Autoencoding Gaussian Splats in 2D

Rok Mokotar[*]
LMU Munich
Rok.Mokotar@campus.lmu.de

Federico Bernardo Harjes Ruiloba[*]
LMU Munich
f.harjes@campus.lmu.de

Abstract

TODO - at the end :D

1. Introduction

Recent advances in neural rendering and generative modeling have demonstrated the effectiveness of Gaussian splatting for representing images and 3D scenes. In this work, we explore the novel task of autoencoding for Gaussian splats, where we leverage autoencoders (AEs) to learn compact representations of images modeled as 2D Gaussian splats. Our primary objectives are twofold: *(1)* to construct and train an autoencoder for Gaussian splat representations of images and *(2)* to compare its performance against a standard autoencoder trained directly on raw image data.

The importance of this problem arises from the growing need for efficient and flexible representations of visual data. Gaussian splats offer a structured and parametric alternative to pixel-based representations, encapsulating key image features such as position, scale, rotation, opacity, and color in a compact form. This representation aligns well with modern neural rendering techniques and has the potential to enhance interpretability, adaptability, and compression efficiency in learned visual representations.

We adopt Gaussian splats and autoencoders for this task due to their complementary strengths. Gaussian splatting has proven to be a powerful representation technique for reconstructing visual data with high accuracy while being inherently adaptable to multi-resolution settings. Autoencoders, on the other hand, provide a well-established framework for extracting meaningful latent representations and reducing data dimensionality in an unsupervised manner. By combining these approaches, we aim to investigate whether autoencoding in the Gaussian splat space can yield benefits in terms of compression efficiency, reconstruction quality, and feature disentanglement compared to standard pixel-based autoencoding.

A key focus of this work is on the compression capabilities of Gaussian splats. Unlike traditional pixel-based approaches, where compression often relies on feature extraction in a high-dimensional space, Gaussian splats inherently provide a more structured, lower-dimensional representation of images. This opens up opportunities for novel encoding strategies that take advantage of the underlying parametric nature of the splat representation.

Our contributions in this study include:

- The creation of a dataset consisting of trained Gaussian splats derived from CIFAR-10 images, enabling further research on learned splat representations.
- The design and implementation of an autoencoder specifically tailored for Gaussian splats, exploring different architectural choices and their impact on reconstruction quality.
- A comparative analysis of Gaussian splat-based autoencoding against conventional pixel-based autoencoding to evaluate reconstruction performance and compression efficacy.

By addressing these objectives, we aim to provide insights into the potential of Gaussian splats as a learned visual representation and contribute to the broader research on neural compression and generative modeling.

The remainder of this report is structured as follows: Section 2 provides essential background on Gaussian splatting and autoencoders. Section 3 details our dataset, splatting process, autoencoder design, and experimental setup. Section 4 presents our findings on different splatting techniques, autoencoder architectures, and various evaluation approaches. Section 5 interprets the results, highlighting key insights, limitations, and future directions. Finally, Section 6 summarizes our contributions.

2. Background

2.1. Gaussian Splats

Recently, scene representation and novel-view synthesis techniques making use of machine learning methods have

[*]Both authors are students at Ludwig Maximilian University of Munich, Geschwister-Scholl-Platz 1, 80539 Munich, Germany. They contributed equally to this work in all aspects, including conceptualization, research, and writing. The order of authorship was determined randomly and does not reflect any difference in their contributions.

gained attention. One of the most popular frameworks in this category are Neural Radiance Fields (NeRFs) [], able to produce high quality implicit representations of scenes. Typically, NeRFs do this by optimizing a deep neural network using a volumetric continuous representation of the scene, using thechniques such as volumetric ray marching. In spite of various improvements to increase the efficiency of this framework [], achieving high visual quality through NeRFs remains computationally expensive due to the training cost of the neural network, in addition to a high rendering cost. To address these issues, the Gaussian Splatting (GS) framework was proposed.

