CITY
UNIVERSITY OF LONDON
EST 1894

Training a CNN end-to-end to generate point clouds from single images

Individual Project Proposal

Daniel Sikar - MSc Data Science PT2
Supervisor: Artur Garcez

June 19, 2020

1 Introduction

3D reconstruction of real-world scenes has a number of applications in diverse fields such as robotics, medical imaging, computer animation, computer graphics and computer vision. It is primarily a depth estimation problem that can be solved analytically with techniques such as clustering and segmentation, where input pixels in individual, or sequence of, images determine features such as edges and shades, from which volume and depth may be inferred. 3D reconstruction can also be achieved with the use of sensors such as LIDAR (Light Detection and Ranging). The output is a scaled volumetric representation, like a mesh or point cloud, defined as discrete representations of a continuous surface.

Developments in artificial intelligence and machine learning have led to a class of learning algorithms known as CNNs (convolutional neural networks), successfully applied to image classification and natural language processing (NLP), initially, and now being applied to fields such as end-to-end autonomous driving and 3D reconstruction.

Traditionally, computing algorithms have been ”rules” based, where given an input, a sequence of instructions is executed producing an output. The term ”end-to-end” when applied to machine learning algorithms, refers to processes with no such instructions, where, based on input data, a network is trained and ”learns” parameters, to then be able to generate predictions based on new inputs.

3D reconstruction performed by neural networks is subject to similar constraints existing in 2D image classification and segmentation, where it is desirable that the algorithm is robust to occlusion, rotation, noise, etc. Concepts in 2D computer vision are also applicable to 3D, leading to common processes. There is a trend in autonomous vehicle research, to solve the 3D reconstruction problem, and indeed, the driving problem, with computer vision alone. Elon Musk, co-founder and CEO of Tesla, with respect to self-driving cars states that ”(...) right now AI and Neural Nets are used really for object recognition (...), identifying objects in still frames and tying

it together in a perception path planning layer thereafter. (...) Over time I would expect that it moves really to (...) video in, car steering and pedals out" (Musk 2019).

Nguyen and Le 2013 presented a survey on 3D Point Cloud Segmentation, stating advances were made possible by the wide availability of LIDAR, and RGB-D (RGB plus depth) cameras such as Microsoft Kinect, and libraries such as the Point Cloud Library (Rusu and Cousins 2011), making point clouds more attractive to such fields as robotics. The references show the predominance of algorithmic approaches relying on feature engineering. A more recent survey (Guo et al. 2019) shows how deep neural networks in general and CNNs in particular have become widely used, at least in research, to address the problem of 3D computer vision. Though designing effective neural networks is an empirical process and, like algorithmic counterparts relying on engineered features, requires domain knowledge.

Based on this scenario, our **research question** is *can accurate point clouds be generated from single images*, the **purpose of this research** is *to find evidence to inform practice* (Oates 2006) on point cloud generation through a CNN computer vision solution. We aim to 1. segment and classify the objects of interest within the image and 2. generate a point cloud for each object of interest that may be used in addition to, or in conjunction with, LIDAR point clouds. The **product of this research** is expected to be a model able to generate point clouds from single images and the **intended beneficiary** is primarily the author, hoping this work will create academic and professional opportunities, as well as anyone else researching in the area that may use this work as a stepping stone or starting point.

2 Critical Context

A CNN is a class of artificial neural network, or simply neural network, having a basic unit known as a neuron, a mathematical model developed by McCulloch and Pitts (1943), based on biological models of the human brain. Neurons are defined as real numbers, in the form of weighted inputs and a bias, and an activation function, which, given a sum of input values multiplied by weights, plus the bias, generates an output value. The combination of weights and bias represents an encoding, the biological equivalent being a memory.

