# Replication of Supervised Learning of the Next-Best-View for 3D Object Reconstruction

**Alexey Boyko** [1] **Dmitrii Smirnov** [1]

## Abstract

The Next Best View planning for 3D reconstruction is a long-standing problem that has been researched by computer vision communities for decades. Since the second half of the last decade, more and more new methods based on Deep Learning (DL) have been proposed to solve this problem. In this paper we review the supervised learning scheme for NBV problem solution proposed by (Miguel Mendoza & Reta, 2019). The original Jupyter Notebook research code was refactored into a Python package with command-line interface. We replicated the base results of the paper, as well as performed additional experiments on the architecture of the deep neural network.

## 1. Introduction

The task of three-dimensional (3D) reconstruction means the understanding of 3D shape and structure of objects from single or multiple of 2D images, typically involving depth channel. This problem was mainly studied in computer vision communities (in an off-line setting) and in robotics communities (in an on-line setting, also known as SLAM). Results of those works were later used in other fields, such as computer graphics, animation, virtual reality, medical surgery, etc.

The process of 3D object reconstruction in its simplified form can be described as follows. Initially, a sensor (such as an RGB-D camera, LiDAR or another depth sensor) is used to obtain a depth image, that can be transformed into a set of points. Since points accumulate very quickly, some kind of refinement (or Fusion, such as TSDF/KinectFusion or its later versions) procedure is used to keep the memory requirements finite.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Dmitrii Smirnov <Dmitrii.Smirnov@skoltech.ru>.

*Final Projects of the Machine Learning 2020 Course*, Skoltech, Moscow, Russian Federation, 2020.

This procedure is repeated from other viewpoints to collect more positional data and thereby more accurately reconstruct the shape of the object. Usually, in synthesize a sufficiently detailed 3D model of a real-world object, it is necessary to get a lot of depth images of object from different positions.

The problem of finding the Next Best View (NBV) is to determine the best next position for a camera in such a manner as to maximize information about unknown parts of the 3D object.

Such algorithms are needed to speed up the scanning process. Nowadays, most state-of-the-art NBV planning techniques use a deep learning (DL) approach. Such methods address the NBV planning paradigm as a classification problem, where the features and the classifiers are learned from the object data. While DL approaches had a big success in tasks of pattern recognition, it's not yet clear what architectures will be best for this task. Although fully-connected layers can imitate any complex function, due to high-dimensional nature of 3D data, such approach is not computationally feasible, and 3D convolutional networks (3D-CNN) are typically used. CNNs have much fewer connections and parameters than standard neural networks with direct communication with layers of the same size, so they are easier to train (A. Krizhevsky & Hinton, 2012). But even in this case, only low-dimensional volumetric data is feasible to (i.e $32^3$ voxels).

In this report we propose a modification of the code of NBV-Net network presented by (Miguel Mendoza & Reta, 2019). We adopted their approach in general but heavily refactored all codes, allowing for scripted automated experimenting, and lated conducted additional experiments for testing other architectures that are not present in the paper.

In summary, the main contributions of this report are as follows:

- We replicate the results presented in the original paper.
- We heavily re-factored the code from spaghetti scientific code to be closer to an industry-standard ML Python package
- We improved the original model architecture as well

as structure of the code.

- We experimentally validate that our approach is able to effectively predict the NBV

- We introduced a different dataloading approach and obtained a 2x speedup

## 2. Related work

At this paragraph we briefly review different pipelines for NBV planning. Both classical techniques and the state-of-the-art DL approaches are covered.

### 2.1. Classical methods

The Next Best View planning has been an active research topic for over 35 years. The term itself was proposed by Connolly in his pioneering work on the creation of a view planning algorithm (Connolly, 1985). In that work the author presented a partial octree model to describe visual objects. However, this approach only allowed to obtain the general shape of the object without giving any information about how to maximize the amount of new data obtained by the next sensor position. The subsequent works have gained more achievements on the NBV. Roy et al. (S. D. Roy & Banerjee, 2001) proposed a scheme for the recognition and pose estimation of a large isolated 3-D object via search tree nodes approach. Blaer et al. introduced a voxel-based occupancy procedure to plan the next nest view (Paul S. Blaer, 2007).

