Improving Neural Radiance Fields for Efficient Scene Reconstruction

Jingyuan Chen *

*Course Number: CIS700 Project Report

Abstract

Artificial intelligence has become a significant development in numerous fields, where it has demonstrated relatively high performance. One of the most promising technologies is Neural Radiance Fields, which have been especially useful in the process of reconstructing detailed 3D scenes from unstructured 2D images. However, the approach is highly computationally demanding and requires extensive learning efforts. This paper uses transfer learning and meta-learning to improve the efficiency of NeRF. Namely, the NeRF model is trained on objects of a similar type and based on reconstruction experience to reduce training time and produce better results from a small dataset. The findings show a decrease in training time and an increase in the NeRF model's adaptability to new data, which is promising for practical use.

Keywords: Neural Radiance Fields, 3D Reconstruction, Transfer Learning, Meta-Learning, Artificial Intelligence

1 Introduction

Today, with the popularity of artificial intelligence, artificial intelligence has shown powerful performance in more and more fields. In this context, the use of AI to assist 3D modeling is a novel technology. Among them, neural radiation field technology (NeRF) has shown very good results. It uses AI models to 3D reconstruct image data sets. This method is very effective. It greatly solves the manpower problems and memory requirements of modeling. According to the NeRF paper, it uses deep learning to build a continuous 3D scene representation, learning the volumetric density and color information of the scene through an optimization process. This method can reconstruct realistic 3D scenes from a set of sparse and unstructured 2D images. and composite new images from unseen angles. However, as the NeRF paper shows, training the model takes more than 12 hours. [1]. This means that if we want to build a model by providing a new input dataset, it will take a lot of time to train the model. In addition, nerf requires a large amount of data sets to support training. However, in real life, we do not always get a large amount of image data as support, and sometimes only some images can be used.

Our paper explores how to reduce the training time of Neural Radiation Field technology (NeRF). We hypothesized that the NeRF model would train faster if it was trained on similar objects. In other words, the model will adapt to new data sets more quickly by having seen similar objects before. The goal of this article is to enable Nerf to quickly perform 3D reconstructions of similar objects based on previously seen datasets. This can both speed up 3D reconstruction and reconstruct small data sets.
The main contributions of our paper are:

- Our model reduces the training time of NeRF techniques by using transfer learning and meta-learning, as well as enhances the reconstruction of small data sets.

- Our model classifies different types of objects and uses different models for transfer learning to ensure no interference.
- For objects that do not have similar classes, we use a meta-learning model to speed up the efficiency of model reasoning through previous reconstruction experience.

Compared with previous work, our innovation is that we propose to classify different types of objects and use different models for transfer learning or meta-learning. In previous work, NeRF technology was retrained from scratch every time for a new dataset, which would result in different types of objects being more difficult to classify. Since NeRF technology generates the corresponding color and volume density by observing the angle of the picture, if we do not classify different types of objects and directly use transfer learning, then different types of objects are likely to damage the quality of our model. But if we classify them, in general, objects of the same type will be similar, such as chairs, flowers, etc. Their shapes are usually similar, and transfer learning can adapt well to this situation. In addition, when we find that we cannot find similar categories, we can use a meta-learning model for training, so that our model can be trained to construct new objects faster through the experience of previously building objects and reduce the demand for datasets.
The layout of the rest of the paper is as follows:
In Section 2, we mainly introduce our related work, Section 3 introduces our method and implementation principle, Section 4 introduces our experimental design and results, Section 5 is a discussion of our results, Section 6 discusses the limitations of our work, Section 7 discusses our future work, Section 8 is our conclusion, and in the Appendix we introduce the implementation of our code.

2 Related Work

In this section, we will introduce our related work. Section 2.1 is about the NeRF technology, Section 2.2 is about the transfer learning method, Section 2.3 is about the meta-learning method,

2.1 NERF

The Prior art of this paper is NeRF technology. NeRF uses deep learning to build a continuous 3D scene representation, learning the volumetric density and color information of the scene through an optimization process. This method can reconstruct realistic 3D scenes from a set of sparse and unstructured 2D images. and synthesize new images from unseen perspectives. [1]

2.2 Transfer learning

In machine learning, transfer learning is a method where a model acquired for a task uses that solution as the place to start for a model on a second task.

Transfer learning is a trendy strategy in deep learning with the use of pre-trained models as the starting point because it can significantly enhance getting to know the performance and reduce the computations. [3]

2.3 Meta learning

We use Model-Agnostic Meta-Learning as the meta-learning method in our model. This method is mainly to better initialize the parameters of the model so that it can adapt faster to new tasks. We adjust its initial parameters by training the model in different scenarios.

3 Approach

In this section, we will introduce our methodology. Our approach is as follows. We have a trained classification model M. Under this model, for n categories, we have n+1 sub-models, where all of these are Nerf-based models.$m_1, m_2, ..., m_n$ is the Nerf model trained using the k-category dataset. Then m_{n+1} is the model we trained using meta-learning based on 1 to n categories of data. When we receive a new data set, we first classify it and then select the appropriate model based on the classification results.