Through Gaussian Splatting, instead of learning a continuous implicit representation of a scene, scenes are explicitly represented through a set of points using a large number of Gaussian primitives. The Gaussian ellipsoids constituting a scene have a set of learnable parameters controlling properties of the reconstruction, such as position, opacity, anisotropic covariance, and spherical harmonic (SH) coefficients. In addition to providing a high reconstruction quality, Gaussian Splatting provides much faster rendering, being able to produce novel-views in real time. GS achieves this by using rasterization-based rendering, which, in contrast to the rendering used by NeRFs, does not require sampling points.

Although GS was originally formulated for the reconstruction of 3D-scenes, and is often studied in that domain, this work considers 2D GS. Considering 2D GS alleviates some of the challenges involved in the process of auto-encoding Gaussian Splats.

2.2. Auto-Encoders

Auto-Encoders (AEs) [] are one of the most widely used models in the field of unsupervised learning. The first component of a conventional AE is the encoder, which maps the input data into a lower-dimensional *latent* space. The second component of this architecture is the decoder, which reconstructs the original input from the reduced latent space. The primary function of AEs is to learn a compact and efficient representation of the input data in the latent space []. This compact representation has proven useful for feature extraction [] and dimensionality reduction [] in particular.

Over time, various modifications have enhanced the capabilities of autoencoders. Some notable variants include Denoising Autoencoders (DAE) [], which introduce noise into input data to improve robustness; Sparse Autoencoders (SAE) [], which enforce sparsity constraints on the hidden layer for better feature selection; and Variational Autoencoders (VAE) [], which integrate probabilistic modeling for generative applications and Convolutional Autoencoders (CAE) [], which leverage convolutional layers for an improved structural understanding of image data.

In this work, different variations of AEs are applied to Gaussian Splats, investigating the potential of generating meaningful, accurate and compact representations of Splat data.

3. Methodology

3.1. Dataset

In our study, we used the CIFAR-10 dataset [4], a widely recognized benchmark for machine learning and computer vision algorithms. It consists of 60,000 color images of size 32×32 pixels, categorized into 10 evenly distributed classes. The dataset is originally divided into 50,000 training images and 10,000 test images.

As mentioned earlier, our goal was to construct a new dataset of trained Gaussian splats, where each Gaussian encapsulates local image features based on the CIFAR-10 dataset. We generated this dataset by mapping each image to a set of five splat parameters: position (mean), scale, rotation (quaternion), opacity, and color. Each parameter is represented as a $1024 \times N$ matrix, capturing the spatial, color, and intensity distributions across the image. The dataset is implemented as a `PyTorch` dataset [5] and is partitioned into training, validation, and test sets using a $4 : 1 : 1$ split.

3.2. Gaussian splatting

To construct a dataset suitable for training an autoencoder for Gaussian Splats, it was essential to first establish an effective method for generating high-quality Gaussian representations of images.

Gaussian representation and optimization. We employed the `gsplat` library [7] to convert images into Gaussian splats. The primary objective was to optimize the placement and parameters of the Gaussians to achieve the most accurate rasterization possible. Various configurations and hyperparameters were explored to assess their impact on the accuracy of the reconstructed images. The implementation was modular, allowing for flexible adjustments and systematic evaluation of different settings.

A key aspect of our methodology was the optimization process, where we implemented and tested several parameter learning strategies, including:

- training iterations and learning rate: adjusting the number of optimization steps and the step size for parameter updates,
- loss functions: evaluating different combinations of loss functions (L1, L2, SSIM) to determine their impact on image reconstruction quality,
- regularization strategies: applying constraints on scales and opacities to prevent degenerate solutions,
- optimization techniques: experimenting with group optimization and adaptive gradient strategies such as selective Adam and sparse gradient methods,

- scheduling and optimization strategies: implementing various learning rate schedulers and optimization algorithms to improve convergence.