### 2.2. State-of-the-art approaches

Following the success of CNNs on image processing tasks, several authors have suggested applying them to 3D reconstruction task and NBV prediction problem. For example VoxNet (Maturana & Scherer, 2015) and ShapeNets (Zhirong Wu & Xiao., 2015) were successfully applied to volumes recognition. In turn, in (Louis Ly, 2018) it was shown how CNN could be trained for visibility-based exploration, reconstruction and surveillance. Wu et al. (Zhirong Wu & Xiao., 2015) designed a model that is able to compute descriptors for object views. To achieve this, authors trained a CNN to compute these descriptors using simple similarity and dissimilarity constraints between the descriptors. Lastly, the Vasquez-Gomez research group published a number of papers (Mendoza, 2018; Miguel Mendoza & Reta, 2019) in which they presented NBV-Net framework used to NBV learning. The NBV-net has proved to be effective in classification and regression of the next best view, therefore we used it in this work.

## 3. Algorithms and Models

In this section we detail the background of NBV-net training and testing processes.

### 3.1. Dataset

In order to address the problem of NBV prediction it was posed as a classification task. Each class corresponds to a certain predefined camera position, from which we can take an additional depth image of our image. Since camera positions are relatively far from each other (14 of equidistant points on the sphere around the object), the classification problem setting may be applied. NBV classes correspond to sensor poses evenly distributed on the superior half of a sphere (14 poses in total).

The dataset contains examples of partial models $M_i$ and their corresponding NBV classes. Each partial model $M_i$ represents volumetric probabilistic occupancy grids of $32^3$ voxels, where each voxel has a probability value between 0 and 1. In the authors' setting if $P > 0.5$ the voxel is considered occupied, if $P < 0.5$ the voxel is considered empty and if $P = 0.5$, the voxel are considered to be an unknown area. NBV in a voxel setting attempts to minimize the number of these unknown voxels.

Figure 1 shows an example of one randomly selected object contained in the dataset. The dataset generation process is essentially a fusion process of point clouds into an occupancy grid, the detailed description of the fusion algorithm is described in the original paper (Miguel Mendoza & Reta, 2019). During the process, the selected object is reconstructed several times using different initial sensor positions and the best next camera position is found by an exhaustive search. The total size of the dataset is approximately 3.5 Gb.

The dataset can be downloaded from (Miguel Mendoza & Taud, 2019a)

### 3.2. NBV prediction

The NBV prediction method presented by the authors is based on the information from a partial model of object, i.e. on the knowledge obtained from previous reconstructions. Its main idea is finding a function $f(M)$ that maximize the surface of the object contained in the partial model $M$. Mathematically, the task of NBV learning is postulated as:

$$f(M) : \mathbb{R}^n \rightarrow \mathbb{R}^3 \times SO(3) \tag{1}$$

where $\mathbb{R}^n$ is the voxel grid and $n$ is the amount of voxels in the grid. A partial model $M$ is defined by a probabilistic grid, where the space is evenly divided into voxels.

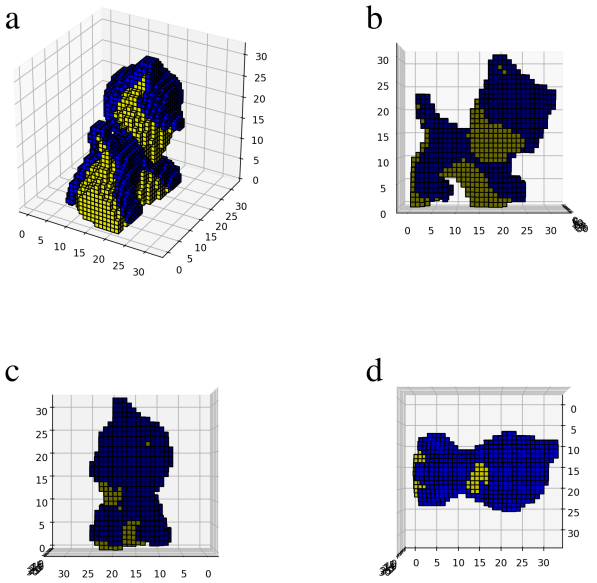


*Figure 1.* An example of an object in the dataset. (a-d) The object "puppy" is shown from four different viewpoints. Each object in dataset is defined using a voxel probability grid. Blue voxels correspond to the scanned surface, yellow voxels show an unknown volume.

