1. Introduction

2. Problem Overview

2.1. Used Techologies

2.1.1. Python

Python is an interpreted, high-level, object oriented, open-source programming language created by Guido van Rossum in 1980s [15, 16]. Python is widely used programming language mainly because it is easy to learn, read and extend, therefore there are many 3rd party libraries designed for variety of tasks. Since python is written in C, performance heavy operations can be implemented in C or C++ [17].

2.1.2. NumPy

NumPy is an open source python library used for scientific computing. The fundamental block of NumPy is ndarray which facilitate many mathematical, statistical and logical operations. Since python was not designed for numerical computing most of NumPy code is optimised and precompiled in C [18].

2.1.3. PyTorch

PyTorch is an open source machine learning library. PyTorch defines multidimensional arrays called tensors which unlike NumPy arrays can be operated on by CUDA capable GPU. PyTorch library also contain modules for optimisation, building computational graphs and backpropagation [19].

2.1.4. Kinoptic Dataset

Kinoptic Dataset is a shared dataset of point clouds dedicated for research purposes. The dataset consist of human body point clouds, captured by 10 synchronised Kinects installed in Panoptic Studio. The dataset also contains RGB videos and 3D skeletons. Currently there are footages of 54 sequences together 6 hours long [20].

2.2. Machine Learning

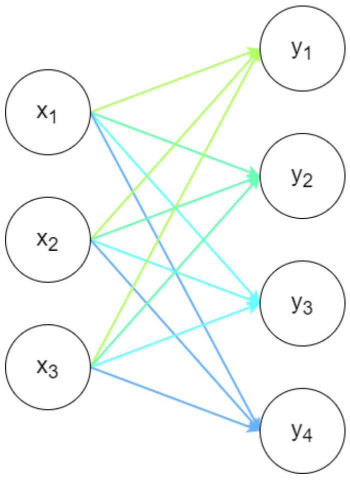
Machine learning is a study of computer algorithms which compute specific tasks and demonstrate the ability to improve their performance automatically from experience [25]. The most common type is supervised learning, in which the program learns mapping:

$$f: x \to y$$

Where x are the features and y are the labels [26]. In order for the machine to learn the mapping *f* it needs examples and desired outputs which substitute the teacher. In optimal scenario the algorithm learns to generalise the mapping *f* to correctly label unseen features.

2.2.1. Multi-layer Perceptron (MLP)

Fully connected layer is a type of layer in neural networks where each component of input signal contributes to each component/neuron of output signal, where contribution of each input signal is defined by weights of neuron [10].



3.

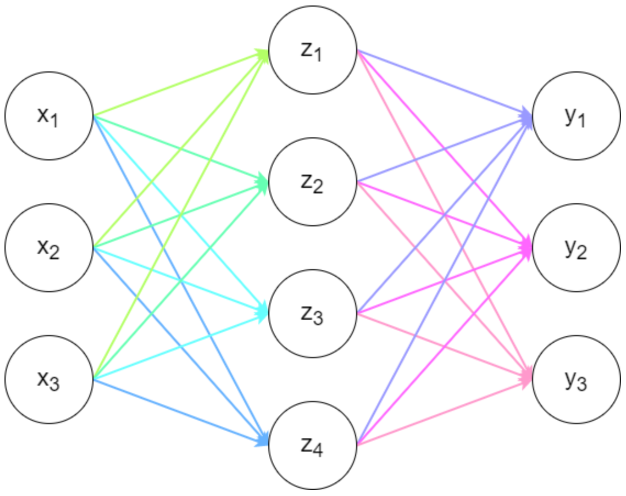
The strength of i-th component of output signal is [8]:

$$y_i = \varphi\left(\sum_{j=1}^{m} w_i x_j + b\right)$$

Where x_j is j-th element of input signal x, w_i is weight of i-th neuron, b is bias and φ is non-linearity function. Using non-linearity enhances the expressiveness of layer.

Goal of MLP is to approximate some function y = *f*(x), which is achieved by learning parameters W in mapping y = *g*(x, W) [9].

Mlp is simply a neural network consisting of multiple fully-connected layers stacked together [10, 11]:



MLP consisting of one input layer, one hidden layer and one output layer (2-layer neural network)

If output y of single fully connected layer can be described as:

$$y = \varphi_1(W_1x + b_1)$$

Then output of n-layer Neural Network is:

$$y = W_n\varphi_n(W_{n-1}\varphi_{n-1}(\dots) + b_{n-1}) + b_n$$

Where W_n is weight matrix of layer n, φ_n is the non-linearity function of layer n and b_n is bias of layer n.

