Mesh-RCNN report-(Alon de jager,Ben banuz)

Abstract

Until Today we made big advancements in image detection that give us the ability to detect every object in images but it ignores the 3d structure of the objects . until now advancements in 3d structure prediction are focused on objects isolated in an image with no background or other objects. Mesh-RCNN is a system that first detects objects in an image and then produces a 3d triangle mesh of that object.

First the system uses a Mask-RCNN backbone to detect objects and extracts features from the image, then it predicts a voxel probability occupancy grid of the 3d structure of the object then it converts it to a 3d mesh triangle graph that goes through refine stages witch are graph convolutions that work on the vertices and edges of the graph. The system will have 2 architectures that will operate each on a different dataset. the first one is ShapeNet dataset that has rendered images of isolated objects, the architecture of the system here will start with a ResNet-50-FPN backbone (not like the full Mesh-RCNN model that has a Mask-RCNN backbone) and the rest of the mesh prediction branch mentioned before . the second one is the Pix3D dataset that contains full real world images of objects from an IKEA catalogue, here we will implement the full Mesh-RCNN model that was explained before.

Intro

The goal of the system

Advancements in 2d perception that gives us the ability to detect objects in real world images and produce corresponding bounding boxes masks and key points don't consider the 3d shape of the detected objects , on the other hand concurrent 3d shape prediction from images Is preformed on datasets of synthetic images that are rendered and contain isolated objects ignoring the full 3d world with not so perfect visualization of objects. Mesh-RCNN is looking to combine the ability to detect 2d objects in real world images and the construction of 3d mesh of an object from an image together, the system will use a state of the art Mask-RCNN for 2d detection and recognition, it outputs class labels ,masks and bounding boxes of objects in the images, with that we extract features of the different objects in the image and then we use a mesh prediction branch that outputs a 3d shape of all detected objects in the image.

Other works

Compared to prior work on 3d shape recognition Mesh-RCNN strives to produce 3d meshes of objects in full real world images therefore the predicted 3d meshes will be in different complexity's and geometry's. Prior work on mesh prediction begin from deforming a fixed mesh template limiting their 3d mesh predictions to fixed topologies, while Mesh-RCNN overcome this drawback using a different 3d object representations , first it will produce a voxel grid representing the object 3d shape, from then the system converts the voxels to a mesh representation turning the voxels into a graph then the graph goes through stages of graph convolutions that refine the mesh resulting in a high resolution mesh in a vary of topologies. Other works for example pixel2Mesh start from a fixed ellipsoid mesh that is being refined with object extracted features to the objects 3d mesh representation, other approaches use 3d points or geometrical basic blocks to represent the 3d shape of the object and some output a voxel representation of the object. Some works were focused on

multi-view reconstruction of objects while our system focuses on single-image shape reconstruction.

Future improvment

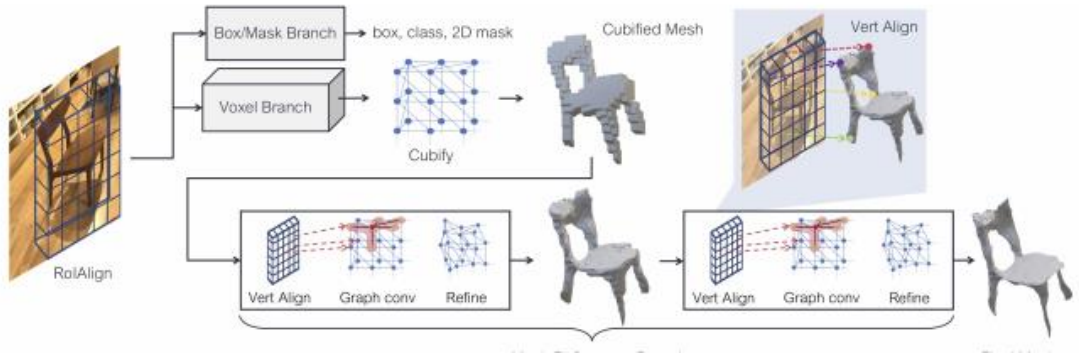
our system inputs a 2d image and from it draws the objects 3d shape a future improvement can involve inputting a RGB-D images making the addition of the depth dimension very helpful to reconstructing the 3d shape.

Methods ,Implementation and experiments

We implemented the full Mesh RCNN models described in the paper with all of its abilities and extensions, for the experiments we used 2 architectures, one to each dataset.
as we was agreed upon with the TA Chaim Baskin we did not add upon the original article due to its technical nature.

