Single View to Radiance Field with Adversarial Loss

Anna Revinskaya (annare)

Abstract

In this paper, we explore generation of 3D shape representations based on a single view input. To achieve this goal, we combine per-scene single-view NeRF [6] optimization with adversarial signals. Furthermore, we compare our per-scene model results with a multi-scene baseline and find that the per-scene training using our setup fails to improve image quality.

1. Introduction

Having the ability to reconstruct a 3D object from a single image could have a wealth of applications, for example, in artistic fields. Digital 3D object modeling is a time-consuming task that requires learning special techniques and software applications. A tool that can map a quick 2D drawing to a plausible 3D shape could greatly speed up the process and make it more accessible. See our project goal in Fig. 1. We aim to take one small step towards this goal by studying a way to map a single view image to a 3D scene.

Additionally, single view 3D reconstruction is a challenging problem to solve because it is under-specified. We have ground truth only for one input view and we have no knowledge of the appearance of the object when viewed from other directions. While there are ways to estimate depth using vanishing point analysis, it is generally only applied to simpler rectangular shapes and would not include color information. Therefore, most 3D scene reconstruction approaches are based on 2 or more images. For example, NeRF [6] is trained on many views of a scene to generate estimates for novel views. In this project, we want to study if we can train a model based on a single view and use a GAN-based approach where a discriminator would encourage the model to generate views that look "real" enough.

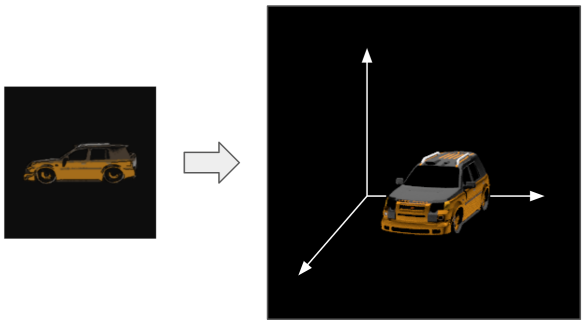


Figure 1: The goal of this project is to map a single image to a 3D scene of an object that matches this single view.

2. Prior work

2.1. 3D reconstruction

Traditionally, methods have largely focused on explicit 3D geometry representation, such as triangle meshes [8], point clouds [1] and voxel grids [13]. However these methods could be space intensive when storing data for entire spatial resolution and restrictive when it comes to the representations they can model.

Recently, implicit representations in learning-based 3D reconstruction emerged as a promising way to address these issues. Scene Representation Network (SRN) [12] and the Neural Radiance Fields (NeRF) [6] are two such examples. Implicit scene representation encodes geometries using multilayer perceptron (MLP), which in turn can model 3D scene continuously, unrestricted in topology, and can make higher spatial resolution and representation fidelity a feasibility.

2.2. Single view reconstruction

Multiple papers looked at learning a 3D representation based on a few or even a single view image. A common theme in these approaches is to train a model on other scenes as well so that it can infer how a novel scene would look like from other viewpoints based on the scenes it has seen in the past. For example, Sajjadi Et Al [10] trains a model across multiple scenes and takes as input a set of images of the same scene encoding them into a set representation. Novel views are generated using a transformer approach by querying this scene representation using ray-based keys. At the same time, "pixelNeRF: Neural Radiance Fields from One or Few Images" [15] builds on the prior NeRF work and proposes to condition it on image in-

puts, so that a single network can learn a representation for multiple scenes.

In previous class, CS236, I combined the pixelNeRF conditional approach with Generative Radiance Field field work by Schwartz Et Al [11] to train a GAN conditioned on a single view image. The view was encoded into a vector and passed to NeRF along with a position along a ray. The model used NeRF [6] as the generator and optimizes a single radiance field and volume rendering model for all scenes. This approach produced good results in some cases, but failed to match the shape when the shape is rare in the training datasets.

For this project, we want to see if adding a per-scene optimization step is beneficial at getting finer results at the expense of higher latency. A lot of prior works, including the original NeRF [6] paper focus on training a separate model for each scene suggesting that per-scene optimization could help with producing more detailed renderings. Another paper, "CodeNeRF: Disentangled Neural Radiance Fields for Object Categories" [5] uses train time optimization but optimizes camera viewpoint, and shape and appearance codes for an input view at test time and uses these values as additional inputs to the trained model. Finally, recent advancement in volume rendering optimization times suggest that test-time optimization could become a viable step for real-world tasks. For example, [7] uses a clever hash map representation and [14] uses a sparse voxel grid to get substantial speed ups.