3.1 Classifier network

The classifier network is for classifier image to each submodel. In Nerf technology, we consider all the image data as multiple rays from different views. In that case, we need to train every time we get a new image dataset. However, that process will take a long time. To accelerate that processing, we want to make a pre-trained model, which can fit new datasets quickly. We used ResNet as our classification model, and we used softmax as the final activation function to convert it into a probability distribution. When we receive a new input dataset, we pass the pictures into our classification model one by one, and then take the average of the prediction results for each picture, so that we can get the most similar classification. Generally, there are two results: the result shows that our new dataset is similar to a previous model, and the result shows that our new dataset does not match the model. When our new dataset is similar to a previous model, we use transfer learning technology (section 3.2). If our new dataset does not have a similar model, we use meta learning technology. (section 3.3)

3.2 Transfer learning

Our approach is to perform transfer learning. We assume that if our model has been pre-trained on a similar object dataset, it will reach the same performance faster than a model without pre-training.
As mentioned in section 3.1, we find the most similar model through the classification model, and then we freeze the first k layers of our model, and then use

our new dataset for training on this basis. In this case, our new dataset will converge very quickly and reach a certain accuracy. This method is that our model can handle small-scale similar datasets well, and the reasoning efficiency and accuracy of similar datasets can be significantly improved

3.3 Meta learning

The meta-learning model contains potential learning skills, which can make our new model have faster training speed and better generalization performance. We use Model-Agnostic Meta-Learning as the meta-learning method in our model. We use the previous data set to train using meta-learning. In meta-learning, the meta-learning model can learn how other models handle these tasks very well. Therefore, by letting our meta-learning model observe how other models complete the task of reconstructing scenes with nerf, we can get some improvement. When facing completely unseen picture types, our meta-learning model can complete training faster and better than ordinary models.

4 Experiment Design:

4.1 Dataset

We will use the dataset from the NeRF paper and the TUM dataset [2]. The TUM dataset gives some images of objects from different viewpoints, which is useful for NeRF techniques. The metric we will use is the PSNR used in the NeRF paper. We Calculated by comparing the original image with the processed (such as compressed or reconstructed) image. PSNR is calculated based on mean square error (MSE). First, the MSE between the original image and the processed image is calculated. High PSNR value: generally indicates lower error and better image quality. We will also use iterations as a metric to compare our model with previous work. The model from the previous NeRF paper is the baseline model in our experiment.

4.2 Result by transfer learning

We use the dataset in NeRF paper to train our classification model, we pass a leaves dataset from TUM dataset and it has been correctly assigned into the leave catalog. Then we select the transfer learning method and the result is shown in fig1

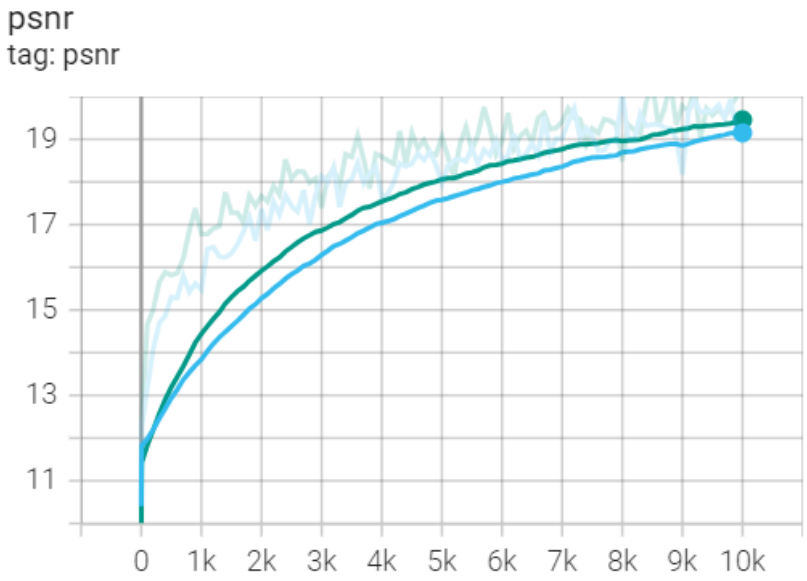

Fig. 1: Transfer Learning Result

The x-axis is the step of iteration, the y-axis is the psnr value. The blue line is the baseline, and the green line is our method result.