Without the non-liner functions mlp would simply express linear function which is not desired when the approximated function is nonlinear, therefore activation/non-linear functions are used between layers.

3.1.1. Convolutional Neural Networks

Since regular mlps do not scale well in image classification due to large number of parameters a different approach has to be taken. This problem is tackled by Convolutional Neural Networks. Three types of layers are used in convnets, namely: Convolutional, Pooling and Fully-Connected.

Convolutional layer consists of multiple filters of size m x n which are convolved across the width and height of input volume producing activation map [13]. The operation of convolution can be summarised as [14]:

$$g(x,y) = w \times f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx,dy) f(x+dx, y+dy)$$

Where f(x,y) is the function of input signal w is the kernel of dimensions 2a x 2b. Therefore the element in i-th row and j-th column of activation map y can be computed as [11]:

$$y_{ij} = \varphi\left(b + \sum_{k=0}^{m} \sum_{l=0}^{n} w_{kl} x_{i+k,k+l}\right)$$

Oftentimes to reduce the size of activation maps pooling layers are used, most commonly max pooling, which outputs maximum activation in specified region [13].

Generally for input size of $H_1 \times W_1 \times D_1$ and stride S and spatial extent F the output is of dimensions $H_2 \times W_2 \times D_2$ where[13]:

$$H_2 = \frac{(H_1 - F)}{S} + 1$$

$$W_2 = \frac{(W_1 - F)}{S} + 1$$

$$D_2 = D_1$$

After pooling layer the activation map is processed by fully connected layer, where each activation of map is passed into neuron, producing new feature vector.

3.1.2. Softmax Function

Softmax function is generalisation of sigmoid function and is often used to normalise each element of vector z of n scores into probability distribution over n classes [5]. The i-th element of vector z can be normalised as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Then each element of vector z is in range from 0 to 1 included.

3.1.3. Cross Entropy Loss

Cross entropy of probability distribution p_j and q_j over n classes is defined as [6]:

$$\mathbb{H}(p,q) = -\sum_{j=1}^{n} p_j \log(q_j)$$

Since p_j is mostly zero, except for some k, where $p_k = 1$ (because each object can belong only to single class, class k), the formula can be simplified as:

$$\mathbb{H}(p,q) = -p_k \log(q_k) = -\log(q_k)$$

Where q_k can be derived from softmax function, hence:

$$\mathbb{H}(p,q) = -\log\left(\frac{e^{q_k}}{\sum_{j=1}^{n} e^{q_j}}\right)$$

Therefore cross entropy loss is:

$$L = CE + R(w) = -\log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) + R(w)$$

Where R(w) is the regularisation loss

3.1.4. Back Propagation

Back Propagation is type of algorithm used in feedforward neural network, its purpose is to compute the gradient of loss function with regard to weights and intermediate results. Backprop is simply application of chain rule on computational graph starting with the output layer and progressing towards the input layer. The calculated gradients are then used to readjust the weights/parameters of network to minimise the loss function and therefore improve its predictive capabilities [21, 22].

3.1.5. Adam Optimiser

After backpropagation, computed gradients are used to update the weights of network. The way those parameters are updated depends on optimiser. One such optimiser is Adam optimiser which is based on RMSporp using momentum [23]. The algorithm is described in [24]:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

$$\widehat{m_t} = \frac{m_t}{(1 - \beta_1^t)}$$

$$\widehat{v_t} = \frac{v_t}{(1 - \beta_2^t)}$$

$$x_t = x_{t-1} - \alpha \cdot \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \varepsilon}$$

Where t is the current iteration starting at 0, x_t is the weights updated during iteration t, g_t is the gradient during t and α is the learning rate, the rest are hyperparameters.

3.2. Image Segmentation

Image segmentation is the practice of partitioning image into meaningful regions be it group of pixels, points or polygons. The two objectives of segmentation are to decompose images into parts for further processing and to change the representation of images where the parts are more meaningful or efficient for processing [27].

3.3. Point Cloud

A point cloud is a set of points defining shape of objects or space. Each point is defined by x, y and z coordinates in captured space, other information such as colour can also be stored in each point. Point clouds are structures created by 3D scanners such as Lidar or 3D laser scanners [28].