2.3. Adversarial networks

Finally, we build on the prior Generative Adversarial Network [4] work to model sampling from the distribution of object views at different angles.

2.4. Problem statement

We start by defining the term "canonical view". Canonical view is the scene view that we have ground truth for.

Mathematically, we want to find the optimal function $F_x(p)$ that takes a camera pose p and outputs a 3D view of a scene with an object that matches the canonical view x as viewed from the camera with pose p.

2.4.1 Data

There are multiple types of information we can use to optimize the function F_x:

1. The test-time input canonical view x and some desired output pose p.
2. Training dataset of ($x_i^{(train)}$, $v_i^{(train)}$, $p_i^{(train)}$) examples, where $x_i^{(train)}$ is a canonical view, $v_i^{(train)}$ is some 3D view of the scene, and $p_i^{(train)}$ is the camera pose that corresponds to the view $v_i^{(train)}$.

2.4.2 Expected results

We plan to compare generated images visually with images generated in our prior project in CS 236 class.

We originally planned to compute FID and KID scores as well, but these scores require a dataset of generated images. However, in our case, we use per-scene optimization and did not get enough time to create enough images for the dataset.

2.5. Technical approach

We train a model that optimizes a single scene. We want to optimize this scene based on the single canonical view we get as an input. This means we don't have any ground truth for other views. Instead, we propose to use GAN[4]-based approach. Specifically, we sample views from a NeRF [6] model and then pass them to a discriminator and use the output to compute the adversarial loss. See the overall architecture diagram in Fig. 2.

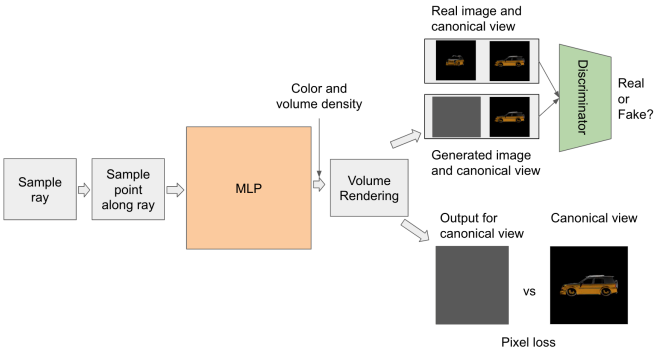


Figure 2: Proposed architecture for a 3D reconstructions model given a single view image. We start by sampling rays and points along the rays and passing these values to NeRF. If the rays correspond to the canonical view (the view provided as input), then we compute pixel loss of the generated view image compared to the canonical view image. Otherwise, we optimize the adversarial loss.

2.6. Generator

We start by sampling a camera pose, constructing the camera-to-world matrix and sampling rays for this camera pose. The rays are then passed to the NeRF implementation that samples points along the rays, passes them to an MLP model that outputs RGB colors and volume densities and then runs differentiable volume rendering to create the final predicted image.

If the sampled camera pose matches the canonical view, then we compute the pixel square loss:

$$L_P = \mathbb{E}_s ||G(p, r, s) - v_x||_2^2$$

Here, $G(p, r, s_i)$ is the view generated by NeRF based on the pose p and rays r with sample positions s along the rays. v_x is the canonical view.

For all other views, we concatenate the generated image and the canonical view image and pass them to our conditional discriminator. We use the adversarial generator loss as defined below:

$$L_G = -\mathbb{E}_{p_i, r_i, s_i}[\log D(\hat{v}_x, G(p_i, r_i, s_i))]$$

Here, p_i is the sampled camera pose r_i are the sample rays and s_i are the sample positions along these rays.

2.6.1 Pose sampling

We start by forming the camera matrix. Our images were rendered with a pinhole camera with image plane located at 0.01 from the camera. Therefore, our camera matrix is as follows:

$$K = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Then, we sample rotation angles θ around the y-axis and construct rotation matrices as follows:

$$R = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2}$$

Finally, we compute $T_{c2w} = RK$ and $o_w = Ro_c$ to get the camera-to-world transformation T_{c2w} that converts direction vectors to the world coordinate system and the camera origin in the world coordinates o_w.

2.7. Discriminator

Our discriminator takes as input either an image generated by our NeRF model and volumetric rendering or a "real" image sampled from our training dataset of various car views. Note that the discriminator is trained on the unposed views since we do not provide it with any pose information.

