The planar baselines are the controlled ablation

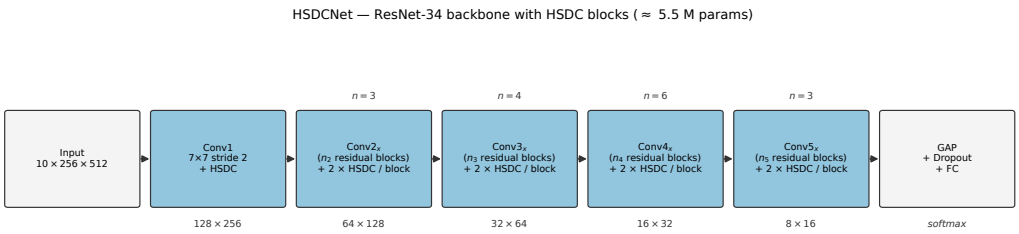
of the blocks; the two subsections below detail the distortion-aware variants.

Each block is paired with the backbone used in its original paper; the complementary combinations (HSDC on ResNet-50, SWHDC on ResNet-34) are not evaluated. First, published geometric-ERP accuracies exist only for ResNet-34 + HSDC (STRINGHINI; JUNG, 2024a) and ResNet-50 + SWHDC (STRINGHINI; JUNG, 2024b), so only these pairings allow the input-modality comparison of Section 5.4.1 with the architecture held fixed. Second, each block was designed around its host: HSDC quadruples the channel count of every convolution it replaces, affordable in the narrow basic blocks of ResNet-34 but not in the sixteen already-wide bottlenecks of a ResNet-50, whereas SWHDC is a parameter-free drop-in for exactly the 3×3 convolution inside such a bottleneck. The cross-pairings, with no published reference to compare against, are left as future work (Section 6.3).

4.4.1 ResNet-34 + HSDC

The first backbone follows the architecture of Stringhini and Jung (2024a) and replaces every standard convolution in ResNet-34 with an HSDC block (Section 2.2.2). The integration follows Stringhini and Jung (2024a, Table 1): the initial 7×7 stem receives one HSDC block, and each of the four residual stages (Conv2–Conv5) receives two HSDC blocks. The first convolution accepts $C = 10$ input channels instead of three, because the input is the RF-ERP of Equation (4.5); this is the only parameter-count difference from the original network. The total number of trainable parameters is about $5.5\,\mathrm{M}$, of which only $0.2\,\mathrm{M}$ come from the wider stem. Figure 4.5 sketches the resulting network, stage by stage, from the 10-channel input to the classification head.

Figure 4.5: HSDCNet (ResNet-34 + HSDC) for 10-channel RF-ERP input: one HSDC block in the stem, two HSDC blocks per residual stage, GAP+FC head; ~ 5.5 M parameters.

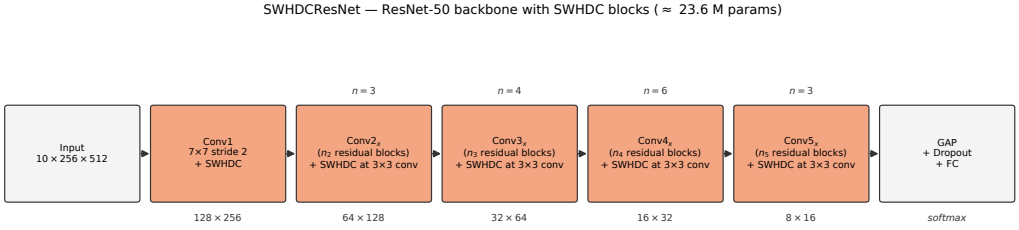


Source: the author, after Stringhini and Jung (2024a).

4.4.2 ResNet-50 + SWHDC

The second backbone follows the architecture of Stringhini and Jung (2024b) and replaces every 3×3 convolution in the bottleneck blocks of ResNet-50 with an SWHDC block (Section 2.2.2). Because the four dilated kernels are shared and the latitude-dependent combination is parameter-free, each substitution keeps the parameter count of the original layer. The total number of trainable parameters of ResNet-50 + SWHDC on the RF-ERP input is about $23.6\,\mathrm{M}$. Figure 4.6 sketches the resulting network; the macro-structure mirrors Figure 4.5, with the substitution happening inside each bottleneck rather than at every convolution.

Figure 4.6: SWHDCResNet (ResNet-50 + SWHDC) for 10-channel RF-ERP input: every 3×3 convolution inside each bottleneck is replaced by an SWHDC block; head matches HSDCNet; ~ 23.6 M parameters.



Source: the author, after Stringhini and Jung (2024b).

4.4.3 Classification Head

Both backbones share the same classification head. The $C_{\text{last}} \times H' \times W'$ feature map from the final residual stage is reduced to a C_{last}-vector by Global Average Pooling, passed through a dropout layer with probability 0.3, and projected to the K-class logits by a single fully-connected layer. At inference, a softmax over the logits gives the class probabilities; at training, the logits are fed directly to the cross-entropy loss (Section 4.5).

4.5 Training Protocol

All models are trained *from scratch*, without any pretraining on auxiliary datasets. This is a deliberate choice: the goal of this work is to test whether the distortion-correction

principle of HSDC and SWHDC generalises from geometric mesh ERP to RF-ERP, and pretraining on natural images would confound that comparison.

The AdamW optimiser (LOSHCHILOV; HUTTER, 2019) is used with momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, decoupled weight decay 5×10^{-4}, and a base learning rate of 10^{-4}, warmed up linearly over the first 10 epochs and then annealed by a cosine schedule down to a floor of 10^{-6}. The objective is the class-weighted cross-entropy with label smoothing (SZEGEDY et al., 2016) of strength 0.1: smoothing discourages arbitrarily confident logits, and class weights inversely proportional to the per-class sample counts mitigate the moderate class imbalance of ModelNet. When MixUp or CutMix is active for a batch, the loss is applied to the correspondingly mixed labels (Section 4.3.4).

Gradients are clipped to a maximum global ℓ_2 norm of 1.0 to guard against the loss spikes that occasionally occur with strong MixUp/CutMix mixing, and mixed-precision training is enabled on CUDA devices, providing roughly a $2\times$ throughput improvement without measurable degradation in final accuracy. The maximum number of epochs is 500 for HSDC and 400 for SWHDC, following the recipes of Stringhini and Jung (2024a) and Stringhini and Jung (2024b); training is stopped early if the validation accuracy fails to improve for 100 (HSDC) or 150 (SWHDC) consecutive epochs, and the checkpoint at the epoch of highest validation accuracy is retained for evaluation.

4.6 Evaluation Protocol

Evaluation is performed once on the official ModelNet test partition, which is held out from the train/validation split. The primary metric is the top-1 overall accuracy (OA),

$$\mathrm{OA} \;=\; \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x},y)\in\mathcal{T}} \mathbf{1}\big[\hat{y}(\mathbf{x}) = y\big], \tag{4.7}$$

where $\mathcal{T}$ is the test set and $\hat{y}(\mathbf{x})$ is the class predicted by the model; OA is the metric reported by every prior work surveyed in Chapter 3. To complement OA in the presence of class imbalance, the mean per-class accuracy (mAcc), the unweighted average of the per-class recalls, is also reported, and the full $K \times K$ confusion matrix is saved for the qualitative analysis of Section 5.6.

The evaluation code also implements a test-time augmentation (TTA) procedure (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), which averages the softmax output across five views: the original ERP, its horizontal flip, and three azimuthal rotations by

$90°$, $180°$, and $270°$. Because the ERP is 2π-periodic along the horizontal axis, these rotations are exact circular shifts of the image columns and introduce no resampling artefacts. However, *TTA is not used in any experiment of this work*: every accuracy in Chapter 5 comes from a single forward pass on the unmodified ERP; the procedure is described only because it ships with the released evaluation code as an optional flag.