Based on the McCulloch-Pitts neuron, Donald Hebb (1949) created the first learning algorithm, that enables a neural network consisting of one single neuron to "learn", or encode, a memory, through an iteration process until, given a cost (or error) function, a set of weights and bias is found such that the same desired output is produced after a number of iterations, and the weights and bias reach stable values while the error is minimized. The Hebbian learning algorithm was able to learn simple tasks such as how to compute the OR truth table, but not more complex tasks such as the XOR truth table. This hindrance is known as a linear separability problem, where, using the inputs as coordinates, the output classes (0,1) of the XOR truth table plotted on a Cartesian plane, cannot be separated by a straight line. Solutions were eventually found, one involving the addition of another layer containing one neuron, known as a "hidden" layer, plus a connection between neurons, plus connections between inputs and the hidden layer neuron. The intuition being, a neural network with more neurons and more connections is able to learn more complex tasks. Such neural network architectures are known as multi-layer perceptrons (MLP) (Garcez 2018). In the model previously described, inputs are multiplied by weights. In the CNN model, inputs are "convolved" with kernels in the convolutional layers. The concept is borrowed from the digital signal processing domain, where a vector known as a filter or kernel, is combined

with a signal to generate a filtered output signal. The operation is expressed by:

$$conv(s1, s2)[t1] = \sum_{t=0}^{N2-1} s1[t1 - t]s2[t] \tag{1}$$

where $s1$ is the input signal, $s2$ is the kernel, $N2$ is the length of $s2$ and $t1$ is the time when input signal $s1$ was acquired. Convolution is similar to cross-correlation (where a measure of similarity between two signals is obtained) except convolution involves "flipping" one of the inputs. This can be observed by the indexing in $s1[t1 - t]$ (Pauwels and Weyde 2020).

For the case of a two-dimensional input and kernel, the operation takes the form:

$$J(x, y) = K * I = \sum_{n,m} K(n, m)I(x - n, y - m) \tag{2}$$

Where J is the convolved signal, K is the kernel, I is the input signal, and n, m are the kernel indexes. We see by the input signal indexing $I(x - n, y - m)$ that both input signal axes are "flipped".

A typical CNN architecture will have a number of convolutional layers, followed by a fully connected MLP, that is, where every unit (neuron) is connected to each other. The convolutional layers are able to compress the input, without losing discriminative information, into a lower dimensional space, where different input categories can be efficiently represented. The fully-connected MLP layers then perform classification. Convolutional layers in neural networks with the ability to "self-organize", were introduced by the neocognitron network, proposed by Fukushima 1980, particularly efficient for image pattern recognition.

Finding optimal weights and biases for a neural network in a large search space is a mathematically intractable problem that cannot be solved analytically. Eventually it was solved numerically by the use of back-propagation and gradient descent algorithms, like proposed by Rumelhart, Hinton, and Williams 1986 and Lecun et al. 1998, as well as input normalization and a number of network training optimization algorithms, leading to the wide adoption of neural networks and "deep" neural networks, with several hidden layers such as CNNs.

The neural network design concepts described, have been implemented in several machine learning libraries such as Facebook's PyTorch, Google's TensorFlow and MATLAB toolkits. TensorFlow has also an embedded port, with a subset functionality. Designing a network from scratch can amount in some cases to writing a few lines of code and networks can be trained on more powerful hardware, and the model then be deployed on less powerful hardware.

Prior to deep neural networks, computer vision was generally achieved by algorithms dealing with a classification problem, with each pixel in any given image belonging to one of two classes, object and non-object. This is achieved by feature extraction, clustering and segmentation. Feature extraction is the process of engineering features that allow classes to be distinguished. Clustering partitions the feature space into mutually exclusive regions and segmentation assigns each image pixel to one region. The goal being to define regions that can be separated by a hyperplane (with dimension of the feature space minus one) boundary. Colour is generally used as the feature space for images. The practice of "engineering" features, in the age of deep neural networks has become known as "manual" feature extraction, which is time consuming and something we aim to avoid in this proposed work although the practice is by no means outdated.

Many state-of-the-art machine learning architectures are available as ready-to-use pre-trained models. Implementations of cutting-edge machine learning network architectures are available in

public code repositories like GitHub such that one could train the network and create their own model. A model in this context is an actual packaged object that can be used to make predictions given the expected input format and environment.