Extended functionality and dataset variants. To enhance the flexibility of Gaussian splatting, we incorporated additional features, including:

- selective learning of splat parameters: allowing control over which Gaussian parameters are optimized during training,
- support for 2D and 3D rasterization: implementing both standard 2D rasterization and extending compatibility with custom 3D rasterization techniques [3],
- bilateral guided radiance support: integrating methods for improved radiance-based rendering [6].

Furthermore, we explored different initialization strategies to generate diverse dataset variants, implementing three approaches:

- random initialization: assigning Gaussian parameters randomly within predefined bounds,
- grid-based initialization: placing Gaussians on a structured grid for uniform coverage,
- KNN-based initialization: distributing Gaussians based on a nearest-neighbor approach to better approximate image structures.

These variations allowed us to construct multiple datasets tailored to different experimental conditions, enabling a comprehensive evaluation of autoencoding techniques for Gaussian splats.

Autoencoder architectures. After generating Gaussian splats, we implemented three distinct autoencoder architectures: a deep autoencoder, a convolutional autoencoder, and a ResNet-based autoencoder. Each model was designed as an implementation of our abstract autoencoder module in `PyTorch` [5], ensuring architectural flexibility and a standardized training pipeline.

The deep autoencoder processes an input vector of dimension 23552, encoding it into an N-dimensional latent space through four fully connected feed-forward layers. The decoder is a mirrored version of the encoder, with a final $\tanh$ activation to constrain outputs within the original value range.

The convolutional autoencoder takes as input a $32 \times 32 \times N$ matrix representation, where N denotes the number of channels. The encoder consists of three sequential convolutional and max-pooling layers. The decoder, built using three transposed convolutional layers, reconstructs the original input with a $\tanh$ activation.

The ResNet-based autoencoder follows the architecture of ResNet-18 [2], with modifications inspired by the convolutional autoencoder. The residual connections enhance gradient flow, improving learning stability and convergence.

Experimental setup and training. To evaluate the models, we conducted multiple experiments tailored to different representations of Gaussian splats:

- vector-based encoding: the deep autoencoder was tested on a flattened representation, combining all Gaussian parameters into a single vector to assess the importance of spatial information,
- full-image encoding: the convolutional and ResNet-based models were trained on a transformed image representation where all 23 parameter channels were combined and learned simultaneously,
- single-channel encoding: an alternative approach involved training models on individual parameter channels, treating them as separate grayscale images to examine per-parameter learning efficiency,
- independent parameter models: a final experiment trained a distinct autoencoder for each splatting parameter, allowing for independent optimization but at the cost of increased complexity.

Hyperparameter optimization. A crucial part of our methodology was optimizing hyperparameters to enhance performance. The training pipeline systematically explored the following factors:

- latent dimension: determining the optimal size of the compressed representation for effective reconstruction.
- learning rate and weight decay: evaluating their impact on stability and convergence using the Adam optimizer.
- number of epochs and gradient clipping: preventing instability in training dynamics,
- initialization strategies: comparing standard and Xavier weight initialization techniques,
- regularization techniques: assessing dropout and batch normalization effects on generalization,
- learning rate scheduling: experimenting with different schedulers to adapt learning dynamics.

4. Results

4.1. Gaussian Splatting

Hyperparameter tuning. To accomplish the first goal of this work, producing a dataset of Gaussians accurately representing CIFAR-10 images, we conducted a thorough hyperparameter optimization using the `gsplat` library.

As expected, the number of Gaussian primitives used was the most influencing factors on reconstruction quality. We tested using 256, 1024 and 4096 Gaussians, and found 1024 to be a good compromise between reconstruction quality and training efficiency. More interestingly, the opacity regularization factor had a particularly large impact on the optimization result. The initial scale of the Splats also played an important role. More details about the hyperparameter tuning process can be found in Fig. **??**.

Training and dataset generation. After finding the optimal hyperparameters for reconstructing the CIFAR-10 images with a high fidelity through Gaussian Splats, we created a dataset reconstructing CIFAR-10 in its entirety using Gaussian Splats. L In addition to the hyperparameters mentioned above, we found out that among the three initialization strategies — random, grid-based and KNN-based — the grid and KNN strategies clearly yielded better reconstructions. For this reason, we decided to generate two datasets of trained splats for the $60,000$ CIFAR-10 images corresponding to these different initialization strategies.