5 RESULTS

This chapter reports the results obtained with the methodology of Chapter 4, following the order of the research questions stated in Section 1.3. Section 5.1 describes the data source and the experimental setup. Section 5.2 reports the headline classification accuracies on ModelNet10 and ModelNet40, and Section 5.3 analyses the training dynamics. Section 5.4 compares the RF-ERP results with the geometric-ERP baselines and with the broader ModelNet classification literature, Section 5.5 studies the parameter–accuracy trade-off, and Section 5.6 inspects the trained models qualitatively. The findings are synthesised in Chapter 6.

5.1 Experimental Setup

5.1.1 Data Source: ModelSplat

All experiments use the *ModelSplat* dataset, released as `ShapeSplats/ModelNet_Splats` on the HuggingFace platform by Ma et al. (2025). ModelSplat provides a 3D Gaussian Splat representation of every CAD model from the original ModelNet40 benchmark of Wu et al. (2015): each mesh was converted by training a standard 3DGS optimiser (KERBL et al., 2023) on multi-view renders of the source mesh, so the resulting Gaussian cloud occupies approximately the same spatial region as the original mesh and approximates its surface up to the densification budget used during training. Each object is stored as a binary little-endian PLY file with a variable number of Gaussian primitives that depends on the geometric complexity of the source mesh, and the directory layout preserves the official train/test partition of Wu et al. (2015). Table 5.1 summarises the headline statistics.

Table 5.1: Headline statistics of the ModelSplat dataset used in this work.

Item	**Value**
Total objects	12,309
Categories	40 (ModelNet40 superset of ModelNet10)
File format	Binary little-endian PLY
Gaussians per object	10^4–10^5 (variable)
Source mesh dataset	ModelNet (WU et al., 2015)
Training pipeline	Vanilla 3DGS (KERBL et al., 2023)

Source: the author, compiled from Ma et al. (2025).

Two benchmarks are evaluated: ModelNet10, with ten common indoor categories,

the standard small-scale benchmark in the ERP-based literature (STRINGHINI; JUNG, 2024a; STRINGHINI; JUNG, 2024b); and ModelNet40, with forty categories, the standard large-scale benchmark in the 3DGS-based literature (Section 3.4). The test split is held out and never touched until final evaluation. The training directory is split into an internal training set and a validation set with a single class-stratified hold-out partition: within each class, 80% of the objects go to the internal training set and 20% to validation, so both sets keep the same class proportions. The partition is drawn once with seed 42 and reused in every run; no K-fold cross-validation is performed, since each configuration already costs two to four days of training (Section 6.2). Table 5.2 gives the resulting sample counts.

Table 5.2: ModelNet10 and ModelNet40 sample counts. The training directory is split internally into a training and a validation set with a class-stratified $80:20$ hold-out partition seeded by 42.

Benchmark	Categories	Internal train	Validation	Test
ModelNet10	10	3,192	799	908
ModelNet40	40	7,874	1,969	2,468

Source: the author, derived from the official ModelNet splits in the ModelSplat dataset.

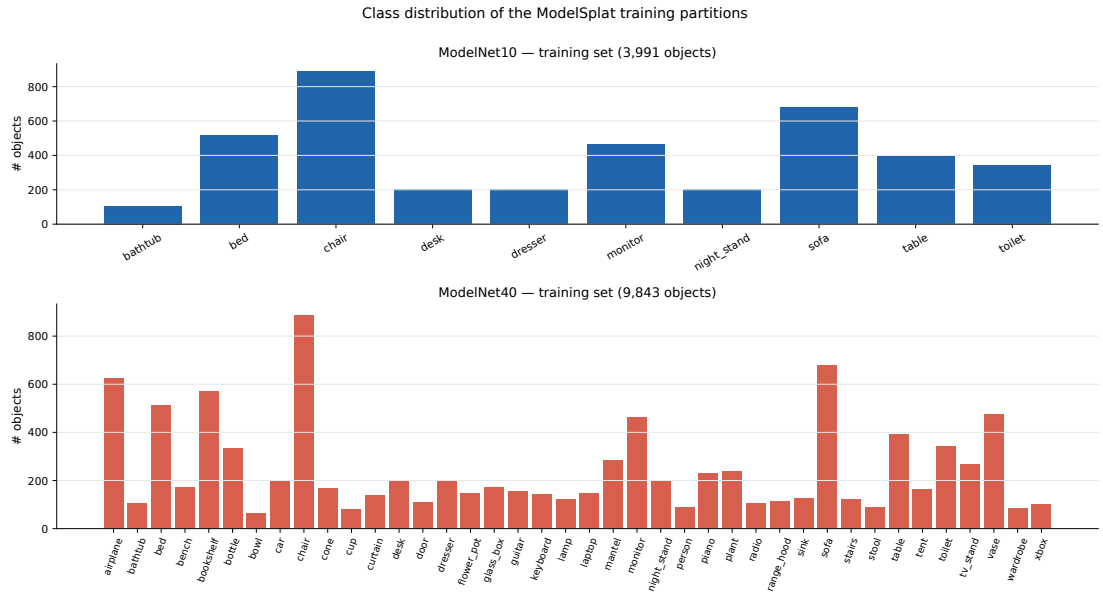
The class distribution of both benchmarks is moderately imbalanced, as shown in Figure 5.1: ModelNet10 has 889 training samples for `chair` but only 106 for `bathtub`, and ModelNet40 has nearly 900 training samples for `chair` but fewer than 90 for `stool` or `wardrobe`. This imbalance justifies the class-weighted cross-entropy loss adopted in Section 4.5.

5.1.2 Hardware, Software, and Preprocessing Cost

All experiments run on a single workstation with one NVIDIA RTX 3090 Ti GPU (24 GB) and an x86-64 CPU, using Python 3.10 and PyTorch (≥ 2.0) with CUDA $12.x$ and Automatic Mixed Precision. The preprocessing pipeline, model definitions, training and evaluation loops, and analysis scripts live in a single Python package, and the YAML configurations make every reported run a single command.

Preprocessing one ModelSplat object into its RF-ERP tensor takes between one and two seconds on a single CPU thread, dominated by the density evaluation of Section 4.2.5. Each cached tensor occupies 4 MB, for a total cache of about 19 GB (ModelNet10) and 48 GB (ModelNet40). The cache is built once before training; every subsequent training and evaluation run reads from it.

Figure 5.1: Per-class training-sample count on the official ModelNet10 (3,991 objects) and ModelNet40 (9,843 objects) splits used by ModelSplat. Both are moderately imbalanced: the largest class (`chair`) has roughly an order of magnitude more samples than the smallest.



Source: the author, from the canonical ModelNet split counts of Wu et al. (2015).

5.1.3 Hyperparameter Configurations

Each experiment is defined by a single YAML configuration file. Eight runs are evaluated: each combination of backbone depth (ResNet-34 vs. ResNet-50) and benchmark (ModelNet10 vs. ModelNet40) is trained both with its distortion-correction block (HSDC on ResNet-34, SWHDC on ResNet-50) and as a plain ResNet baseline, giving four block variants and four ablation baselines. A plain baseline shares every preprocessing and training setting with its block counterpart and differs only in replacing the block with standard convolutions, so the two configurations together isolate the contribution of the block. All eight runs share the preprocessing pipeline and most training hyperparameters (Table 5.3) and differ only in the backbone, the number of classes, and the epoch budget (Table 5.4). The epoch budgets and early-stopping patiences follow Stringhini and Jung (2024a), Stringhini and Jung (2024b); the patience values are large enough to let the cosine annealing schedule traverse its slow-decay phase before a run can be terminated.

Table 5.3: Hyperparameters shared by all four experimental configurations.

Group	**Value**
Preprocessing	
Number of shells N_{shells}	8
ERP resolution	256×512
Shell percentile bounds	$(p_{\text{near}}, p_{\text{far}}) = (5, 95)$
Ray-march steps per shell	$K = 4$
Spatial culling threshold	3σ
Log compression	enabled
Derived channels	pseudo-depth, MIP
Classifier input shape	$10 \times 256 \times 512$
Training	
Optimiser	AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$)
Base learning rate	10^{-4}
Learning-rate floor	10^{-6}
Weight decay	5×10^{-4}
Schedule	linear warmup (10 epochs) + cosine
Label smoothing	$\epsilon = 0.1$
Class-weighted cross-entropy	enabled
MixUp parameter	$\alpha = 0.4$
CutMix parameter	$\alpha = 0.4$ ($50/50$ with MixUp)
Gradient clipping	$\|\nabla\|_2 \leq 1.0$
Batch size	32
Dropout (head)	0.3
Mixed precision (AMP)	enabled
Train/val split	class-stratified $80:20$
Random seed	42
Augmentation	
Horizontal flip probability	0.5
Single-sample augmentation probability	0.3 per primitive

Source: the author.