4. Related Work

5. Solution

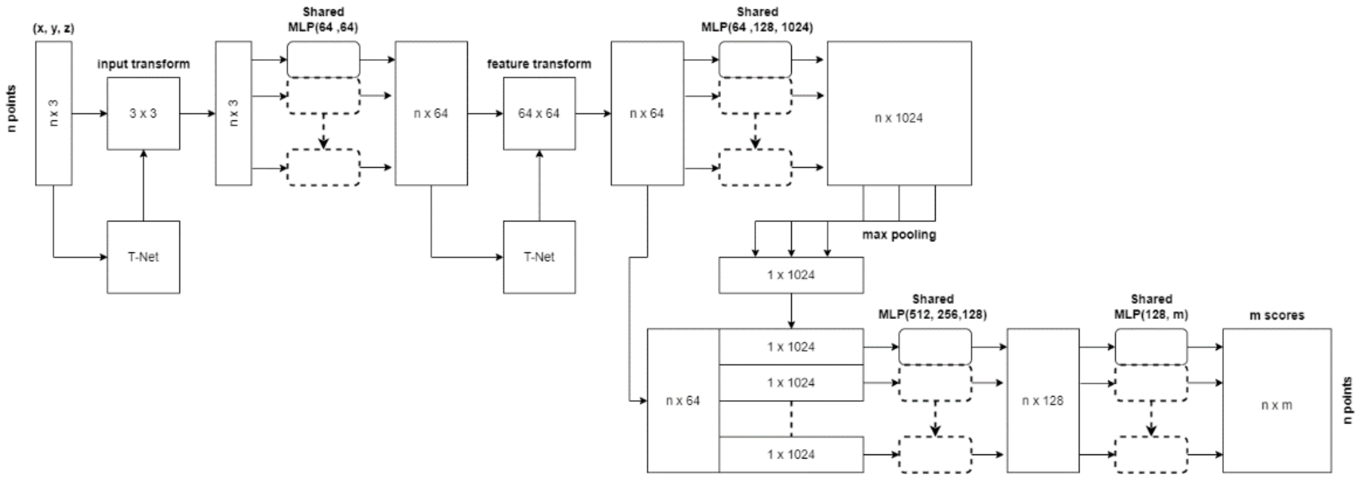
5.1. Dataset Pre-processing

5.2. PointNet

For the task of segmentation on 3D human body point clouds the Deep Neural Network PointNet proposed by Charles R. Qi et al. was used. Their approach was one of the first one to process raw point clouds without pre-processing (Voxelization) showing strong performance in both classification and segmentation.

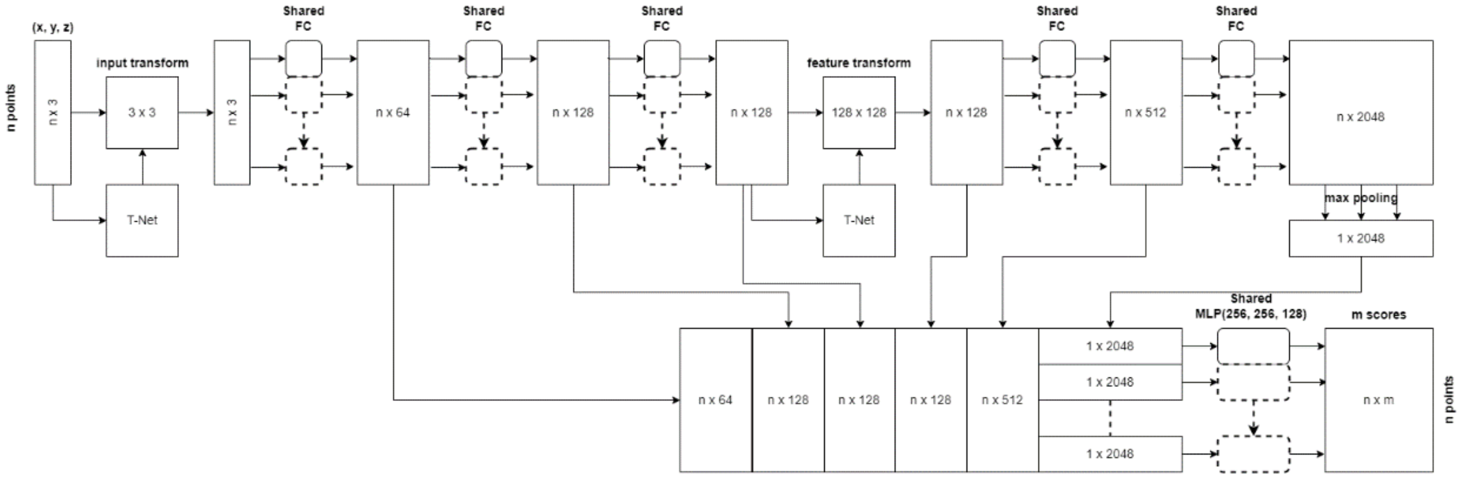
Since point clouds are sets of points the order of points does not matter. The points are not isolated and form meaningful subsets. Point set represent the same information without regard to some transformations [29].