Together with the network architectures, one very important aspect is the availability of data to train and test neural networks. The ImageNet (Deng et al. 2009) repository and the related *ImageNet Large Scale Visual Recognition Challenge* spawned a number of 2D image classification deep networks, such as AlexNet, VGGNet, ResNet, etc, that promoted significant advances in 2D computer vision.

The Fast R-CNN network (Girshick 2015) is an example of a deep network designed for image segmentation (Fig. 1). This network is trained end-to-end relying on no engineered features and taking as inputs the image and one optional viewpoint. It consists of convolutional layers followed by a region of interest pooling layer where region proposals are determined, followed by two general fully connected layers followed by one fully connected layer specific to the image, bounding box and viewpoint.

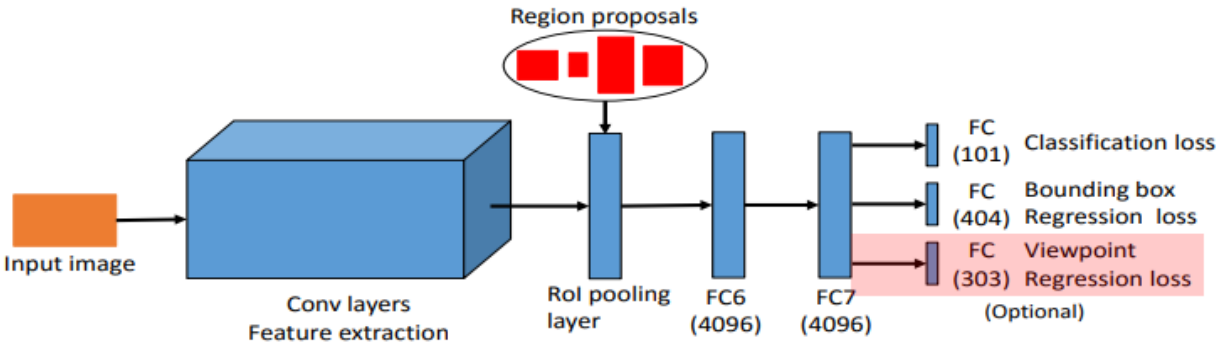

Figure 1: Fast R-CNN architecture. An image and multiple regions of interest (RoIs) are input into a convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss, combining the losses of classification and bounding box regression.

The 3D computer vision research community, having realised the importance of large publicly available datasets, has created a number of 3D datasets. In the lines of ImageNet, these datasets consist of images labelled, with 3D metadata such as object shape and alignment, usually by crowdsourcing or by setting HITs (Human Intelligence Tasks) in Amazon Mechanical Turk. We discuss these further in section 3.2.

Large data make computer vision neural networks more robust to overfitting and better at generalising, and in line with increasing the amount of data available for training and testing to improve network performance, one common practice is data augmentation, achieved by blurring, rotating and adding noise to image subsets. Another option is to generate so called synthetic data. Generative adversarial networks (GANs) are a popular choice for creating synthetic data. In the context of 3D objects, this is also possible with open source packages such as OpenSCAD and Blender.

The ability to generate synthetic training data, for a specific camera intrinsic matrix, with a subset of expected objects in a finite number of poses is attractive, as it constrains the computation

and ultimately the size of the model, where the neural network can be designed and trained to solve a very specific problem.

We aim to replicate neural networks such as implemented by Fan, Su, and Guibas 2016. The work is seminal in bringing point clouds to the fore of 3D reconstruction, underlining the advantages of the simple and uniform structures that are easier to learn and do not have to encode multiple shape primitives or combinatorial connectivity patterns. The network outputs the point positions in a 3D frame, that is determined by the input image and the inferred viewpoint position (Figure 2).