Table 5.4: Hyperparameters that vary across the four block-variant configurations. Each also has a plain-ResNet ablation baseline with identical settings but standard convolutions in place of the block (≈ 21.3 M for ResNet-34, ≈ 23.6 M for ResNet-50).

Hyperparameter	**HSDC/MN10**	**HSDC/MN40**	**SWHDC/MN10**	**SWHDC/MN40**
Backbone	ResNet-34	ResNet-34	ResNet-50	ResNet-50
Distortion-correction block	HSDC	HSDC	SWHDC	SWHDC
Number of classes K	10	40	10	40
Maximum epochs	500	500	400	400
Early-stopping patience	100	100	150	150
Trainable parameters	≈ 5.5 M	≈ 5.5 M	≈ 23.6 M	≈ 23.6 M

Source: the author.

5.2 Classification Accuracy

Each backbone is trained twice: once with its distortion-correction block and once as a plain ResNet of the same depth and stem, trained from scratch on the identical 10-channel RF-ERP input. The plain variants are the controlled baselines that isolate the contribution of each block: the original papers (STRINGHINI; JUNG, 2024a; STRINGHINI; JUNG, 2024b) compared their blocks against other published methods, but not against a plain ResNet of the same depth trained identically on the same input, so that direct block-versus-plain comparison is added here. The ResNet-50 + SWHDC comparison is exactly capacity-matched: SWHDC adds no learnable parameters, so it shares the exact parameter count of its plain ResNet-50 baseline (23.55 M on ModelNet10), and any accuracy difference reflects only the latitude-weighted reduction. HSDC, in contrast, is narrower by construction, at 5.47 M parameters, about a quarter of the 21.31 M plain ResNet-34 baseline. The eight runs are trained to convergence on the internal training split, which is drawn with a class-stratified $80:20$ partition, and evaluated once on the official test split using the checkpoint with the highest validation accuracy. The headline metrics are overall accuracy (OA) and mean per-class accuracy (mAcc), both in percent.

Table 5.5 reports the test-set performance on ModelNet10. The strongest model is the plain ResNet-50 baseline at 91.52%, ahead of ResNet-50 + SWHDC (89.54%) by almost two percentage points at the same parameter count. The picture differs for the smaller backbone: ResNet-34 + HSDC reaches 90.75% and beats its plain ResNet-34 baseline (89.76%) by about one point while using roughly a quarter of the parameters. The mean per-class accuracy stays within 0.1 percentage points of the overall accuracy in every run, so the class-weighted loss balanced the moderate class imbalance of the benchmark.

Table 5.5: Test-set results on ModelNet10. All four backbones consume the 10-channel RF-ERP and are trained from scratch (Section 4.5). Each block variant is paired with a plain-ResNet baseline of the same depth; SWHDC is capacity-matched to its baseline. Evaluation uses the highest-validation-accuracy checkpoint.

Backbone	**Best val. OA**	**Test OA**	**Test mAcc**	**Params (M)**
ResNet-34 (plain)	95.74%	89.76%	89.83%	21.31
ResNet-34 + HSDC	96.12%	$\mathbf{90.75}\%$	90.70%	5.47
ResNet-50 (plain)	95.49%	$\mathbf{91.52}\%$	91.43%	23.55
ResNet-50 + SWHDC	94.99%	89.54%	89.48%	23.55

Source: the author.

Table 5.6 reports the same evaluation on ModelNet40, where the plain baselines

win outright. ResNet-50 (plain) reaches 88.73%, the best of the four, and ResNet-34 (plain) follows at 88.37% despite being the smaller network. Both block variants trail their baselines: SWHDC reaches 87.11%, 1.62 points below its capacity-matched ResNet-50 baseline, and HSDC reaches 86.38%, 1.99 points below the plain ResNet-34 baseline. Among the two block variants the ordering inverts relative to ModelNet10, with SWHDC now ahead of HSDC, but neither improves on a plain CNN of comparable or larger capacity. The mean per-class accuracy sits three to four percentage points below the overall accuracy across the board, and the gap widens for the block variants, so the blocks lose most of their ground on the rare fine-grained categories of ModelNet40.

Table 5.6: Test-set classification results on ModelNet40. Same setup as Table 5.5.

Backbone	**Best val. OA**	**Test OA**	**Test mAcc**	**Params (M)**
ResNet-34 (plain)	90.86%	88.37%	85.07%	21.33
ResNet-34 + HSDC	89.14%	86.38%	82.04%	5.49
ResNet-50 (plain)	90.86%	$\mathbf{88.73}\%$	86.11%	23.61
ResNet-50 + SWHDC	90.81%	87.11%	83.31%	23.61

Source: the author.

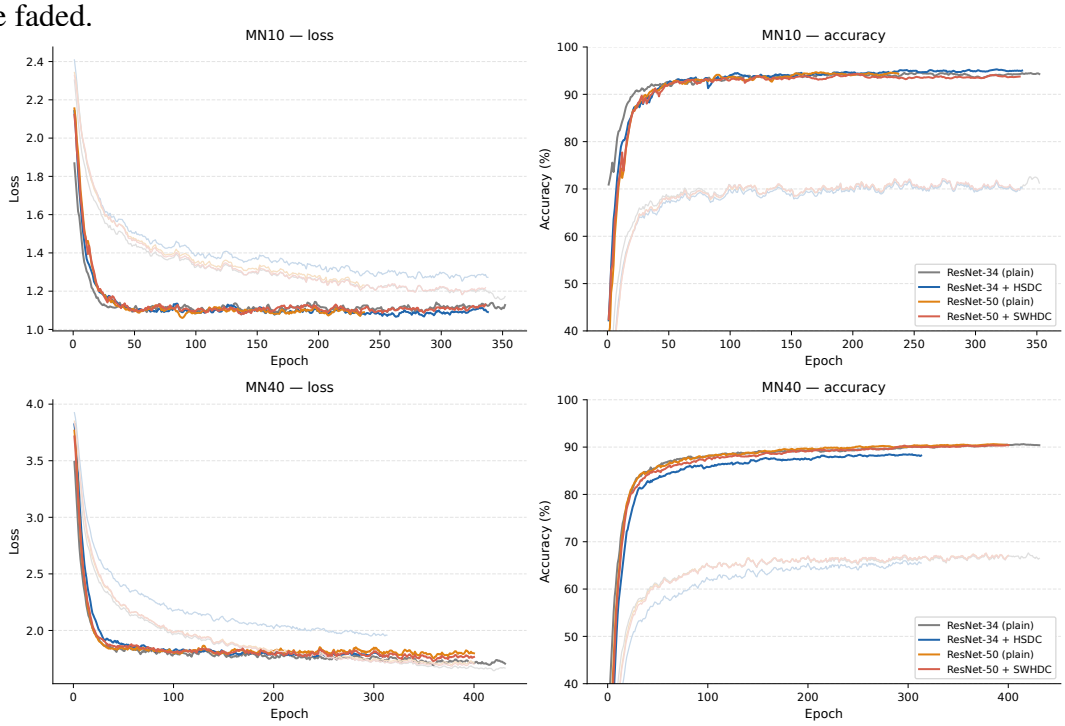
A gap of about five to six percentage points separates the best validation accuracy from the test accuracy on ModelNet10, but narrows to two to three points on ModelNet40. This gap comes from selecting the best checkpoint on a small validation set. ModelNet10 has only 799 validation objects (Table 5.2), so picking the *maximum* validation accuracy over hundreds of epochs tends to catch lucky fluctuations, and the selected number ends up higher than what the model achieves on the test set; the effect shrinks on the larger benchmark. The class-stratified split matters here too: with a global random shuffle, the class proportions of a 799-object validation set can drift away from those of the test split, whereas matching the proportions class by class stabilises the validation metric and keeps the validation and test figures closer together. The official ModelNet test split is left untouched, so the test figures themselves remain unbiased.