Furthermore, the discriminator is conditioned on the canonical view image to encourage generated representations to be consistent with the shape and appearance suggested by the only ground truth view we have. The loss function follows the traditional discriminator loss definition:

$$\begin{aligned} L_D = &-\mathbb{E}_{v_j}[\log D(v_x, v_j)] \\ &-\mathbb{E}_{p_i, r_i, s_i}[\log(1 - D(v_x, G(p_i, r_i, s_i)))] \end{aligned} \tag{3}$$

where v_j is a real view image from our training dataset.

The discriminator is implemented using sequences of 4x4 convolutions with stride 2, LeakyReLU with $\alpha = 0.2$ and batch normalization layers. The final layer is fully-connected.

I also tried to use consecutive convolution, ReLU and average pooling layers, but this produced worse results. The training was very unstable where discriminator loss would drop very quickly and generator loss would keep rising.

2.8. Symmetry assumption

We assume the cars are symmetric and use our ground truth image for the original canonical view (at angle 0^0 around y-axis) as well as its mirror reflection as the ground truth for the opposite view (at angle 180^0 around y-axis). This assumption provides additional data point to NeRF to simplify learning.

2.9. Experiments

2.9.1 Implementation

We reuse the JaxNeRF implementation [3] for NeRF and volumetric rendering. Additionally, we implement the discriminator, ray generation, dataset processing and the training and eval flow.

2.9.2 Training

We distribute the training across 4 Nvidia k80 GPUs and use batches of 16 32x32 images.

2.9.3 Dataset

We use the ShapeNet [2] dataset because it provides a large collection of simple 3D shapes. To reduce complexity and training time, we only focus on the car models.

For each 3D car model in our dataset, we only consider camera positions in the xz-plane and render views with camera positioned at random angles around the y-axis (see Fig. 4). We setup each scene by adding two spotlights (see Fig. 3) and then render each view using pbrt [9]. We render with the camera located at a radius 3 from the origin with near plane at 0.01 and the field-of-view of 22 degrees. We then use these parameters to construct corresponding camera-to-world matrices.

We set aside some input canonical views to use for the per-scene optimization and leave the rest for the main training dataset.

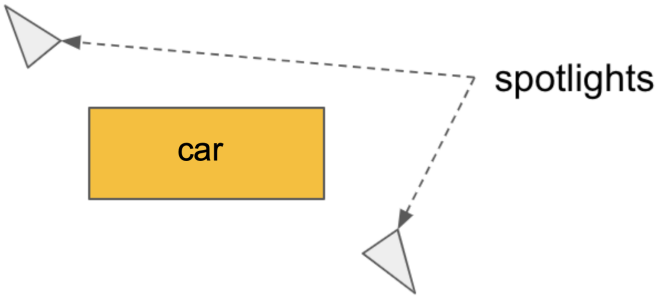
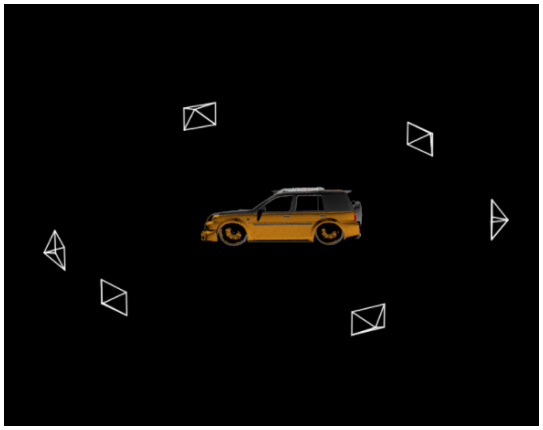


Figure 3: Spotlight placement in the xz-plane. Along the y-axis, the spotlights are placed higher than the car. The spotlights are oriented to point towards the origin.



(a) Sample camera locations.

(b) Sampled rendered ground truth views.

Figure 4: We sample camera positions at different angles around the y-axis to create real samples for our GAN.

2.9.4 Experiment: Cropping images

The large black background area makes it easy to the model to achieve very low pixel loss values just by setting entire image to black. To reduce the likelihood of this happening, we crop images (see Fig. 6). We start with a 256x256 image, then remove a 30 pixels on each side and finally resize

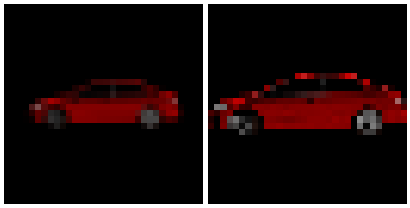


Figure 5: Ground truth original input image on the left and a cropped image on the right.

to 32x32 before feeding it to the model. Note that cropping the image has an impact on the FOV angle. The new FOV must be smaller so that a smaller part of the image plane is within the visual frustum.