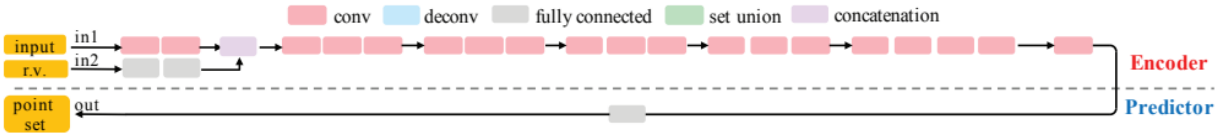

Figure 2: The PointOutNet *vanilla* architecture

The network consists of two distinct inputs, the image input and the reference view (r.v.). The image input if followed by two convolutional layers, while the reference view if followed by two fully connected (MLP) layers. Both inputs are combined in a concatenation layer and then followed by a number of convolutional layers. This is the encoding part, where features are extracted. Finally there is a fully connected layer where a prediction (classification) is made to generate an output point set as shown in Figure 3.

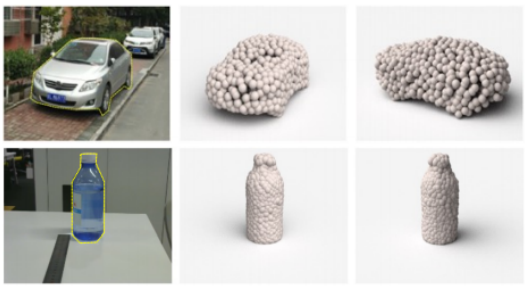

Figure 3: The PointOutNet 3D point cloud reconstruction from a single image. A segmentation mask is used to indicate the scope of the object in the input images on the left. On the right are the reconstructions viewed at 0 and 90 degrees along azimuth. Each point is visualised as a small sphere.

Operations performed by robots which require 3D vision, could benefit from having a compact model, optimised for the camera, generating point clouds that can be used in conjuction with LIDAR point clouds.

We propose to develop a model prototype, that could be deployed at the embedded, desktop or cloud level. As there are many approaches but as far as we can gather, none specifically aimed at mapping image generated point clouds to be coalesced with LIDAR generated point clouds, the work involves some experimentation, replicating existing models and adapting the workflow to the specific task at hand.

3 Approaches: Methods & Tools for Analysis & Evaluation

In this section we describe the components shown in our work breakdown structure (Figure 5)

3.1 Literature Survey

We initially searched for keywords *3D reconstruction*, *point cloud generation*, *deep learning*, *3D point cloud deep learning survey* and found there was rich literature since around 2016. From 2019 there has been a surge in research activity. One marker being Facebook's release of PyTorch3D, with features related specifically to point cloud and image processing. Our survey led to a number of public 3D datasets, tools, models and evaluation metrics discussed next.

3.2 Data & Tools

We have identified 4 similar datasets. We have also identified 4 candidate toolsets. We have set time aside to select and augment the data, and to evaluate the tools.

The **ShapeNet** repository (Chang et al. 2015) "is an ongoing effort to establish a richly-annotated, large-scale dataset of 3D shapes". Subset data are provided in **ShapeNetCore**, containing over 50 common object categories and over 50k unique 3D models, and **ShapeNetSem**, a smaller, more densely annotated set of 12k models and 270 categories, manually verified for labels, consistent alignment, real-world dimensions and estimates of material composition at the category level, as well as estimates of total volume and weight.

The **ObjectNet3D** repository (Xiang et al. 2016) consists of 100 object categories, 90k 2D images, 200k objects in these images and 44k 3D shapes. The images are collected from the ImageNet repository (Deng et al. 2009), while the 3D shapes are from the ShapeNet repository (Chang et al. 2015). In addition to 2D bounding box annotations for objects of interest, a key aspect of ObjectNet3D is that each object in an image is aligned with a 3D shape. The alignment enables the 3D shape to be projected onto the image, where the projection of the 3D shape matches the corresponding 2D object in the image (Fig. 4). The alignment between 2D and 3D provides 3D annotations to objects in 2D images, i.e., the 3D pose annotation and the closest 3D shape annotation for a 2D object.