5.3 Training Dynamics

Figure 5.2 shows the per-epoch train and validation loss and accuracy for the eight runs, with each block variant overlaid on its plain baseline. Two features are consistent across all eight runs. First, every run plateaus early: each reaches 95% of its peak validation accuracy within the first fifth of its epoch budget (Table 5.7), and the curves are essentially flat well before the best-validation epoch, which ranges from 87 (plain ResNet-

50 on ModelNet10) to 398 (SWHDC on ModelNet40). The late best-validation epochs therefore reflect small fluctuations on an already-flat curve rather than continued learning. Second, training accuracy ends in the 65–70% range while validation accuracy exceeds 88%; this gap is an artefact of the regularisation recipe (MixUp, CutMix, label smoothing, and random erasing make the training samples harder than the clean validation inputs), not a sign of underfitting.

Figure 5.2: Training and validation curves for the eight runs, grouped by benchmark (top: ModelNet10, bottom: ModelNet40; left: loss, right: accuracy). Within each panel the four runs of that benchmark are overlaid; validation curves are solid and training curves are faded.



Source: the author.

Table 5.7 summarises convergence speed as the epoch at which the smoothed validation curve first reaches 90% and 95% of each run's peak validation accuracy. All eight runs reach 95% of their peak within the first fifth of the schedule. The block variants are marginally slower than their plain baselines on ModelNet40, but the difference is a handful of epochs against budgets of several hundred, so the block does not change the qualitative convergence behaviour.

Table 5.7: Convergence speed of the eight runs: epoch at which the EMA-smoothed validation curve first reaches the given fraction of that run's peak validation accuracy. "Peak val. OA" and "Peak epoch" are the raw best-checkpoint values used for evaluation.

Run	90% **peak**	95% **peak**	**Peak val. OA**	**Peak epoch**	**Budget**
ResNet-34 (plain) / MN10	15	28	95.74%	252	500
ResNet-34 + HSDC / MN10	21	43	96.12%	238	500
ResNet-50 (plain) / MN10	21	37	95.49%	87	400
ResNet-50 + SWHDC / MN10	20	37	94.99%	186	400
ResNet-34 (plain) / MN40	26	53	90.86%	331	500
ResNet-34 + HSDC / MN40	30	64	89.14%	213	500
ResNet-50 (plain) / MN40	25	60	90.86%	306	400
ResNet-50 + SWHDC / MN40	30	70	90.81%	398	400

Source: the author.

5.4 Comparison with Prior Work

This section places the RF-ERP results in two contexts: a controlled comparison against the geometric-ERP baselines that share this work's backbones and training recipe (Section 5.4.1), and a broader comparison against the main families of ModelNet classifiers (voxel, multi-view, point-cloud, and 3DGS-based) in Section 5.4.2.

5.4.1 Geometric-ERP Baselines

This work keeps the backbones and training recipe of Stringhini and Jung (2024a), Stringhini and Jung (2024b) almost unchanged and replaces only the input, so the accuracy difference relative to those papers isolates the effect of the input modality. Table 5.8 adds the plain-ResNet RF-ERP baselines alongside the block variants, so the same table separates the effect of the input from the effect of the block.

The first thing the table shows is the size of the drop: both block variants lose between 4.5 and 7.5 percentage points relative to the geometric-ERP results of the original papers. The input modality is the dominant factor: a mesh-derived ERP exposes exact surface geometry, while the 3DGS-derived ERP exposes an optimised RF in which floater Gaussians and reconstruction artefacts (MA et al., 2025) add noise and replace sharp boundaries with soft density. The plain RF-ERP baselines show that this gap is not caused by the block: a plain ResNet on the same RF-ERP input reaches 89.76–91.52% on ModelNet10 and 88.37–88.73% on ModelNet40, inside or above the range of the block variants, so the distortion-correction block is not the bottleneck on this input. The controlled ablation, finally, is unfavourable to the blocks on RF-ERP. The capacity-matched

Table 5.8: Comparison with the geometric-ERP baselines of the original HSDC (STRINGHINI; JUNG, 2024a) and SWHDC (STRINGHINI; JUNG, 2024b) papers. "Geometric ERP" is the ray-cast ERP of those papers, "RF-ERP" the radiance-field ERP of this work; Δ_{OA} is the block's accuracy difference versus its paper baseline.

Backbone	**Input**	**MN10 OA**	**MN40 OA**	**Params (M)**
ResNet-34 + HSDC	Geometric ERP	97.10%	93.90%	5.30
ResNet-34 + HSDC *(ours)*	RF-ERP	90.75%	86.38%	5.47
ResNet-34 (plain) *(ours)*	RF-ERP	89.76%	88.37%	21.31
Δ_{OA} HSDC vs. paper		−6.35 pp	−7.52 pp	—
ResNet-50 + SWHDC	Geometric ERP	94.11%	91.90%	25.50
ResNet-50 + SWHDC *(ours)*	RF-ERP	89.54%	87.11%	23.55
ResNet-50 (plain) *(ours)*	RF-ERP	91.52%	88.73%	23.55
Δ_{OA} SWHDC vs. paper		−4.57 pp	−4.79 pp	—

Source: the author. Paper accuracies are taken from Stringhini and Jung (2024a, Table 4) and Stringhini and Jung (2024b, Table II).

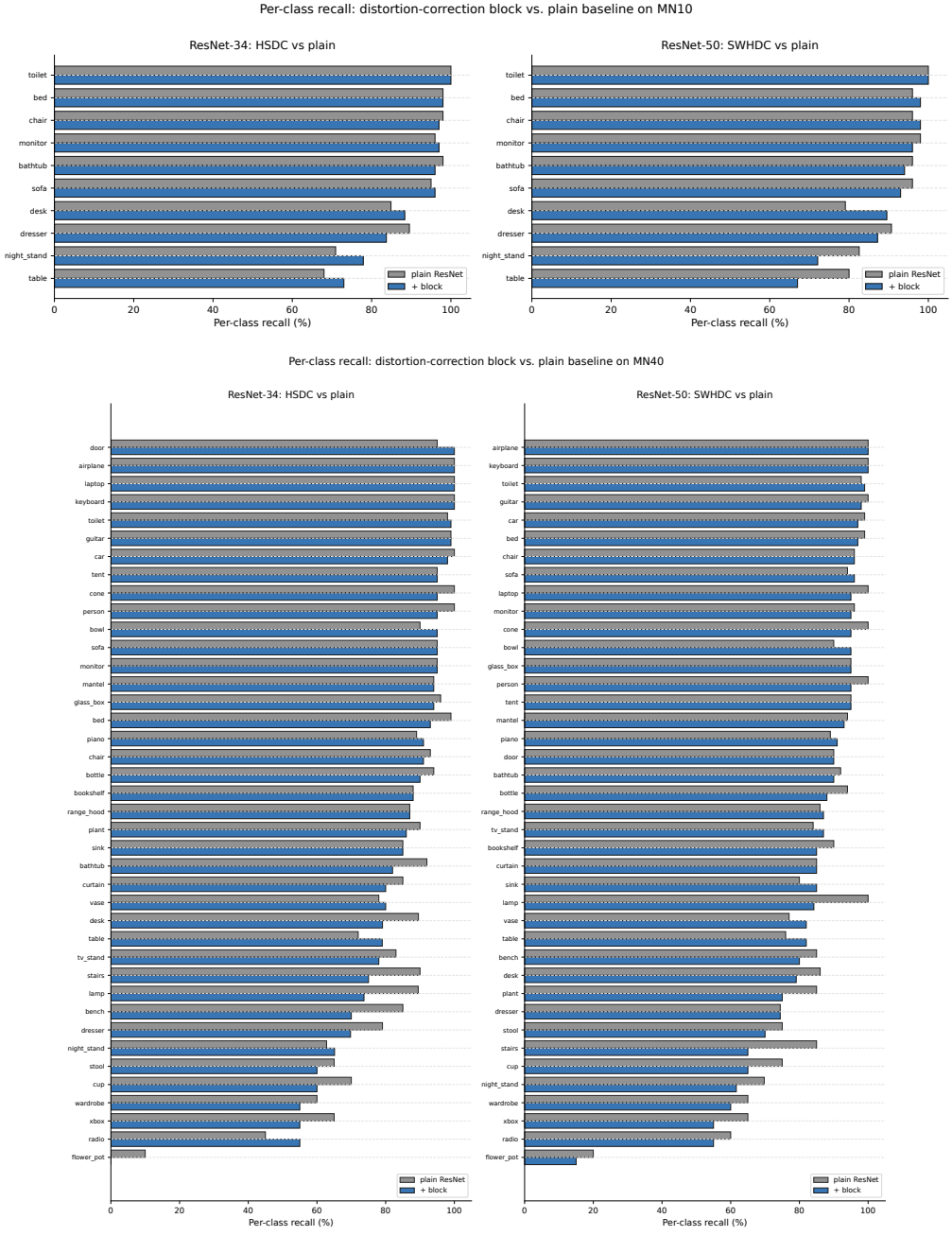
SWHDC underperforms its plain ResNet-50 baseline on both benchmarks (-1.98 pp on ModelNet10, -1.62 pp on ModelNet40), so its parameter-free latitude-weighted reduction is a net negative on the RF input. HSDC, four times smaller than its plain ResNet-34 baseline, improves on it only on ModelNet10 ($+0.99$ pp) and loses on ModelNet40 (-1.99 pp), so its value on RF-ERP is parameter efficiency on the smaller benchmark rather than a consistent accuracy gain. Figure 5.3 resolves this at the class level: the block helps mainly on the classes with strong horizontal-stretching structure that its dilation was designed for, and the advantage does not extend to the fine-grained long tail of ModelNet40, where plain capacity wins.