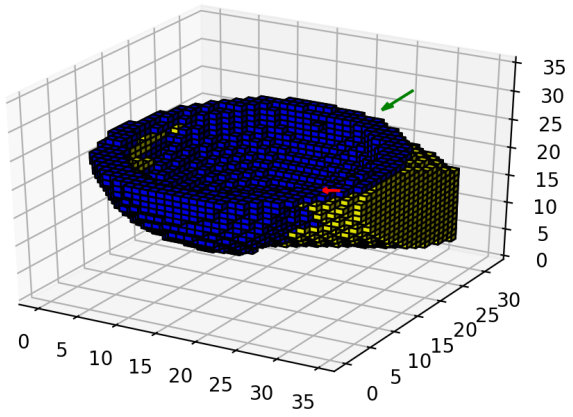

*Figure 2.* An example of the NBW predicted by a NBV-net for a randomly selected object from a dataset. The red arrow corresponds to the predicted NBW, the green arrow indicates ground-truth NBV.

### 3.3. Code refactoring

The main problem we encountered during the project work was barely comprehensible structure of the original code (Miguel Mendoza & Taud, 2019b). The github repository structure was a collection of Jupyter Notebooks that contained a large amount of anti-patterns, such as repetitive patterns, global statements, etc. After our refactoring, the code became available as a Python package. We also paid a great attention to memory consumption during the code execution. This issue may be crucial for the original NBV-Net implementation, because of the large data files and the possible restrictions on the total amount of available memory. To avoid this problem we implement the dataset fully casting to gpu device.

### 3.4. Architecture

Convolutional neural networks (CNNs) are a very broad class of architectures whose main idea is to reuse the same parts of the neural network to work with different small, local pieces of input data. A 3D CNN is very much similar to 2D CNN, except that it differs in kernel volumes.

Figure 3 shows a diagram of NBV-Net architecture proposed in the original paper. It consists of two main parts. The first one extracts 3D features and includes three 3D convolutional layers. Another one comprises five fully connected layers and outputs predicted NBV class. The NBV architecture was implemented in PyTorch.

As part of our project, we also tried to modified the original architecture of NBV-net. It was not very clear why exactly five fully-connected layers were the chosen by authors. In order to understand how the number of fully-connected (FC) layers affects the values of the loss function and accuracy, we conducted additional experiments. In our experiments we trained up models with different numbers of FC layers, ranging from 1 to 5.

### 3.5. Network training details

The dataset was spited 20%-80% between testing and training subsets. As in original paper, the Adam stochastic optimization was chosen for network training (Kingma & Adam, 2015). Adam optimizer was initialized with a learning rate of 0.001 with a batch size of 1000. The number of epochs during training were equal to 400. Model weights were initialized uniformly. The fully reproducible code is available here NBV-Net: A 3D Convolutional Neural Network for Predicting the Next-Best-View

## 4. Experiments and Results

The experiment regarding the studies of neural network architectures was conducted using the dataset proposed.

We compare the original NBV-Net architecture (NBV-Net-5FC) against architectures with 1,2,3 and 4 FC layers (NBV-Net-1FC, NBV-Net-2FC, NBV-Net-3FC, NBV-Net-4FC).