The **ScanNet** repository (Dai et al. 2017) is an RGB-D video dataset containing 2.5M views in 1513 scenes annotated with 3D camera poses, surface reconstructions, and semantic segmentations. The annotation process includes placing a closely matching 3D model from a large database in the scene, precisely aligning each model to the 3D position of a corresponding model in the reconstructed indoor scene. ScanNet uses the same categories as ShapeNet and ModelNet (see next).

ModelNet (Wu et al. 2014) is a large-scale 3D CAD model dataset containing 151,128 3D CAD models, downloaded 3D CAD models from 3D Warehouse, and Yobi3D search engine indexing 261 CAD model websites, belonging to 660 unique object categories. It was built specifically to train the *3D Shapenets* model.

Time allowing we would like to build a **synthetic dataset**, though this might prove over-ambitious given our tight schedule - the final dissertation must be delivered by December 21st, 2020.

For prototyping we have identified **PyTorch**, including the recently (February 2020) released **PyTorch3D**, **Keras** (Python TensorFlow wrapper), **TensorFlow Lite for Microcontrollers** and

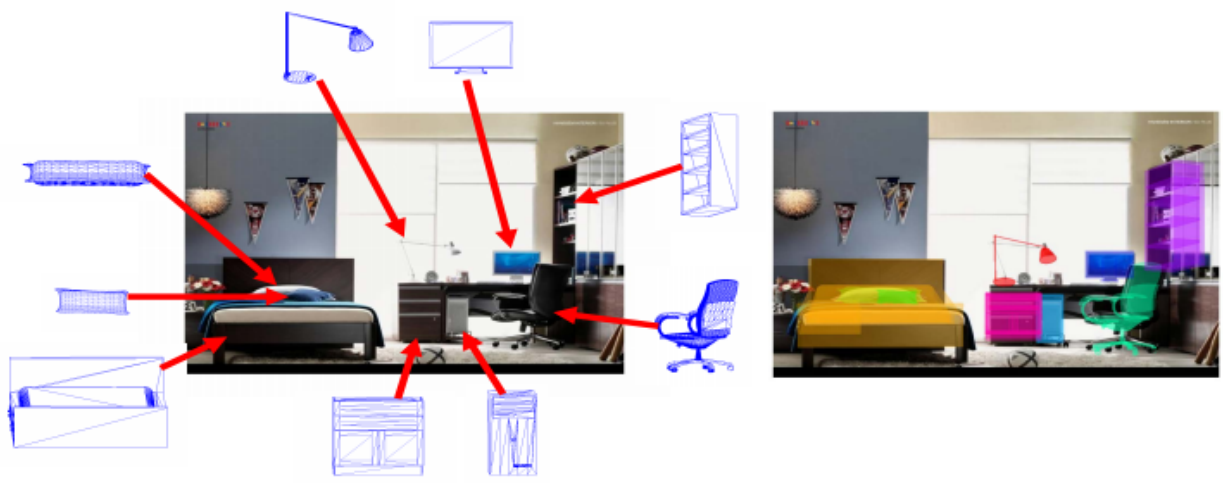

Figure 4: Example image in the ObjectNet repository with 2D objects aligned with 3D shapes. The alignment enables each 3D shape to be projected onto the image, where its projection overlaps with the 2D object as shown in the image on the right

MATLAB toolkits as potential tools, all to be used under open source licenses, except MATLAB to be used under an academic license.

3.3 Develop Models

Once dataset and tools have been selected, we plan to

1. Build on prior work similar to Liu et al. 2016, where, in addition to images, camera intrisic matrix parameters are used as input features. The expectation being to increase the accuracy of predicted geometries

2. Narrow the object segmentation scope to a known set of geometries, on a per-scene basis. The expectation is this will make models simpler to train and deploy

3. For a given scene, generate a prediction using the model, and perform segmentation on the LIDAR point cloud to obtain ground truths

4. Evaluate the predicted point cloud with respect to the ground truth point clouds with the metric proposed in section 3.4

5. Interpolate the predicted point cloud with the nearest LIDAR point cloud object of interest segmentation to obtain the final output

Our aim is to initially replicate existing models, before fine-tuning our own. We identified two models that process images to generate point clouds (RealPoint3D and PointOutNet) and two models that take point cloud inputs to perform segmentation (PointNet and PointNet++):

RealPoint3D (Xia et al. 2018) is a promising model that integrates prior 3D shape knowledge into the network to guide 3D generation. Given a query image, a shape is retrieved from a 3D model database. The image together with the retrieved shape is fed into RealPoint3D to generate a 3D point cloud.