5.4.2 The Broader ModelNet Literature

The geometric-ERP comparison isolates the input modality but says nothing about where the RF-ERP pipeline stands relative to the wider ModelNet literature. Table 5.9 situates it among the four dominant representation families: volumetric (voxel grids), multi-view (rendered 2D images), point clouds, and the emerging 3DGS-based methods. Gaussian-MAE (MA et al., 2025) is the most directly comparable entry, as it is the strongest published 3DGS classifier and consumes the same ModelSplat dataset.

Table 5.9 supports three comparisons. First, against the older single-representation baselines that, like this work, train from scratch, the RF-ERP pipeline is competitive: at 86.4–88.7% on ModelNet40 it clears the classic voxel networks 3D ShapeNets (77.3%) and VoxNet (83.0%) and the panorama CNN DeepPano (77.6%) by a wide margin, and its best variant trails the multi-view MVCNN (90.1%) by about 1.4 points while using

Figure 5.3: Per-class recall of each block against its plain-ResNet baseline on the same RF-ERP input. Top: ModelNet10; bottom: ModelNet40. Left panels: ResNet-34 + HSDC vs. plain ResNet-34; right panels: ResNet-50 + SWHDC vs. plain ResNet-50. Classes are sorted by the block's recall.



Source: the author.

a fraction of its parameters. MVCNN, moreover, relies on ImageNet-pretrained 2D features and twelve rendered views. A single structured ERP therefore captures more of the object than a low-resolution voxel grid or a single panorama, without the rendering and pretraining cost of the multi-view pipeline.

Second, the modern point-cloud methods set the from-scratch ceiling that the RF-

Table 5.9: The RF-ERP pipeline of this work against representative ModelNet classifiers from the four main representation families ("SSL" = self-supervised pretraining on ShapeNet). Reference accuracies are the single-model figures reported by the cited works; "—" marks a value not reported for that benchmark.

Method	Input modality	Pretraining	MN10 OA	MN40 OA	Params (M)
3D ShapeNets (WU et al., 2015)	Voxel grid	None	83.5%	77.3%	—
VoxNet (MATURANA; SCHERER, 2015)	Voxel grid	None	92.0%	83.0%	≈ 0.9
DeepPano (SHI et al., 2015)	Panorama	None	85.5%	77.6%	—
MVCNN (SU et al., 2015)	Multi-view	ImageNet	—	90.1%	≈ 60
PointNet (QI et al., 2017a)	Point cloud	None	—	89.2%	3.5
PointNet++ (QI et al., 2017b)	Point cloud	None	—	91.9%	1.7
PointMLP (MA et al., 2022)	Point cloud	None	—	94.1%	12.6
Point-MAE (PANG et al., 2022)	Point cloud	SSL ShapeNet	94.9%	93.2%	22.1
Gaussian-MAE (MA et al., 2025)	3DGS params	SSL ShapeNet	95.4%	93.4%	≈ 22.1
ResNet-34 + HSDC *(ours)*	RF-ERP	None	90.8%	86.4%	5.47
ResNet-50 + SWHDC *(ours)*	RF-ERP	None	89.5%	87.1%	23.55
ResNet-50 (plain) *(ours)*	RF-ERP	None	91.5%	88.7%	23.55

Source: the author. Reference accuracies are taken from the cited papers.

ERP input does not yet reach: PointNet++ (91.9%) and PointMLP (94.1%) outperform every proposed variant on ModelNet40 without any pretraining. The most likely cause is input quality rather than architecture: those methods consume uniformly sampled point clouds derived directly from the original meshes, a cleaner geometric signal than anything derivable from ModelSplat, whose optimised Gaussians carry floaters and reconstruction noise (MA et al., 2025; BERG, 2024).

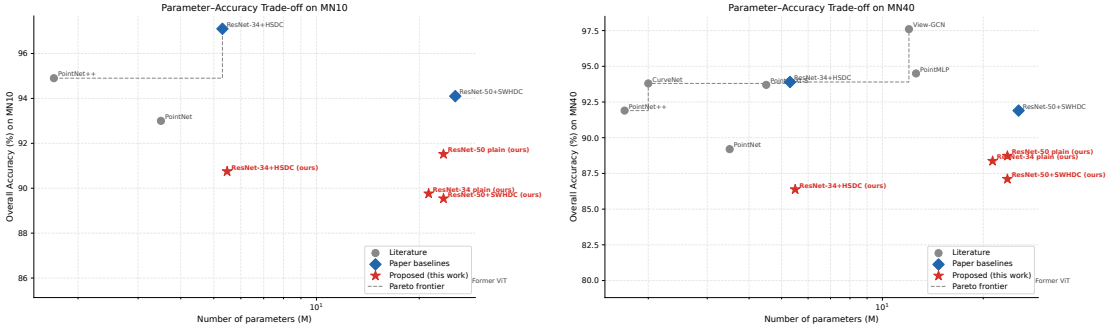
Third, among methods that consume a 3DGS representation, Gaussian-MAE beats the best RF-ERP variant by about four percentage points on ModelNet10 (95.4% vs. 91.5%) and five on ModelNet40 (93.4% vs. 88.7%). Two methodological differences account for the gap: Gaussian-MAE is pretrained with a masked autoencoder on a 52 K-object subset of ShapeNet before fine-tuning, a step known to add three to five percentage points on the same architecture (YU et al., 2022; PANG et al., 2022), while this work trains from scratch; and Gaussian-MAE processes the Gaussian primitives directly as an unordered set, while this work first projects them onto a 256×512 ERP, a lossy step. In return, the proposed pipeline needs at most a quarter of the parameters, no external dataset, and no view rendering, and it is the only entry in Table 5.9 that turns a 3DGS object into a single image classifiable by a standard CNN trained from scratch.