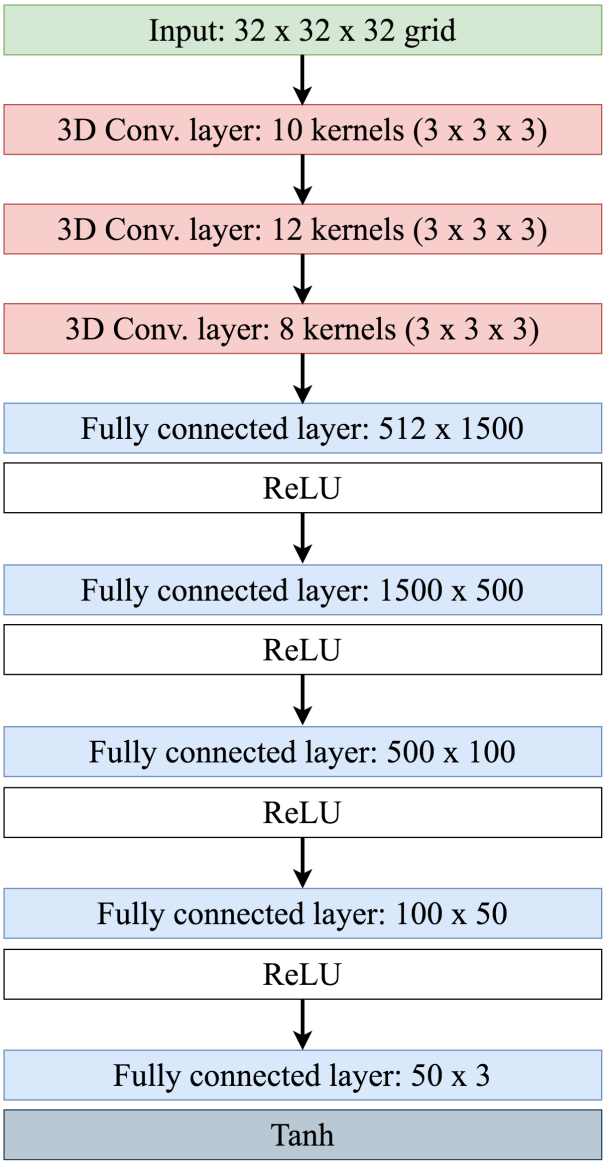


*Figure 3.* NBV-net architecture. The original version of it contains three 3D convolutional layers and five fully-connected layers. After each FC layer, except the last, the ReLu activation function is used.

Figures 4 and 5 show the training loss and accuracy for all models described. We found three of them (NBV-Net-3FC, NBV-Net-4FC, NBV-Net-5FC) to perform almost indistinguishably. NBV-Net-1FC and NBV-Net-2FC showed significantly less successful results in comparison to others. In addition, both training and testing time for all models were approximately the same: 3951 - 3991 s. Table 1 shows that in more detail.

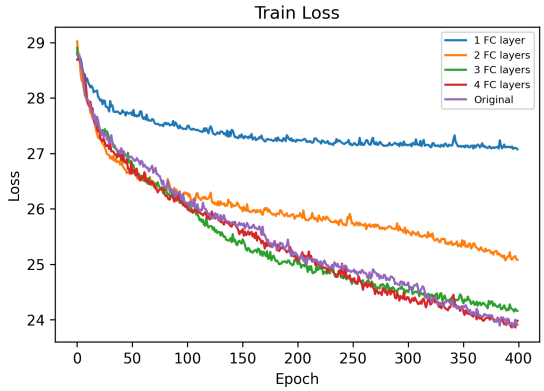


*Figure 4.* The dependence of the train loss on the epochs number for different types of neural network architectures

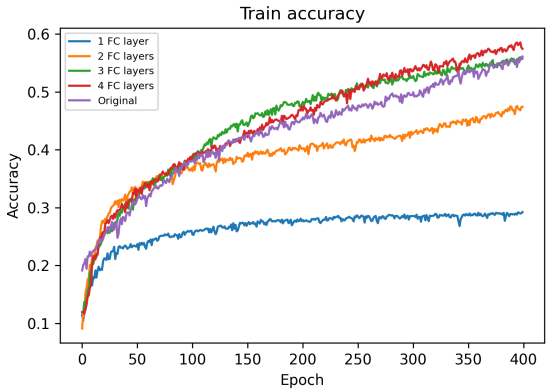


*Figure 5.* The dependence of the train loss on the epochs number for different types of neural network architectures

### 4.1. Timing

We use a GPU device in our experiments. Our slowest configuration, NBV-net training and testing without fully casting the dataset on the device takes about 30 s per epoch. Instead, the optimized configuration takes approximately 10 s per epoch.

## 5. Conclusion

In this work we have tested an algorithm that reconstruct 3D objects using a convolutional neural network. The original algorithm proposed by (Miguel Mendoza & Reta, 2019) has been significantly modified by introducing. Our experimental results demonstrate that the proposed method is able to successfully predict the NBV and has relatively high real-time performance.