4.3 Result by meta learning

We use the dataset in NeRF paper to train our classification model, but in this time, we didn't include leave dataset in it, we pass a leaves dataset from NeRF paper dataset and it cannot be classified. Then we select the meta learning method and the result is shown in fig2

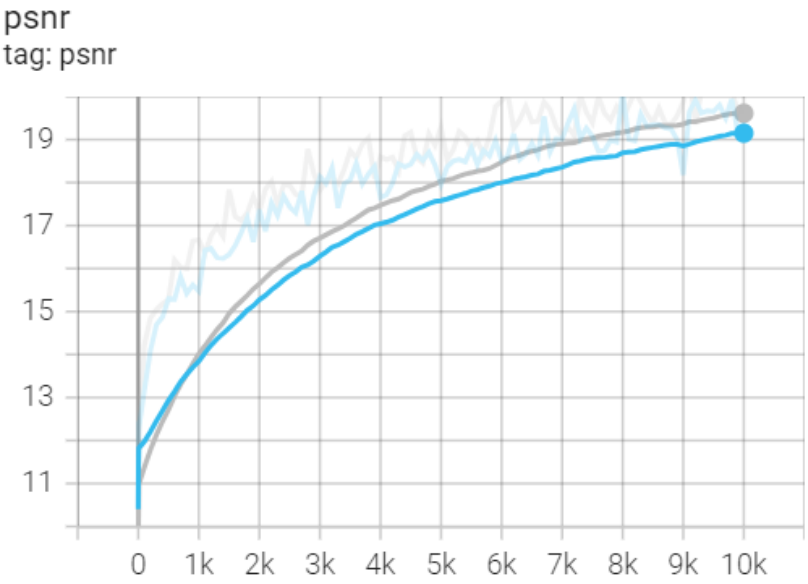

Fig. 2: Meta Learning Result

The x-axis is the step of iteration, the y-axis is the psnr value. The blue line is the baseline, and the gray line is our method result.

5 Discussion

According to our results, we found Our model has a very good representation of both the data we have seen and the data we have not seen. We also discovered that our transition learning method works better on certain data sets, especially those with very different backgrounds. This is because our model is learning to move forward on a picture with a large difference in background, and because the difference is large, we have frozen the way we extracted the features several times before and were unable to extract the features that are valid for our new data set, so it appears poorly. The effectiveness of the proposed modifications to the NeRF technology was further confirmed through the results acquired in our experiments. On the one hand, as already mentioned, transfer learning as our first modification turned out to be highly beneficial when training on a new data source and target distributions when the background was heavily modified. In other words, it allowed the model to effectively reuse the features learned on a similar dataset and achieve a similarly good performance without retraining from scratch. However, extension of the backgrounds to more complex and distinctive to some extent settings shows that the method benefits from more training stable only on similar datasets; i.e. simple influential datasets. In

other words, transfer learning allows for versatile use of the network capabilities on similar datasets, but does not allow generalizing over dramatically dissimilar datasets. On the other hand, the results acquired using meta-learning as our second modification enhanced generalization capabilities, as expected. Therefore, general more training stable over more complex datasets allows more efficient and faster adaptation to new datasets. In summary, our findings show that the proposed learning strategies may significantly reduce the training costs of using NeRFs and make them more feasible.

6 Limitations:

In our paper, there are several limitations First, as pointed out in the chair example above, however, not all objects of the same type can have similar shapes, for example, some chairs may be significantly different from ordinary chairs.

In addition, due to the limited conditions of computing equipment, we did not try a large number of iterations. We chose 10k iterations as the result display, which is a bit small in NeRF technology.

7 Future Work:

In future work, We will conduct experiments on more datasets and use a larger number of iterations to compare our model with the baseline model. In addition, we found that our model was affected in transfer learning. We considered that using a mask to block interfering light is a possible solution. Our future work will focus on studying this to make our model work for background images with large differences. Last but not least, we also consider how to combine our meta learning model and our transfer learning model. We will study how to combine the advantages of the two methods to further improve the reconstruction efficiency.

8 Conclusions:

In conclusion, our model will make the Nerf model training on new datasets more efficient by using transfer learning and meta-learning technology. Our model will have to achieve better performance on the quality of 3D restructure in the same amount of iteration. That imply our model can get the same quality with a lower iteration number than Nerf paper.

References

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

[2] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)* (Oct. 2012).

[3] Torrey, L., and Shavlik, J. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, E. Olivas, J. Guerrero, M. Martinez-Sober, J. Magdalena-Benedito, and A. Serrano López, Eds. IGI Global, 2010, pp. 242–264.

9 Appendix

In this appendix, We include a short explanation of our code part. Compared with the code part of the previous NeRF paper, our main contributions are

9.1 train_nerf.py

In this code, This code is similar to run_nerf.py in nerf. We added more parameters to run_nerf.py to adjust N_iters, whether to perform transfer learning, etc.

9.2 run_nerf_helpers.py

In this code, We have implemented the option of using transfer learning when initializing the nerf model. That is, whether the first few layers need to be frozen.

9.3 image_classifier.py

In this code file, we use ResNet50. We add a softmax activation function after to convert the result into probability. We use ImageDataGenerator to generate more image data for training to ensure that we have enough data to complete the classification model fitting. After training, we store the model. When we need to use it, we load the previous model to make predictions on the data set.

9.4 meta_learning.py

In this code file, we wrote a meta-learning model. We used the load dataset method in the original paper to obtain images and poses as scene data, and then used meta-learning for training. We randomly selected some views and poses from the scene data in each iteration, and then trained them. We used the meta-learning model to observe the training process and adjust the initial parameters of the meta-learning model. Then save the model after training.