5.5 Parameter Efficiency Analysis

Figure 5.4 reports the parameter–accuracy trade-off for the methods listed in Tables 5.8 and 5.9. Two points define the RF-ERP front. At the low-parameter end, ResNet-34 + HSDC reaches the highest accuracy of any from-scratch method under 10 M pa-

rameters operating on 3DGS-derived input in the surveyed literature (5.47 M, 90.75% on ModelNet10), so the horizontally-dilated shared-weight design is an efficient backbone when the parameter budget is tight. At the high-accuracy end, the plain ResNet-50 baseline (23.55 M) is the strongest RF-ERP model on both benchmarks. SWHDC sits strictly inside the front: at the same 23.55 M parameters as the plain ResNet-50 it scores lower on both benchmarks, so its parameter-free reduction is dominated by its own baseline on this input. Gaussian-MAE dominates every RF-ERP variant in accuracy but needs $\approx$ 22 M parameters and an external ShapeNet pretraining stage.

Figure 5.4: Parameter–accuracy Pareto plot on ModelNet10 (left) and ModelNet40 (right): parameter count (log scale) versus test overall accuracy; the RF-ERP variants of this work occupy the low-parameter region of the front.

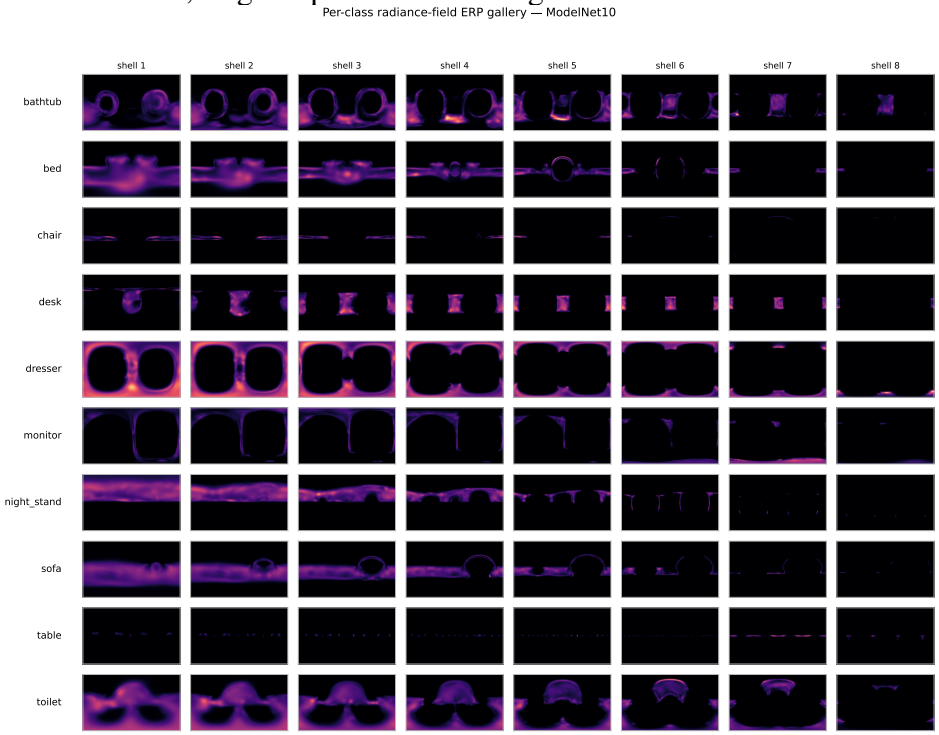


Source: the author.

5.6 Qualitative Analysis

Figure 5.5 shows one representative ModelSplat object per ModelNet10 category after preprocessing. Each row corresponds to one class: the first eight columns show the density shells, ordered from the innermost to the outermost radius, and the two rightmost columns show the derived pseudo-depth and MIP channels (Section 4.3.2) computed from those shells, so each row displays the complete 10-channel input the classifier receives. The gallery makes the discriminative content of the RF-ERP input visible: objects with a strong vertical axis of symmetry (`toilet`, `monitor`, `chair`) produce vertical bands of high density; objects with broad horizontal extents (`table`, `bed`, `sofa`) produce density bands near the equator that propagate through several shells; and the latitude-dependent stretching of the ERP (Section 2.2.1) is visible at the top and bottom of each panel; this is the distortion that the correction blocks target.

Figure 5.5: Per-class RF-ERP gallery on ModelNet10. One representative object per category; the first eight columns show the log-compressed density shells produced by the pipeline of Section 4.2, from the innermost (shell 1, left) to the outermost radius (shell 8), and the two rightmost columns show the derived pseudo-depth and MIP channels of Section 4.3.2; brighter pixels indicate higher values.



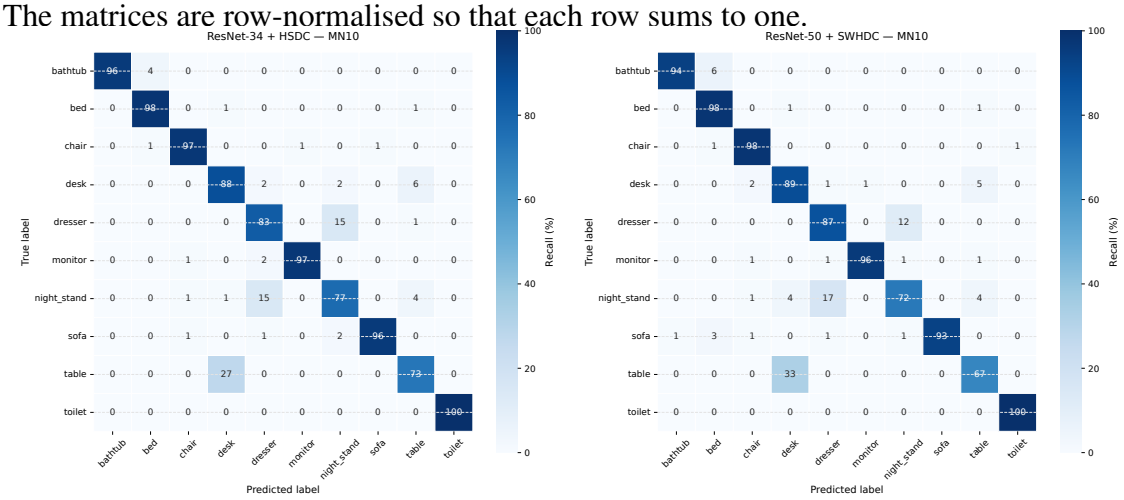
Source: the author.

5.6.1 Confusion Matrices

Figure 5.6 reports the row-normalised confusion matrices of the two block variants on ModelNet10. Both matrices are strongly diagonal, but two category pairs absorb most of the errors. The hardest class is `table`, with 73% recall under HSDC and 67% under SWHDC, confused with `desk` in 27 to 33% of its cases; the two share the horizontal-surface-with-vertical-legs geometry. A secondary, symmetric confusion appears between `dresser` and `night_stand` (about 15% each way), which differ by scale rather than shape. Apart from these pairs the remaining categories exceed 84% recall, and the wider spread under SWHDC mirrors its lower overall accuracy on this benchmark.

