Report for 194.077 Applied Deep Learning

CapsVoxGAN

Franz Papst

January 22, 2020

1 Introduction

My project aimed to be a three-dimensional generative adversarial network (GAN) [2] for generating voxel models, using a three-dimensional capsule network [4].

CNNs are quite good when it comes to detecting features, but they do not take the part-to-whole-relation into account. If all features are present, but their relationship to each other is odd, a CNN would still recognise it. Figure 1 illustrates this: figure 1a shows a simple house with normal proportions, figure 1b shows how a house with skewed proportions, resembling an abstract painting. A CNN would recognise both, since all the features (roof, windows and door) are there, but in a strange relation to the whole. A capsule network would not be affected by this.

2 Fundamental Questions

2.1 What is the problem I tried to solve?

I tried two tackle two different problems:

1. Using a capsule network for three-dimensional model classification
2. Generating voxel models using a GAN

Despite the attention deep learning got in recent years, three-dimensional machine learning is still rather in its infancy. The reason for that lies in the highly increased complexity of adding one more dimension.

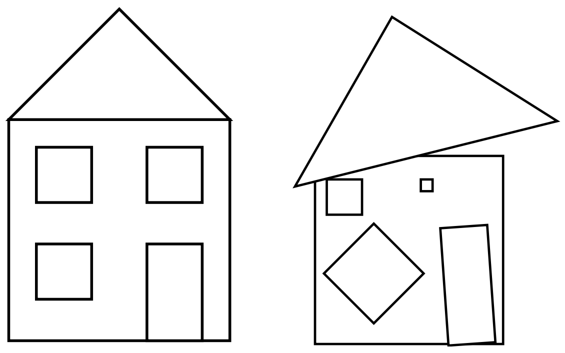
(a) Regular house. (b) Skewed house.

Figure 1: Example of odd part-to-whole relation.

2.2 Why is it a problem?

1. Magnetic resonance (MR) scans of brains are often represented in voxels [1, 3]. Developing deep learning methods, which work well on this kind of data would be highly beneficial, e.g. by giving researches tools to better analyse the large scaled data-sets produced by MR, or for the automatic detection of regions of interest by physicians.

2. While classification based on voxels can be used for very serious use-case, the ones for model generation are more light-hearted. Generating 3D models can be used in the entertainment sector for computer games or films. It would greatly cut the production costs for large and smaller studios as well as allow for new experiences, e.g. games which get generated on the fly based on the user's preferences, making it a unique experience for each player.

2.3 What is your solution?

My intended solution was to build a GAN which uses a 3D capsule network as discriminator. In my actual solution, I did not use capsule networks, since I wasn't able to get them to work in 3D.

2.4 Why is it a solution?

Even though, the capsule network is not working as intended, the GAN is and it is able to produce voxel models as shown in figure 2.

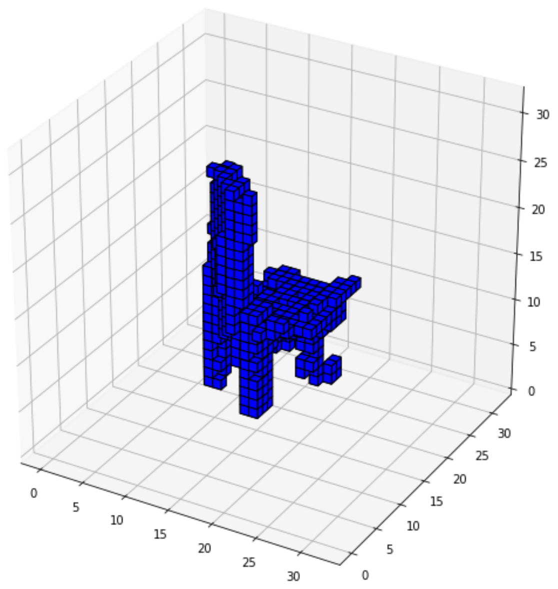

Figure 2: Voxel model generated by the GAN after 490 training epochs.

3 Main Take-Aways

Unsurprisingly the main take-away is that 3D machine learning is hard. Luckily there are datasets 3D datasets available [6], but training a GAN for 3D data is still very challenging, even when having access to a GPU-cluster. In order to keep the training time reasonable, I had to limit the training data to four classes: wardrobe, bed, chair, laptop and let the training only run for 500 epochs, which took around 50 hours on a NVIDIA Tesla K20Xm with 6GB VRAM.

The results I got are reasonably good[1], but still far away from what GANs can achieve for image data. The generated models loosely resemble the training data, but when looking at them it soon becomes evident, that some voxels are missing or misplaced. This is very common when 3D models are generated by machine learning systems [7, 5].