PointNet (Qi, Su, et al. 2016) provides a unified architecture for applications ranging from object classification (identifying a specific object), part segmentation (identifying the parts that make up a single object), to scene semantic parsing (identifying pre-determined families of objects e.g. chairs). PointNet consumes point clouds as inputs so could be used to process LIDAR data.

PointNet++ (Qi, Yi, et al. 2017) builds on PointNet by applying it recursively and capturing local differences e.g. density variability than can come from perspective effects. The idea is to partition the set of points into overlapping local regions by the distance metric of the underlying space. In a fashion similar to CNNs, local features are extracted from small neighborhoods. However, contrary to CNNs, where smaller kernels improve the ability of the network to extract features, in PointNet++ small neighbourhoods may consist of too few points which might be insufficient to allow PointNets to capture patterns robustly, the solution found by the authors is to define neighborhoods at multiple scales.

PointOutNet (Fan, Su, and Guibas 2016) is a conditional shape sampler, capable of predicting multiple plausible 3D point clouds from an input image, and also capable of 3D shape completion. The pipeline infers viewpoint position, which is something we wish to implicitly add to our input. However, as the source code is freely available and the work is influential in generating point clouds from single images, we find it is worth investigating.

3.4 Evaluation

A metric is required for computing how close two point clouds are from each other. In our case, a point cloud generated by LIDAR to be compared with a point cloud generated by our model. We examined the *mean squared error* (MSE), *intersection over union* (IoU), *mean cross entropy* (CE) *loss* and *chamfer distance* (CD) which we found the most suitable.

3.4.1 Chamfer distance (CD)

Using the definition given by ibid., we represent the point cloud as the unordered set in $\mathbb{R}^3$, $S = (x_i, y_i, z_i)_{i=1}^{N}$ where N is a predefined constant. $N = 1024$ is considered to be sufficient to preserve the major structures. We compute a loss $L(S^{pred}, S^{gt})$ between the point cloud prediction and ground truth:

$$L(S^{pred}, S^{gt}) = \sum d(S^{pred}, S^{gt}) \tag{3}$$

where $d(S^{pred}, S^{gt})$ is the chamfer distance:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2 \tag{4}$$

We omit the i indexes given by the authors in equation 3 because we use the loss function to compute the minimum distance between only one output (from our model) and ground truth (from LIDAR). Additionally, we define a threshold T_{loss} such that if the computed loss is below threshold, the point cloud generated by the model is deemed to be acceptable and included in the processing pipeline, otherwise, the generated model is discarded and only the LIDAR ground truth is used for further processing. There are other aspects including scale that must be considered at a future stage.

3.5 Complete Report

The report writing component of this work is intended to be a continuous process, from the project start date until estimated completion date. The final reporting component being the first draft, then two rounds of reviews and addressing comments. The deliverable for this component is the final project dissertation.

4 Work Plan

Our work plan consists of a work breakdown structure (Figure 5) and a Gantt chart (Figure 6). We identified 5 principal components, each successive component has a dependency on, and cannot start before, the previous component being completed, which represents a milestone.