Figure 5.7 reports the same matrices on ModelNet40. Both stay mostly diagonal but contain a denser set of off-diagonal entries, dominated by visually ambiguous category pairs: `flower_pot`/`plant`, `cup`/`vase`, `dresser`/`night_stand`, `wardrobe`/`bookshelf`, and `table`/`desk`. The most extreme case is `flower_pot`, whose ModelSplat reconstructions are dominated by the plant they hold: it is read as `plant` in 55 to 65% of

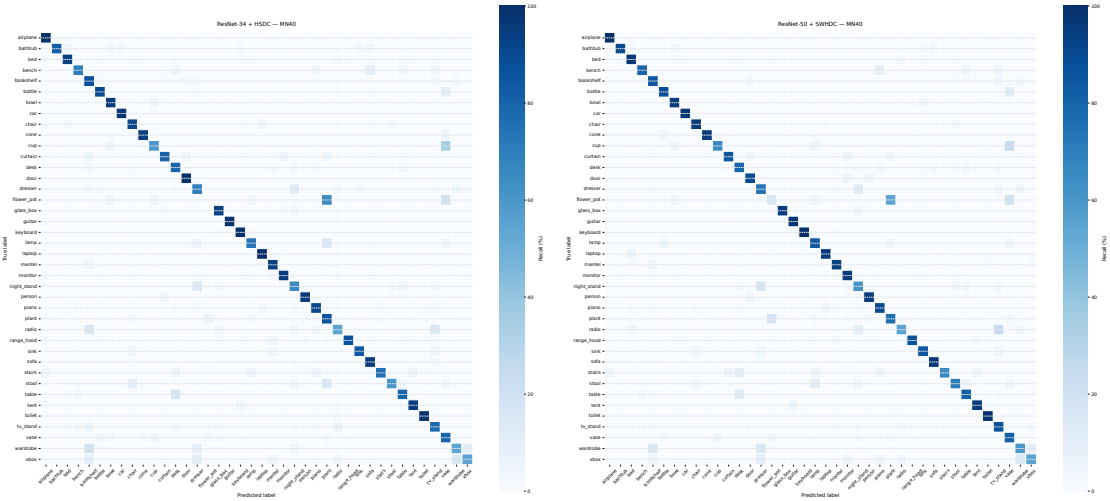
Figure 5.6: Test-set confusion matrices on ModelNet10. Left: HSDC. Right: SWHDC. The matrices are row-normalised so that each row sums to one.



Source: the author.

cases and reaches only 0 to 15% recall. These confusion patterns match those reported by Stringhini and Jung (2024a) on the geometric-ERP baseline and reflect intrinsic ambiguities of the ModelNet40 label space rather than failures specific to the RF-ERP input.

Figure 5.7: Test-set confusion matrices on ModelNet40. Left: HSDC. Right: SWHDC.



Source: the author.

5.6.2 Failure Cases

The hardest categories (`flower_pot`, `plant`, `stairs`, `bookshelf`, `cup`) share a common failure mode: their 3DGS reconstructions are dominated by thin or sparse structures, around which the 3DGS optimiser of Kerbl et al. (2023) tends to produce small

floater Gaussians and a low-density interior. Concentric shells centred at the opacity-weighted centroid intersect thin features only over narrow azimuth ranges, so the resulting RF-ERP shows sparse, high-variance density across shells; the associated errors come from the input modality, not from the distortion-correction blocks. A second failure mode involves objects whose opacity-weighted centroid is far from the geometric centre of the shape because the opacity mass is strongly asymmetric (e.g. a `lamp` with a large opaque base and a thin shade): the spherical camera then sits at the base, leaving the upper half of the ERP largely empty. The same failure mode affects the geometric-ERP pipeline of Stringhini and Jung (2024a) but is more pronounced under the 3DGS input, because the opacity field is optimised rather than derived from a closed mesh.

6 CONCLUSION

This work asked whether the HSDC and SWHDC distortion-correction blocks introduced by Stringhini and Jung (2024a) and Stringhini and Jung (2024b) for ERPs derived from triangle meshes remain effective when the ERP is generated from a 3D Gaussian Splatting representation instead. To answer it, the work produced a preprocessing pipeline that converts 3DGS PLY files into multi-channel RF-ERP tensors; four CNN classifiers trained on this input (the two distortion-correcting backbones and a plain-ResNet ablation baseline for each); and an empirical comparison against both the geometric-ERP baselines that motivated the question and the Transformer-based 3DGS classifiers that define the state of the art on ModelSplat.

The answer is qualified. The blocks train stably from scratch on the RF input and reach competitive accuracy, but on this input they no longer beat a plain CNN of the same depth the way they do on geometric ERP: the parameter-free SWHDC trails a plain ResNet-50 on both benchmarks, HSDC improves on its baseline only on ModelNet10, and the strongest result of all comes from a plain ResNet-50. What carries the work is the data representation itself, not the correction block: the planar, image-structured re-encoding of the volumetric 3DGS data preserves enough of the object's discriminative structure for standard 2D networks to classify it competitively.

6.1 Summary of Contributions

The central contribution of this work is a way of *processing the data*: the RF-ERP pipeline of Section 4.2 is, to the best of our knowledge, the first method that converts a 3D Gaussian Splat into an equirectangular projection through RF sampling rather than geometric ray–mesh intersection. Lacking a surface to intersect, the pipeline accumulates density on concentric, exponentially spaced shells anchored at the opacity-weighted centroid, with a log compression and two derived channels (pseudo-depth and MIP) for explicit depth and silhouette cues. The conversion costs one to two CPU seconds per object and needs no mesh, oriented surface, or clean topology, so it applies to any object reconstructable as 3DGS.

The classification experiments serve both as a validation of this representation and as a controlled ablation of the blocks. All four backbones train stably from scratch on the RF-ERP tensor without any architectural change. The block variants reach accuracies

within 4.5 to 7.5 percentage points of their geometric-ERP counterparts (ResNet-34 + HSDC: 90.75% / 86.38% on ModelNet10 / ModelNet40; ResNet-50 + SWHDC: 89.54% / 87.11%), a gap attributable mainly to the information loss of the RF input rather than to the blocks. The plain baselines, trained here specifically as controls, make the result easier to read: the best RF-ERP model is in fact a plain ResNet-50 (91.52% / 88.73%), the parameter-free SWHDC reduction trails its capacity-matched baseline on both benchmarks, and HSDC improves on its baseline only on ModelNet10. In other words, the blocks train without problems on this noisier input, but they no longer give a reliable accuracy gain; what preserves the discriminative structure of the shape is the representation, not the block. As a by-product, the HSDC variant is a small model that needs no pretraining and still stays competitive, at roughly a quarter the parameter count of Gaussian-MAE, and the complete preprocessing, training, and analysis code is released as an open-source, reproducible codebase.

Placed against the broader literature, the pipeline stays competitive despite the RF detour (Figure 5.4): the best RF-ERP model clears the classical voxel and panorama baselines, approaches the multi-view MVCNN without its rendering or pretraining cost, and trails the strongest from-scratch point-cloud networks and the pretrained Gaussian-MAE by a margin that pretraining alone is known to cover. The central result is methodological: a single structured image, classifiable by a standard CNN, recovers most of the discriminative content of an optimised RF, and the remaining accuracy gap is an acceptable price for a representation that applies to any 3DGS object.

6.2 Limitations

This work has five main limitations. First, the from-scratch training regime enables a like-for-like comparison with the original HSDC and SWHDC papers but concedes the top of the absolute accuracy ranking to methods that exploit large auxiliary datasets, such as Gaussian-MAE (MA et al., 2025). Second, each configuration is trained once, with a single seed, so every reported accuracy is a point estimate without a mean$\pm$std analysis. This matters because the block-versus-baseline margins on which the ablation rests are only one to two percentage points, small enough that a multi-seed study could shift them. Training with a single seed was a practical necessity: each configuration takes two to four days of wall-clock time to train. Third, the RF-ERP-specific hyperparameters (number of shells and their spacing law, ray-march steps, ERP resolution, log compres-

sion, derived channels, augmentation probabilities) are fixed by design rather than ablated. Fourth, the pipeline ingests the ModelSplat PLY files as-is, with no Gaussian filtering. This matters for *floaters*: small, typically low-opacity Gaussians that RF optimisation leaves drifting away from the object surface (PHILIP; DESCHAINTRE, 2023; MA et al., 2025). Floaters enter the density field of Equation (2.3) like any other primitive. The percentile shell bounds of Section 4.2.3 keep them from inflating the radial sampling range, but do not remove their density contribution, and they account for a measurable share of the failure modes of Section 5.6.2. A pre-filtering step, such as discarding Gaussians below an opacity threshold or applying statistical outlier removal to the centre distances, was not explored. Likewise, the opacity-weighted centroid can drift from the geometric centre of the object when the opacity mass is strongly asymmetric. The expected impact of both effects is concentrated on specific classes rather than spread uniformly. Floaters hurt mostly the categories with thin or sparse geometry (`flower_pot`, `plant`, `lamp`), whose true density is low enough for the spurious contributions to compete with the real ones, while a drifted centroid leaves part of the panorama empty and pushes the object away from the low-distortion equatorial band, wasting ERP resolution. Both effects depress the per-class recalls of Figure 5.3, and filtering should recover part of that loss, but quantifying how much requires the ablation left as future work. Fifth, the representation is geometry-only by design (Section 4.2.5): the spherical-harmonic coefficients play no role in the volumetric density of Equation (2.3), so discarding them does not alter the density channels, but the appearance information they carry, which could be discriminative in its own right (for example, rendered as additional per-pixel colour channels), is left unused by the pipeline.