If the original image width is w and the new width is αw, then the reduced field of view v_α becomes:

$$v_\alpha = 2\arctan(\alpha \tan(\frac{v}{2})) \tag{4}$$

where v is the original field of view.

2.9.5 Experiment: Weighted loss

We also tried using a weighted square pixel loss as an alternative to cropping. In this case, we give a lower weight to the square loss for pixels that are black in the ground truth image.

Weighted loss:

$$L_w = \frac{1}{N}\sum_i^N w_i(x_i - \hat{x}_i)^2 \tag{5}$$

where

$$w_i = \frac{1 + \beta|x_i \neq 0|}{\sum_{j=1}^N (1 + \beta|x_j \neq 0|)} \tag{6}$$

$\beta > 0$ is some constant that provides extra weight for the non-black pixels. Here, x_i is the ground truth image pixel value and $\hat{x}_j$ is the model estimate.

However, we observed similar image quality at the same stages of training when using the regular L2 loss and its weighted variant. The generated images are almost identical and differ very slightly. Therefore, we just use L2 loss for further results reported in this paper.

2.10. Baseline

Our baseline is our CS236 project which used a GAN with a NeRF generator conditioned on the input sketch. See sample images in Fig. 7.

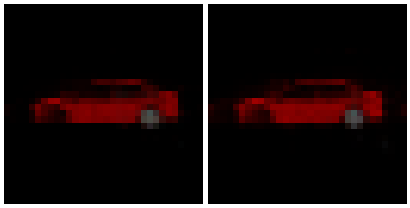


Figure 6: Generated canonical view image after 2000 iterations with L2 loss on the left and weighted L2 loss on the right with $\beta = 9$. The images look almost identical.

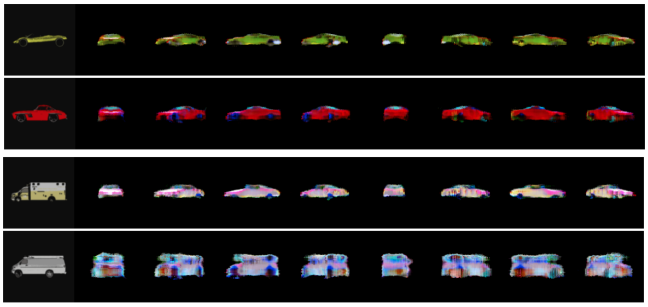


Figure 7: Baseline images.

2.11. Results

First, we train the NeRF model only using the photometric loss for the canonical view and the opposite view. The model is able to replicate the ground truth with high accuracy (see the top image in Fig. 8). However, other views lack detail.

Then, we train the NeRF model along with our discriminator using both the photometric loss when we have ground truth and the adversarial loss for all other views. Overall, the generated images for all our sample poses look rough and don't resemble cars (see the output color images at the bottom of Fig. 8). Specifically, these results are worse then our baseline results in Fig. 7. However, we observe two positive signals here: 1. The model is able to learn to match the color of the canonical view; 2. the 90^0 and 270^0 views look noticeably narrow compared to other views. The latter is in line with our intuition that a car looks smaller from the front and the back compared to the sides. This observation also demonstrates that the adversarial loss helps the model learn this relationship when we compare to the top image in Fig. 8 that only uses pixel loss and has wide outputs for any viewpoint.

Note that we report results for a single car view. Due to slow training time and since we need to train a separate model for each view, we didn't get time to get results for other inputs.

Further note that we didn't compute quantitative metrics to compare to the numeric baseline results because our generated images are visibly worse when compared to qualitative baseline results.

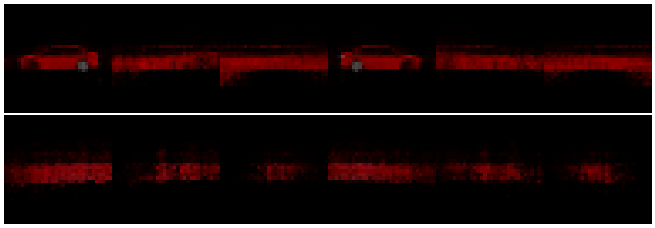


Figure 8: Top: Generated images for the model trained without adversarial loss (only pixel loss for canonical views) after 2000 steps. Bottom: generated images for the model trained both with canonical view pixel loss and the adversarial loss after 400 steps. Left-to-right camera angles around y-axis: 0^0, 45^0, 90^0, 180^0, 225^0, 270^0