- Segmentation Network Architecture:



The input layer is parametrised and can process point clouds consisting of various amount of points each possessing 3 channels (x, y, z coordinate). The model uses two T-Net networks, both based on PointNet which predict feature transformation matrices. These matrices are used to align input features to a canonical space before feature extraction. After computing the 3 × 3affine transformation matrix, it is directly applied on each point. The transformed point cloud is then processed by shared multilayer perceptron of size 3 × 64, 64 × 64 which produce tensor of dimension n × 64. These shared multi-layer perceptrons are effectively implemented via one dimensional convolutions which convolve across each point of point cloud. Later a second feature transformation matrix of size 64 × 64 is computed from tensor of size n × 64, this tensor is again aligned by the transformation matrix and processed by shared mlps of size 64 × 64, 64 × 128, 128 × 1024. Using max pooling layer across each column on tensor of size n × 1024 produces tensor of size 1 × 1024 called global feature, which generalises the shape of point cloud. Since global feature is insufficient for semantic segmentation global feature is copied and concatenated to intermediate result of feature transformation (local features). This new tensor of size n × (64 + 1024) is then passed through shared mlps of sizes 1088 × 512, 512 × 256, 256 × 128 and then 128 × 128, 128 × m, where m is the number of classes. Since each layer uses shared mlps the order of input features does not matter moreover the max pooling layer aggregates information from each point into global feature which is invariant to input order. This is an important property of point net since point clouds are sets of points and therefore each point cloud represents the same structure no matter the order.

- Part Segmentation Network Architecture:



The architecture is very similar to regular PointNet for semantic segmentation, it consists of input layer of size n × 3, transformation matrix 3 × 3 from T-Net is gained and applied to input tensor, then shared multilayer perceptron are applied 3 × 64, 64 × 128, 128 × 128 then another T-Net is used to gain transformation matrix of size 128 × 128 which is used on to align tensor, which is the processed by shared mlps of sizes 128 × 512, 512 × 2048. Similarly max-pooling is used to gain global feature this time of higher dimensionality. Each intermediate result of shared mlp is concatenated with n copies of global feature to produce tensor of size n × 3008 processed by mlps of sizes 3008 × 256, 256 × 128, 128 × m to produce scores n × m. One-hot was not used.

- T-Net Architecture:

T-Net is similar to PointNet for classification tasks. It consists of shared mlps of sizes m × 64, 64 × 128, 128 × 1024 where m is the second dimension of the input tensor of size n × m. After the last convolution max-pooling is used to obtain tensor of size 1 × 1024 which is processed via fully connected layers of sizes 1024 × 512, 512 × 256, 256 × (m × m). The one dimensional feature vector is then transformed to transformation matrix of sizes m × m. All layers except the last one use ReLU activation function and batch normalisation. Dropout is used on the last fully connected layer with probability 0.3.

6. Testing Evaluation

6.1. Mean Intersection over Union (mIoU)

IoU is metric proposed by Jaccard [2, 1] for measuring similarity of two attributes associated with finite set of objects. Nowadays IoU is commonly used in segmentation tasks as evaluation metric [3, 7].