6.3 Future Work

Several directions follow from this work. The most immediate is to test the RF-ERP representation under other classification architectures. One candidate is a Vision Transformer adapted to the spherical domain, such as the panorama-aware PanoFormer (SHEN et al., 2022), which could exploit the global azimuthal context of the ERP that a convolutional backbone captures only locally. Another is the complementary block–backbone pairings (HSDC on ResNet-50, SWHDC on ResNet-34), for which no published reference exists (Section 4.4).

A second direction is a systematic, multi-seed ablation of the representation's fixed

hyperparameters: the number of shells and their spacing law, the ray-march resolution K, the ERP resolution, the number of dilation branches, and the contribution of each derived channel. Three of these analyses look the most promising: the sensitivity of the projection to the virtual camera centre, since the failure modes of Section 5.6.2 suggest that off-centre centroids degrade the ERP; the per-shell contribution to the decision, since the sparse outer shells may add more noise than signal; and the derived channels, which unlike the independent feature channels of the geometric ERP (STRINGHINI; JUNG, 2024a) are deterministic functions of the shells they are stacked with, a redundancy that may help as an inductive bias or merely inflate the input. A related question is whether filtering floater Gaussians before projection (Section 6.2) recovers part of the gap to the geometric ERP.

Beyond the representation, two extensions could close part of the gap to Gaussian-MAE. The first is self-supervised pretraining on the 52 K-object ShapeSplats subset, with a masked-autoencoder objective on the RF-ERP. The second is an appearance-aware variant of the projection that renders the spherical-harmonic colour, which the current pipeline leaves unused, into additional per-pixel ERP channels; it would test how much of the remaining signal is chromatic rather than geometric.

REFERENCES

BERG, L. van den. **Utilising 3D Gaussian Splatting for PointNet Object Classification**. Dissertation (BSc Thesis) — Delft University of Technology, 2024. Available at: <https://repository.tudelft.nl/record/uuid:ef9e967a-92bf-4ad6-83da-7d6e0b4fd470>.

CAO, Z.; HUANG, Q.; KARTHIK, R. 3D object classification via spherical projections. In: **International Conference on 3D Vision (3DV)**. [S.l.: s.n.], 2017.

CHANG, A. X. et al. **ShapeNet: An Information-Rich 3D Model Repository**. 2015.

CHOI, C.; KIM, S. M.; KIM, Y. M. Balanced spherical grid for egocentric view synthesis. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2023. p. 16590–16599.

COHEN, T. S. et al. Spherical CNNs. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2018.

COORS, B.; CONDURACHE, A. P.; GEIGER, A. SphereNet: Learning spherical representations for detection and classification in omnidirectional images. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 518–533.

GUO, Y. et al. Deep learning for 3D point clouds: A survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 43, n. 12, p. 4338–4364, 2021.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778.

JAMES, J. **3D Gaussian Point Encoders**. 2025.

KANEZAKI, A.; MATSUSHITA, Y.; NISHIDA, Y. RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: **IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2018.

KERBL, B. et al. 3D Gaussian Splatting for real-time radiance field rendering. In: **ACM Transactions on Graphics (SIGGRAPH)**. [S.l.: s.n.], 2023. v. 42, n. 4.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2012.

LI, H. et al. UniGS: Unified language-image-3D pretraining with Gaussian splatting. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2025. ArXiv:2502.17860.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2019.

LUO, Y. et al. GS-PT: Exploiting 3D Gaussian splatting for comprehensive point cloud understanding via self-supervised learning. In: **Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2025. ArXiv:2409.04963.

MA, Q. et al. ShapeSplat: A large-scale dataset of Gaussian Splats and their self-supervised pretraining. In: **Proceedings of the International Conference on 3D Vision (3DV)**. [S.l.: s.n.], 2025. ArXiv:2408.10906.

MA, X. et al. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2022.

MATURANA, D.; SCHERER, S. VoxNet: A 3D convolutional neural network for real-time object recognition. In: **Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2015. p. 922–928.

MESCHEDER, L. et al. Occupancy networks: Learning 3D reconstruction in function space. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019. p. 4460–4470.

MILDENHALL, B. et al. NeRF: Representing scenes as neural radiance fields for view synthesis. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2020.

PANG, Y. et al. Masked autoencoders for point cloud self-supervised learning. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2022.

PAPADAKIS, P. et al. PANORAMA: A 3D shape descriptor based on panoramic views for unsupervised 3D object retrieval. **International Journal of Computer Vision**, v. 89, n. 2–3, p. 177–192, 2010.

PARK, J. J. et al. DeepSDF: Learning continuous signed distance functions for shape representation. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019. p. 165–174.

PHILIP, J.; DESCHAINTRE, V. Floaters no more: Radiance field gradient scaling for improved near-camera training. **Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)**, v. 42, n. 4, 2023.

QI, C. R. et al. PointNet: Deep learning on point sets for 3D classification and segmentation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 652–660.

QI, C. R. et al. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2017.

QIAN, G. et al. PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2022.

RAMAMOORTHI, R.; HANRAHAN, P. An efficient representation for irradiance environment maps. **Proceedings of SIGGRAPH**, p. 497–500, 2001.

SFIKAS, K.; PRATIKAKIS, I.; THEOHARIS, T. Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval. **Computers & Graphics**, v. 71, p. 208–218, 2018.

SFIKAS, K.; THEOHARIS, T.; PRATIKAKIS, I. Exploiting the PANORAMA representation for convolutional neural network classification and retrieval. In: **Eurographics Workshop on 3D Object Retrieval (3DOR)**. [S.l.: s.n.], 2017.

SHEN, Z. et al. PanoFormer: Panorama transformer for indoor 360° depth estimation. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2022.

SHI, B. et al. DeepPano: Deep panoramic representation for 3-D shape recognition. **IEEE Signal Processing Letters**, v. 22, n. 12, p. 2339–2343, 2015.

SHOEMAKE, K. Animating rotation with quaternion curves. In: **Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)**. [S.l.: s.n.], 1985. p. 245–254.

SILVEIRA, T. L. T. da et al. 3D scene geometry estimation from 360° imagery: A survey. **ACM Computing Surveys**, v. 55, n. 4, p. 1–39, 2022.

STRINGHINI, L. A. W.; JUNG, C. R. Single-panorama classification of 3D objects using horizontally stacked dilated convolutions. In: **Proceedings of the IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2024.

STRINGHINI, L. A. W.; JUNG, C. R. Spherically-weighted horizontally dilated convolutions for omnidirectional image processing. In: **Proceedings of the Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.: s.n.], 2024.

STRINGHINI, R. M. **Convolutional and Attention Mechanisms for 3D Object Classification Using Panoramas and Beyond**. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, Instituto de Informática, Porto Alegre, 2025.

SU, H. et al. Multi-view convolutional neural networks for 3D shape recognition. In: **IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015.

SU, Y.-C.; GRAUMAN, K. Learning spherical convolution for fast features from 360° imagery. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2017.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 2818–2826.

WALLIS, J. W. et al. Three-dimensional display in nuclear medicine. **IEEE Transactions on Medical Imaging**, v. 8, n. 4, p. 297–303, 1989.

WU, Z. et al. 3D ShapeNets: A deep representation for volumetric shapes. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 1912–1920.

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2016.

YU, X. et al. Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2022.

YUN, S. et al. CutMix: Regularization strategy to train strong classifiers with localizable features. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2019. p. 6023–6032.

ZHANG, H. et al. mixup: Beyond empirical risk minimization. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2018.

ZHONG, Z. et al. Random erasing data augmentation. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2020. p. 13001–13008.