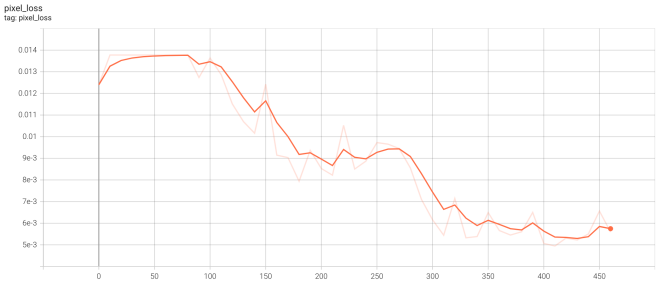


Figure 9: Pixel loss that compares the generated image for the canonical pose with the canonical view ground truth.

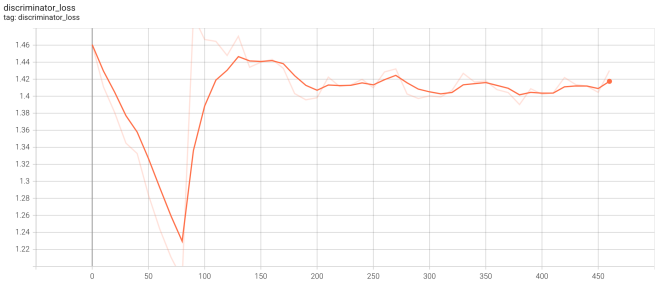


Figure 10: Discriminator adversarial loss.

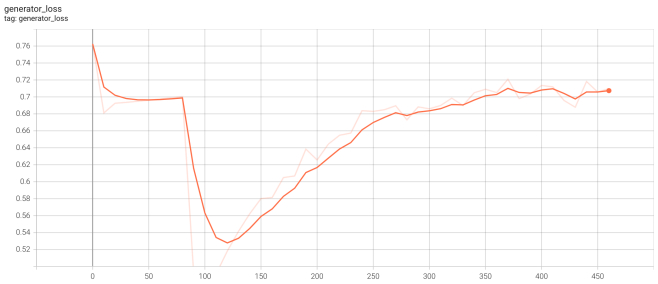


Figure 11: Generator adversarial loss.

3. Conclusions

We set out to compare two NeRF and GAN-based approaches for novel view generation based on a single view: a per-scene trained one (this paper) and a multi-scene approach (our baseline). We wanted to see if the per-scene traing can help add more detail compared to our baseline that attempts to fit entire scene family.

Surprisingly, we found that the per-scene training actually under-performed the baseline. Our proposed method learned to match the color but failed to generate images that resemble the intended object shapes. However, it is possible that we just didn't train our model long enough and that further tuning could improve performance.

4. Future work

Most importantly we want to leave the current model to train for longer and check results for other cars as well. On a related note, reducing the training latency could make experimentation and tuning a lot easier and this approach - more applicable for real-world applications. Therefore, we could look at optimization techniques such as the ones proposed in [7], [14] to speed it up.

Additionally, there are multiple further experiments we could try:

1. Increase the dataset size. Currently, the real images passed to the discriminator are sampled from the dataset that has 1902 images total across 317 car models.

2. Pass an input view rotated at such angle so that both the front and one of the sides are visible. This view would provide more information to the model compared to the current single-side view.

3. Include pose in the discriminator input.

4. Use the approach described in this work as a way to fine-tune a NeRF representation instead of training from scratch.

5. Code

The code is available at: `https://github.com/annarev/cs231a_project`.

6. Acknowledgments

We would like to thank the CS 231A course staff including instructors and TAs for their instruction and guidance throughout the course!

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds, 2018.

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[3] B. Deng, J. T. Barron, and P. P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[5] W. Jang and L. Agapito. Codenerf: Disentangled neural radiance fields for object categories. *CoRR*, abs/2109.01750, 2021.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[7] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding, 2022.

[8] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia. Deep mesh reconstruction from single rgb images via topology modification networks, 2019.

[9] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016.

[10] M. S. M. Sajjadi, H. Meyer, E. Pot, U. Bergmann, K. Greff, N. Radwan, S. Vora, M. Lucic, D. Duckworth, A. Dosovitskiy, J. Uszkoreit, T. A. Funkhouser, and A. Tagliasacchi. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. *CoRR*, abs/2111.13152, 2021.

[11] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[12] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020.

[13] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[14] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. *CoRR*, abs/2112.05131, 2021.

[15] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. *CoRR*, abs/2012.02190, 2020.