The rasterized images corresponding to the trained Splats resemble the original images accurately for both the grid- and KNN-based initializations, as Fig. **??** depicts. While the KNN-based Splats achieved lower losses during training, the grid-based images are visually closer to the original images in general, as can be noticed in the images corresponding to the automobile, deer, truck and ship classes. In some instances, the KNN-based Splats achieved a higher accuracy on local details, such as for the frog's eye or the cat's rings. A further observation is that both initialization strategies struggled with white areas and very bright areas of the images. This is particularly evident for the cat class and smaller areas of other images, such as the back of the automobile image. This problem is slightly more prominent for the KNN-based initialization.

Both generated Gaussian Splat datasets were implemented as `PyTorch` datasets using a $4:1:1$ split for training, validation, and test sets. The full datasets have a size of approximately 5 GB.

4.2. Auto-encoding of 2D Splats

Model comparison. TODO: write on the performance (quality of the reconstruction for each model) TODO: loss plot here, emphasize that we did hyperparameter tuning for all models on the plot -¿ maybe mention the hyperparams used or use a small table. TODO: write on the compute performance + limitation of model 2 to using batchsize 1

Reconstructing the original images. TODO: put in the image of the generated outputs of one model of each class, maybe cherrypick

5. Discussion

As mentioned previously, one of the primary objectives of this study was to perform a comparative analysis between our Gaussian splat-based ResNet autoencoder and a more conventional pixel-based ResNet autoencoder. Both approaches, utilizing the same model architecture, were evaluated in terms of compression efficiency and reconstruction performance.

For our model, we trained a ResNet architecture on five separate models, each corresponding to a distinct Gaussian splat parameter, based on our newly created CIFAR-10 dataset of Gaussian splats. In contrast, for the conventional approach, we employed the ResNet-18 model implementation[1] and trained it on the original CIFAR-10 dataset.

Figure **??** illustrates the comparison between the two approaches using random test images from each class. The results indicate that our Gaussian splat-based model struggles to reconstruct images effectively, primarily capturing certain relationships related to rotations and colors while failing to preserve finer details and structures. Quantitative analysis, such as the Structural Similarity Index (SSIM), supports this observation, with the SSIM difference between the original and our model being TODO, while the difference between the original and the conventional model is TODO.

Moreover, when comparing the compression ratios between both approaches, the conventional method remains more efficient, achieving a compression ratio of TODO, whereas our model achieves a compression ratio of TODO. This suggests that our approach not only fails to maintain high reconstruction accuracy but also results in inefficient compression, in some cases even increasing the image size. If the compression ratio exceeds 1, further investigation is required to determine whether this is an inherent limitation of our method or a fundamental trade-off associated with representing images using Gaussian splats.

5.1. Future Work

Finally, we have identified several potential areas for future research and applications where this approach could be utilized:

- **Exploring the effect of latent space size on reconstruction error:** It would be valuable to investigate how the reconstruction error sinks as the size of the latent space increases. Understanding this relationship could help optimize the balance between compression efficiency and reconstruction quality.
- **Loss function refinement:** An interesting avenue of investigation would be to explore whether the autoencoder should be trained by minimizing the loss based only on the Gaussian splats, or if it would be beneficial to enforce consistency between the rendered splat and the original image. This could potentially improve the reconstruction accuracy or reduce artifacts.
- **Implementation of Hierarchical Perceiver (HiP):** Incorporating HiP, as discussed in [1], could be an interesting future work. The hierarchical nature of HiP may