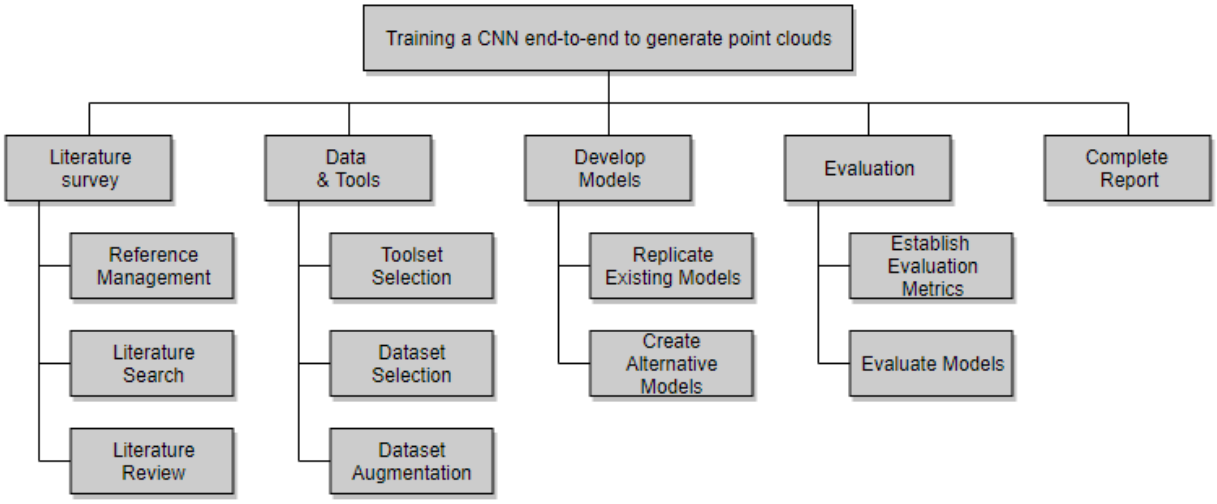

Figure 5: Work breakdown structure

5 Risks

To generate our risk register (Table 1), as suggested in Dawson 2009 we use:

$$I = L * C \tag{5}$$

where Likelihood L is categorized according to three-point scale Low/Medium/High, Consequence C according to five-point scale Very Low/Low/Medium/High/Very High and Risk Impact I is the product.

6 Ethical, Legal and Professional Issues

Having completed *Part A: Ethics Checklist* of the Research Ethics Review Form, as determined by the Computer Science Research Ethics Committee (CSREC 2020), we find this work complies with research ethics guidelines and does not require ethical approval.