*Table 1.* Time of execution during training and testing

| ARCHITECTURE | NUMBER OF FC LAYERS | TIME (S) |
|---|---|---|
| NBV-NET-1FC | 1 FC LAYER | 3977 |
| NBV-NET-2FC | 2 FC LAYERS | 3955 |
| NBV-NET-3FC | 3 FC LAYERS | 3951 |
| NBV-NET-4FC | 4 FC LAYERS | 3991 |
| ORIGINAL | 5 FC LAYERS | 3956 |

## References


A. Krizhevsky, I. S. and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *NIPS*, pp. 1097–1105, 2012.

Connolly, C. The determination of next best views. *PROCEEDINGS OF THE 1985 IEEE INT. CONF. ON ROBOTICS AND AUTOMATION*, pp. 432–435, 1985.

Kingma, D. and Adam, J. B. A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Louis Ly, Y.-H. R. T. Autonomous exploration, reconstruction, and surveillance of 3d environments aided by deep learning. *arXiv preprint arXiv:1809.06025*, 2018.

Maturana, D. and Scherer, S. Voxnet: A 3d convolutional neural network for real-time object recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015.

Mendoza, M. Nbv-net: una red neuronal convolucional 3d para predecir la siguiente mejor vista. *Tesis de Maestría, Instituto Politécnico Nacional*, 2018.

Miguel Mendoza, J. Irving Vasquez-Gomez, H. T. L. E. S. and Reta, C. Supervised learning of the next-best-view for 3d object reconstruction. *arXiv preprint arXiv:1905.05833*, 2019.

Miguel Mendoza, J. I. V.-G. and Taud, H. Nbv classification dataset. *https://www.kaggle.com/miguelmg/nbv-dataset*, 2019a.

Miguel Mendoza, J. I. V.-G. and Taud, H. Nbv-net: A 3d convolutional neural network for predicting the next-best-view. *https://github.com/irvingvasquez/nbv-net*, 2019b.

Paul S. Blaer, P. K. A. Data acquisition and view planning for 3-d modeling tasks. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 417–422, 2007.

S. D. Roy, S. C. and Banerjee, S. Recognizing large 3-d objects through next view planning using an uncalibrated camera. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol.2:276–281, 2001.

Zhirong Wu, Shuran Song, A. K. F. Y. L. Z. X. T. and Xiao., J. 3d shapenets: A deep representation for volumetric shapes. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920,, 2015.

## A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

**Dmitrii Smirnov (50% of work)**

- Reviewing literature on the on the NBV (3 papers)
- Using code to run experiments
- Adding some parts in code
- Preparing a larger part of the text of this report
- Preparing parts of the presentation

**Alexey Boyko (50% of work)**

- Reviewing the proposed paper, finding its flaws and negotiating with our project instructor how to modify the problem setting and which milestones for the code we could try to implement
- Fully re-factoring the repo from 2 notebooks into Python package
- Giving the presentation
- The majority of all coding
- Adding fixes in this report

## B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **General comment:** If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   **Students' comment:** The basic functionality of the code remained, but we did full refactoring of the code and repository. What used to be a large notebook with spaghetti code has become a python package with a cleaner structure, that could also be used with CLI (as well as in notebook setting). Many parts of the code were also fully re-written in a less redundant way.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☐ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☐ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** Here we experimented with a number of FC layers as our hyperparameter, and measured the best solution by comparing time of learning and accuracy achieved

9. The exact number of evaluation runs is included.

   ☐ Yes.
   ☐ No.
   ☐ Not applicable.

   **Students' comment:** None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    **Students' comment:** None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

**Students' comment:** None

12. Clearly defined error bars are included in the report.

    - ☐ Yes.
    - ☑ No.
    - ☐ Not applicable.

    **Students' comment:** Even to conduct a single experiment (and we did many, with different architectures) on this 3D dataset and this large network requred 1 hr on Colab's TPU or 2 hrs on our personal GPU. With the time and computational resources that ware given it was unfeasible to conduct statistical experiments

13. A description of the computing infrastructure used is included in the report.

    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    **Students' comment:** None