[1] `https://github.com/eleannavali/resnet-18-autoencoder`

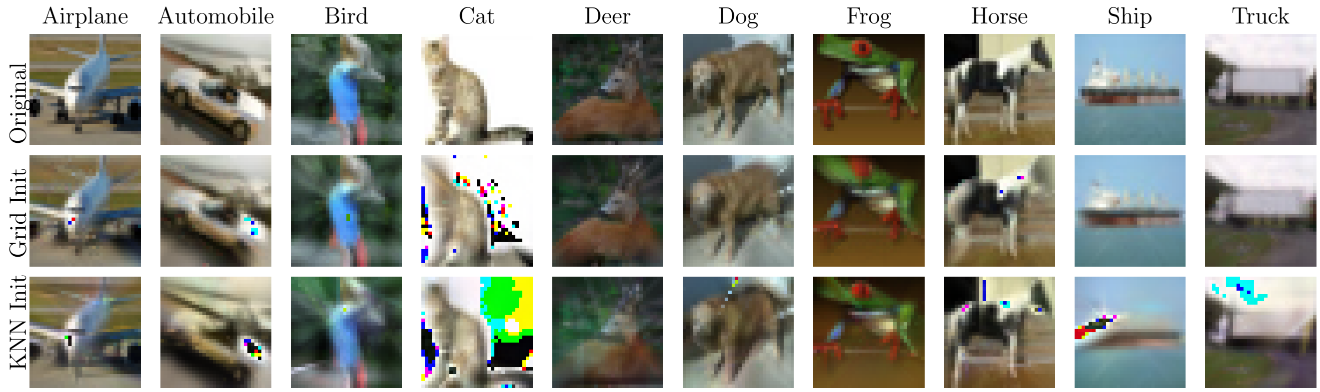

Figure 1. **Gaussian Splat Reconstruction of the CIFAR-10 Classes.** Example reconstructions for one of each CIFAR-10 classes. The top row shows the original image, while the middle and lower rows show the Gaussian Splat reconstructions using the grid- and KNN-initializations respectively.

provide better structure to the encoding process, enhancing performance on Gaussian splats.

- **Latent space classification:** Exploring the possibility of using the latent space for downstream tasks, such as classification, could open up new applications for Gaussian splat-based representations.
- **Generative modeling on the latent space:** Building a generative model (e.g., Generative Adversarial Networks or Stable Diffusion) on top of the latent space could allow us to generate new images or 3D scenes from the compressed representations, facilitating content creation or data augmentation tasks.
- **Gaussian splat generation:** Developing a model capable of generating new Gaussian splats (e.g., a Variational Autoencoder) would be an intriguing direction. This could lead to more sophisticated image synthesis techniques, allowing us to generate realistic images directly from the splat parameters.
- **Masked Autoencoders (MAE) for smaller images:** Finally, investigating the application of MAE on smaller images could help determine if this approach could provide more efficient representations and faster processing times, especially in cases where the dataset is constrained in size.

6. Conclusion

In this study, we explored the use of Gaussian splats for autoencoding 2D images with a ResNet-based architecture. While we showed that Gaussian splats are capable of reconstructing images with a high fidelity, all autoencoder variants we tested struggled to accurately reconstruct images, capturing only limited relationships, primarily in terms of rotations and color distributions. The results revealed significant challenges in balancing reconstruction quality and compression efficiency. Future work is needed to refine the architecture, enhance the model's ability to learn structural features, and improve overall representation accuracy.

References

[1] João Carreira, Skanda Koppula, Daniel Zoran, Adrià Recasens, Catalin Ionescu, Olivier Henaff, Evan Shelhamer, Relja Arandjelović, Matthew M. Botvinick, Oriol Vinyals, Karen Simonyan, Andrew Zisserman, and Andrew Jaegle. Hierarchical perceiver. *ArXiv*, abs/2202.10890, 2022. 4

[2] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 3

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

[4] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 2

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019. 2, 3

[6] Yuehao Wang, Chaoyi Wang, Bingchen Gong, and Tianfan Xue. Bilateral guided radiance field processing. *ArXiv*, abs/2406.00448, 2024. 3

[7] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *ArXiv*, abs/2409.06765, 2024. 2