The general formula of IoU also known as Jaccard index is:

$$\sigma_{ik} = \frac{N(A_i \cap A_k)}{N(A_i \cup A_k)}$$

Where $N(A_i \cap A_k)$ is the number of times an object possesses both attributes i and k, while $N(A_i \cup A_k)$ is the number of times an object possesses one or both of attributes i and k. In segmentation tasks Jaccard index has the form:

$$IoU_j = \frac{|A_j \cap B_j|}{|A_j \cup B_j|}$$

Where IoU_j is the intersection over union of class j, with prediction A_j and ground truth B_j. Then mIoU of n classes can be calculated as:

$$mIoU = \frac{\sum_{i=1}^{n} IoU_i}{n}$$

6.2. Accuracy

Accuracy was measured as the number of points correctly predicted out of all points in point cloud. Then overall accuracy was calculated as the mean of all accuracies during testing.

6.3. PointNet Testing

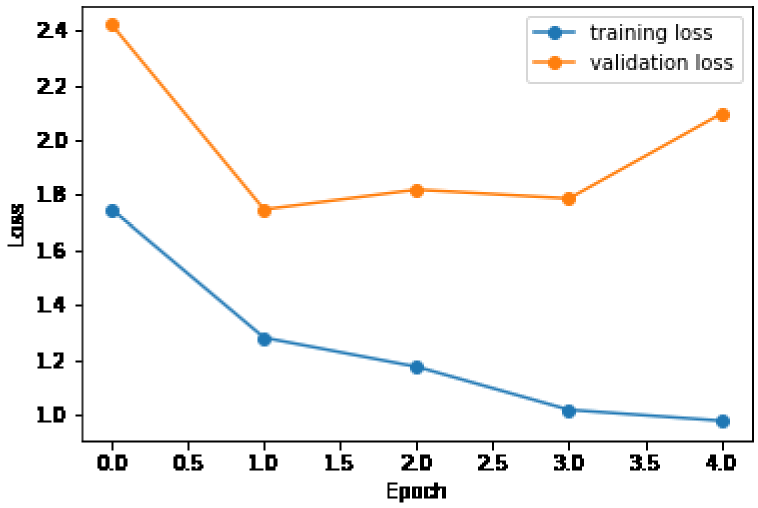
In this thesis we reimplemented and trained both of their proposed segmentation models.

Dropout with probability 0.3 was used on the last two fully connected layers of both architectures. As the loss function cross entropy loss was used with regularisation term:

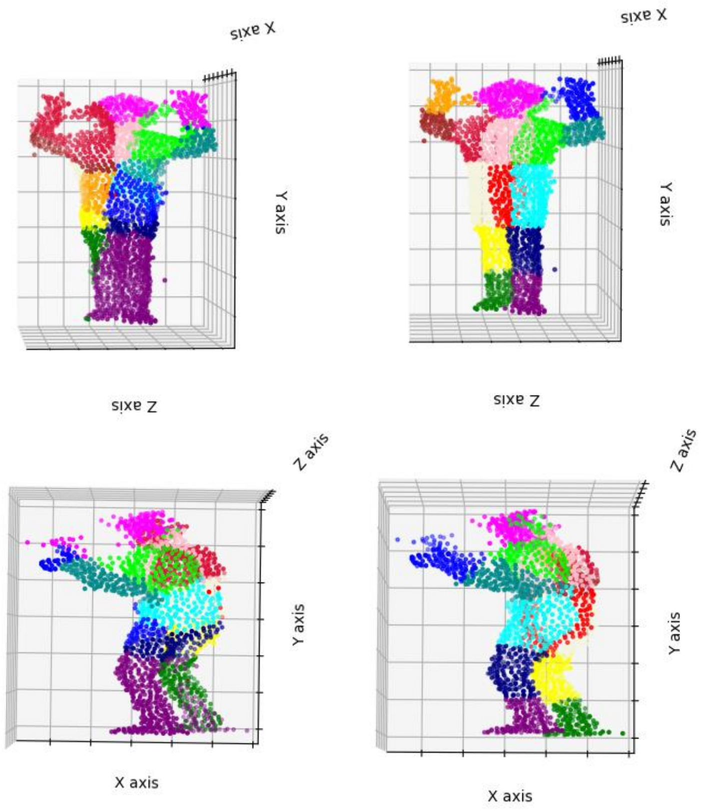
$$L_{reg} = 0.001 \cdot ||I - AA^T||_F^2$$

As an optimiser was used Adam optimiser with learning rate α = 0.001, β_1 = 0.9, β_2 = 0.999. The learning rate was multiplied by 0.5 every 20 epochs. Size of batches was 32.