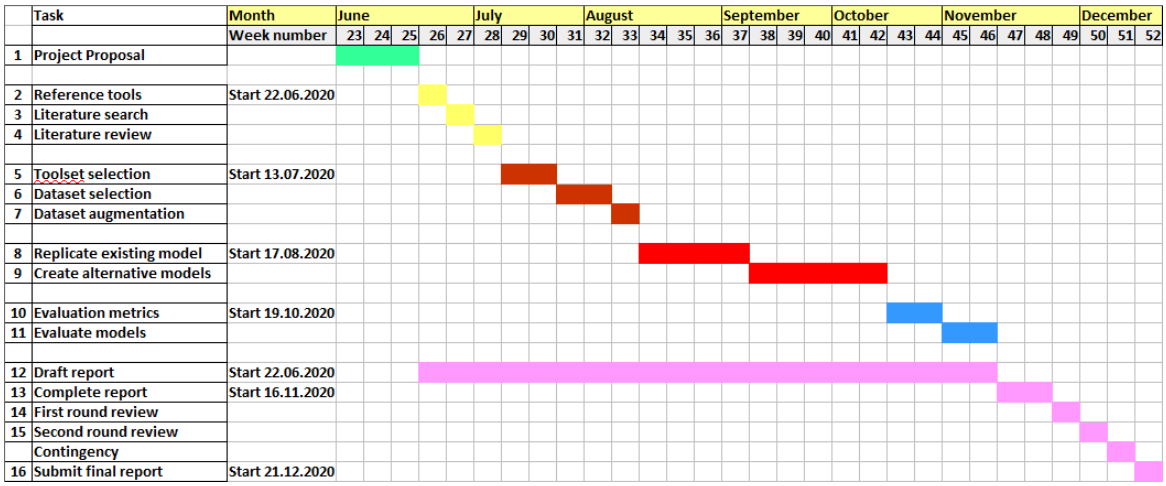

Figure 6: Gantt chart

Description	**L**	**C**	**I**	**Mitigation**
Insufficient domain knowledge	2	4	8	Extend scope for literature survey and toolset selection and decrease scope for alternative models
Cannot replicate all models in chosen environment	1	5	5	Redefine project scope
Unable to create synthetic data	2	3	6	Use existing 3D shape libraries only
Output point clouds too course	2	4	8	Use shape-to-point-cloud lookup database
Unable to create alternative models	2	4	8	Use existing models only
Accidental loss of data.	2	5	10	Keep all source code, reporting and data online (github.com, overleaf.com, aws.amazon.com, camber.city.ac.uk and devcloud.intel.com)
Continued COVID-19 lockdown disruption	2	3	6	Scale down project
Final report delay	2	5	10	One additional week has been added for contingency

Table 1: Risk register

References

Chang, Angel X. et al. (2015). *ShapeNet: An Information-Rich 3D Model Repository*. arXiv: 1512.03012 [cs.GR].

CSREC (2020). *Research Governance Framework*. Department of Computer Science Research Ethics Committee, City, University of London.

Dai, Angela et al. (2017). *ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes*. arXiv: 1702.04405 [cs.CV].

Dawson, Christian W. (2009). *Projects in Computing and Information Systems: A Student's Guide*. 2nd. Pearson Prentice Hall. ISBN: 978-0-273-72131-4.

Deng, J. et al. (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*.

Fan, Haoqiang, Hao Su, and Leonidas Guibas (2016). *A Point Set Generation Network for 3D Object Reconstruction from a Single Image*. arXiv: 1612.00603 [cs.CV].

Fukushima, Kunihiko (1980). "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: *Biological Cybernetics* 36, pp. 193–202.

Garcez, Artur (2018). *INM427 Neural Computing Lecture Notes*.

Girshick, Ross (2015). *Fast R-CNN*. arXiv: 1504.08083 [cs.CV].

Guo, Yulan et al. (2019). *Deep Learning for 3D Point Clouds: A Survey*. arXiv: 1912.12033 [cs.CV].

Lecun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*, pp. 2278–2324.

Liu, F. et al. (2016). "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.10, pp. 2024–2039.

Musk, Elon (2019). *TESLA Autonomy Day Event*. URL: https://www.youtube.com/watch?v=tbgtGQIygZQ&t=2h25m52s (visited on 05/01/2020).

Nguyen, A. and B. Le (2013). "3D point cloud segmentation: A survey". In: *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 225–230.

Oates, Briony J. (2006). *Researching information systems and computing*. 1st. SAGE Publications. ISBN: 9781446235447.

Pauwels, Johan and Tillman Weyde (2020). *INM378 Digital Signal Processing and Audio Processing Lecture Notes*.

Qi, Charles R., Hao Su, et al. (2016). *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. arXiv: 1612.00593 [cs.CV].

Qi, Charles R., Li Yi, et al. (2017). *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. arXiv: 1706.02413 [cs.CV].

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-propagating Errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

Rusu, R.B. and S. Cousins (May 2011). "3D is here: Point Cloud Library (PCL)". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567. URL: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5980567&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5980567.

Wu, Zhirong et al. (2014). *3D ShapeNets: A Deep Representation for Volumetric Shapes*. arXiv: 1406.5670 [cs.CV].

Xia, Yan et al. (2018). *RealPoint3D: Point Cloud Generation from a Single Image with Complex Background*. arXiv: 1809.02743 [cs.CV].

Xiang, Yu et al. (2016). "ObjectNet3D: A Large Scale Database for 3D Object Recognition". In: *European Conference Computer Vision (ECCV)*.

Research Ethics Review Form: BSc, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part.
The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk.
The approval may be ***provisional*** – *identifying the planned research as* likely to involve MINIMAL RISK.
In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/*	**NO**
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/*	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	**NO**
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your	**NO**

	research study (including online content and other material)?	
2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings*	**NO**
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data?	**NO**