Another take-away is that reproducibility is very important. Out of my frustration with generally having to spend at least one day to get code found on GitHub to run (and because I was working on two different machines),

[1]Go to `https://github.com/CPUFronz/CapsVoxGAN#results` for more examples

I decided to use a Conda environment and make it as easy as possible for someone interested to download and run my code. It is not that much more effort and it makes things so much easier, not only for other people, but also for myself.

Related to the previous paragraph: Conda is a great tool to make development environments portable, but it is not suited for deployment (to be fair, I don't think that this was the intention behind it). I tried to create an AppImage[2] out of my Conda environment, with a tool that was created for doing so[3], but it didn't really work. Afterwards I tried to put my Conda environment inside a Docker container[4], but that didn't work either. In the end just using the official Docker image for PyTorch[5] and adding all the other needed packages worked best.

Speaking of deployment: deploying took actually much more time than expected. The main reason was, that creating a desktop application with a simple GUI and pack it into a AppImage, didn't work out as expected. Therefore, I had to change to a Flask[6] based web app. The programming was rather easy, but turning it into a proper Docker file took some time.

In general, I have to say that I chose this project, because it is something I probably would not touch in my regular work[7], but something I am interested in and want learn more about play around with. And indeed, I did learn a lot: from capsule networks, to 3D machine learning and deployment. It was also the first project I carried out from start to finish in PyTorch[8]

3.1 What would I do different, if I did the same project again?

Start with a simple 2D capsule network for image classification or object detection and then going for something in 3D. Right from the beginning, I knew that this project is anything but easy, but 3D machine learning is hard and capsule networks are hard, so combining them is double hard. Unlike CNNs capsule networks are not shipped in popular deep learning frameworks, let alone 3D capsule networks.

[2]`https://appimage.org/`

[3]`https://github.com/linuxdeploy/linuxdeploy-plugin-conda`

[4]There are official Docker images for Miniconda: `https://hub.docker.com/r/continuumio/miniconda`

[5]`https://hub.docker.com/r/pytorch/pytorch`

[6]`https://palletsprojects.com/p/flask/`

[7]I mostly work with time-series data.

[8]In the past, I only used TensorFlow.

Capsule networks use reconstruction as regularisation method in order to encode the instantiation parameters for a given input. This reconstruction is also generating a variation of the input, which is preserving a lot of the details, while smoothing out noise [4]. For example, when trained on the MNIST data set, a capsule network reconstructs the digit it currently receives as input. This feature can be seen as an implicit generator and it should be possible to tap into it and use it for generation. It would significantly reduce the training time, since the generation would no longer be a minimax game between generator and discriminator. But since it is not trained on noise as input, it would be limited in variation. That would be something interesting to look into, but it is also out of scope for this course.

4 Timetable

Task	**Planned Hours**	**Actual Hours**
Getting familiar with data / used libraries	10	10
Reading of related publications	10	15
Coding of solution	25	50
Deploying	-	30
Writing report / creating presentation	10	10

As seen in table above, I did work much more hours than originally anticipated. The reason for that is that coding needed more time than expected and deploying, which I didn't anticipate, turned out to be very time-consuming. I stated the reasons why in section 3.

References

[1] Yongjun Chen et al. "Voxel Deconvolutional Networks for 3D Brain Image Labeling". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD '18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 1226–1234. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219974. URL: https://doi.org/10.1145/3219819.3219974 (visited on 01/22/2020).

[2] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27.* Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.

nips.cc/paper/5423-generative-adversarial-nets.pdf (visited on 11/24/2019).

[3] Pim Moeskops et al. “Automatic Segmentation of MR Brain Images With a Convolutional Neural Network”. In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1252–1261. ISSN: 1558-254X. DOI: 10.1109/TMI.2016.2548501.

[4] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic Routing Between Capsules”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3856–3866. URL: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf (visited on 11/23/2019).

[5] Jiajun Wu et al. “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 82–90. URL: http://papers.nips.cc/paper/6096-learning-a-probabilistic-latent-space-of-object-shapes-via-3d-generative-adversarial-modeling.pdf (visited on 11/23/2019).

[6] Zhirong Wu et al. “3D ShapeNets: A Deep Representation for Volumetric Shapes”. In: 2015, pp. 1912–1920. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Wu_3D_ShapeNets_A_2015_CVPR_paper.html (visited on 11/24/2019).

[7] Yongheng Zhao et al. “3D Point Capsule Networks”. In: 2019, pp. 1009–1018. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Zhao_3D_Point_Capsule_Networks_CVPR_2019_paper.html (visited on 01/22/2020).