The change of validation and training loss during training of PointNet for semantic segmentation:



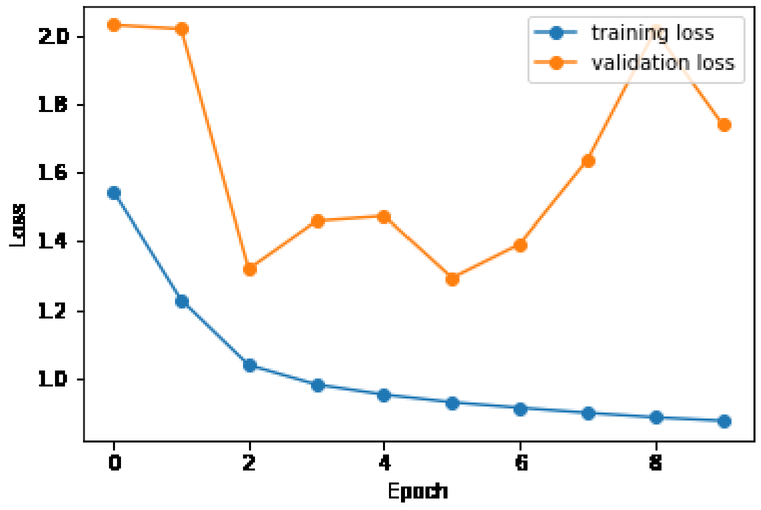
The model stopped improving after epoch number 1 (second training iteration). The validation loss starts slowly increasing therefore the model trained after epoch 1 was used for testing.



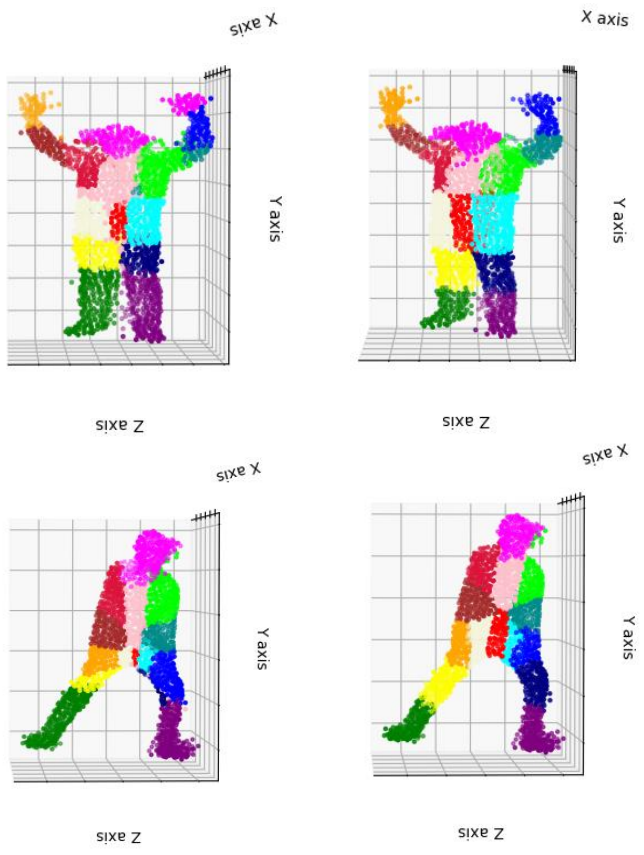
Left - network prediction, right - ground truth.

Overall testing accuracy was approximately 44.432%, overall mIoU was approximately 24.742%.

The change of validation and training loss during training of PointNet for part segmentation:



The network stopped improving after 2nd epoch (3rd iteration), therefore the model trained during 2 epoch was used for testing.



Left - network prediction, right - ground truth.

Overall testing accuracy was approximately 53,39%, overall mIoU was approximately 32,909%. Which is significant improvement compared to model for semantic segmentation

The shape of validation loss should be to some extent similar to training loss, or ideally should be approaching zero loss similarly to training loss. The sudden rise of validation loss is caused by overfitting an phenomenon which occurs when the trained model starts to "memorise" correct labels for training data. If overfitting happens the model no longer generalises the function for problem stated, such model is undesired.

The proposed solutions (model, optimiser, hyperparameters) for point cloud segmentation needs to be readjusted for human body segmentation since the performance is relatively poor.

7. Conclusion

8. References

1. T. T. Tanimoto, "An Elementary Mathematical Theory of Classification and Prediction," International business Machine Corporation, New York, New York, USA 1958
2. P. Jaccard, "The Distribution of the Flora in the Alpine Zone," New Phytologist, vol. 11, no. 2, pp. 961-967, Feb. 1912, doi: 10.1111/j.1469-8137.1912.tb05611.x
3. H. Rezatofig, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese "Generalised Intersection over Union." Stanford. https://giou.stanford.edu/ (accessed Feb 1, 2022)
4. P. Tosteberg, "Semantic Segmentation of Point Clouds using Deep Learning" M.S. thesis, Dept. Electrical Engineering, Linköping University, Sweden, 2017.
5. I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Cambridge, Massachusetts, USA, MIT Press, 2016 [online]. Available: https://www.deeplearningbook.org/contents/mlp.html (accessed Feb 1, 2022)
6. K. P. Murphy, Machine Learning A Probabilistic Perspective, Cambridge, Massachusetts, USA, MIT Press, 2012
7. "Semantic Segmentation on Semantic3D." Papers with Code. https://paperswithcode.com/sota/semantic-segmentation-on-semantic3d (accessed Feb 1, 2022)
8. P. Návrat et al. Umelá inteligencia, Slovak University of Technology in Bratislava, Bratislava, Slovakia, 2002.

9. I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Cambridge, Massachusetts, USA, MIT Press, 2016 [online]. Available: https://www.deeplearningbook.org/contents/mlp.html (accessed Feb 1, 2022)
10. Course CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, Stanford, California, USA. [Online]. Available: https://cs231n.github.io/neural-networks-1/ (accessed Feb 1, 2022)
11. M. Nielsen: Neural Networks and Deep Learning. [Online] . available: http://neuralnetworksanddeeplearning.com/ (accessed Feb 1, 2022)
12. M. Nielsen: Neural Networks and Deep Learning. [Online] . available: http://neuralnetworksanddeeplearning.com/ (accessed Feb 1, 2022)
13. Course CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, Stanford, California, USA. [Online]. Available: https://cs231n.github.io/neural-networks-1/ (accessed Feb 1, 2022)
14. Example of 2D convolution. [Online]. Available http://www.songho.ca/dsp/convolution/convolution2d_example.html (accessed Feb 1, 2022)
15. Python [Online]. Available https://www.python.org/about/ (accessed Feb 1, 2022)
16. Python 3.10.2, General Python FAQ. [Online]. Available https://docs.python.org/3/faq/general.html (accessed Feb 1, 2022)
17. Python 3.10.2, Extending Python with C or C++. [Online]. Available. https://docs.python.org/3/extending/extending.html (accessed Feb 1, 2022)
18. What is NumPy. [Online]. Available. https://numpy.org/doc/stable/user/whatisnumpy.html (accessed Feb 1, 2022)
19. PyTorch Documentation. [Online]. Available. https://pytorch.org/docs/stable/index.html (accessed Feb 1, 2022)
20. H. Joo, H Liu et al. “Panoptic Studio: A Massively Multiview System for Social Motion Capture”, IEEE Transactions on Pattern Analysis and Machine Intelligence http://domedb.perception.cs.cmu.edu/ptclouddb.html, 2017
21. Course CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, Stanford, California, USA. [Online]. Available: https://cs231n.github.io/neural-networks-1/ (accessed Feb 1, 2022)
22. K. P. Murphy, Machine Learning A Probabilistic Perspective, Cambridge, Massachusetts, USA, MIT Press, 2012

23. Course CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, Stanford, California, USA. [Online]. Available: https://cs231n.github.io/neural-networks-1/ (accessed Feb 1, 2022)
24. D. P. Kingma, J. L. Ba: Adam: A Method for Stochastic Optimization, 2017
25. T. Mitchell, M. Hill, Machine Learning (1997). [Online]. Available http://www.cs.cmu.edu/~tom/mlbook.html (accessed Feb 1, 2022)
26. K. P. Murphy, Machine Learning A Probabilistic Perspective, Cambridge, Massachusetts, USA, MIT Press, 2012
27. L. Shapiro, Computer Vision, The University of Washington, Seattle, Washington, USA (2000)
28. "What Are Point Clouds?" [Online]. Available tech27.com/resources/point-clouds/ (accessed Feb 1, 2022)
29. Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, Stanford University, Stanford, California, USA