Overall Architecture overview

the general structure of our model is a backbone feature extractor tasked with identifying the different objects in the image followed by a GCN(Graph convolution network).
the backbone omits a feature map which is then converted into a voxel occupancy grid, which we convert into cubified mesh using a novelty algorithm.
the resulting cubified mesh is then processed by several refinement stages resulting which are tasked with modifying vertice positions by using graph convolutions and projections and pooling of features from the backbone.



pix3d architecture RoiAlign is the maskrcnn output that is fed to the GCN

ShapeNet architecture

the purpose of this model is given a rendered image of a single object perform classification and 3d model estimation.

for the shapenet backbone we use a ResNet50 model. intermediate feature maps are extracted and fed into the GCN model. we use one feature map in order to produce the voxel grid and initial mesh whereas the 4 other feature maps are used in order to pool features from the original image into our vertices using bilinear interpolation(the vert Align layers)

Pix3d architecture

the aim of this model is to jointly detect,classify ,segment,and produce a 3D representation to all detected object in an image.
this architecture is the highlight of the article and also the more challenging task.
for the backbone we use the mask-rcnn model which is a state of the art object detection and segmentation model. we use the ROI output of the backbone as an input to the GCN to be used to create the initial cubified mesh and a feature maps to pool features from.
the GCN model is architecture is largely the same.

Internal graph representation

In order to perform fast and efficient computations on graph structures we follow the common approach of stacking vertices together resulting in a |V| x 3 matrix.
and using a sparse adjacency matix in order to allow feature propagation between connected vertices. this approach allows us to have a fast model even when it handles large amounts of data (crucial during early parts of training). An additional reason for why we chose this representation is because we allow each graph to have a different amount of vertices.

The cubify layer

In contrast to other works like Pixel2Mesh which uses the same initial mesh in the form of an ellipsoid, we use the voxel representation in order to have produce for each sample a unique starting mesh.
the process of converting a voxel occupancy grid into a mesh is done in the cubify layer.
as each such grid is of size 32x32x32 it was important to keep it's computation as efficient as possible because an untrained model has a tendency to predict huge graph 100000~vertices per sample.
our implementation solves this issues by utilizing custom convolution and vector operators in order be as efficient as possible.

Mesh Sampling

comparing meshes to one another is a difficult task (different scale size etc.) thus another approach is required we achieve this by sampling each ground truth and output mesh online during training. this operation must have 2 properties firstly it must be differentiable in order to allow gradient propagation from back to the model and secondly it must have time efficiency as each mesh(predicted and targets) is sampled multiple times during loss calculation. we achieve this by implementing the approach mentioned in the article.

Utils

as our data is not standard we implemented various methods of manipulating and visualizing voxel grid and 3d meshes.

Data pre processing

For the experiments we used normalized RGB images from the datasets , and because our datasets don't include only images we implemented a serialization and deserialization of voxels and meshes, we sampled old meshes that came from the datasets and created our own scaled meshes of the objects, and we created ground truth pointclouds from sampling meshes, we also resampled voxel grids to fit our sizes ,to view and understand the data that we were dealing with we implemented visualization tools for voxels ,meshes and pointclouds.

Loss metrics

In our experiments we used couple of loss formulations to evaluate performance of the model.

GCN loss metrics:

Voxel loss - binary cross-entropy between predicted voxel occupancy probabilities as outputted from the voxel branch and true voxel occupancies.

The following 3 metrics are computed after each refinement stage using sampled pointclouds from predicted meshes.

Edge loss - $\frac{1}{|E|} * \sum_{(v,v')\in E}\left||v - v'|\right|^2$ where E are the edges in the predicted mesh graph, penalizes edges with large length, this is believed to preduce more pretty meshes.

Normal loss - $-|P|^{-1} * \sum_{(p,q)\in A_{(P,Q)}}\left|u_P * u_q\right| - |Q|^{-1} * \sum_{(q,p)\in A_{(Q,P)}}\left|u_q * u_P\right|$

Where P and Q are pointclouds and u_p is the unit normal to point p

And $A_{(P,Q)} = \left\{\left(P,\ argmin\left||p - q|\right|\right) : p \in P\ q \in Q\right\}$

Penalizes mismatched normals between 2 pointclouds, , this loss helps the predicted object to have a good orientation.

Chamfer loss - $-|P|^{-1} * \sum_{(p,q)\in A_{(P,Q)}}\left||p - q|\right|^2 - |Q|^{-1} * \sum_{(q,p)\in A_{(Q,P)}}\left||q - p|\right|^2$

Where P and Q are pointclouds and u_p is the unit normal to point p.

And $A_{(P,Q)} = \left\{\left(P,\ argmin\left||p - q|\right|\right) : p \in P\ q \in Q\right\}$

Penalizes mismatched point positions between 2 pointclouds. This loss makes the vertice positions prediction be more accurate.

And the classifier loss for shapenet is a negative log likelihood , and for pix3d we use the one from the Mask RCNN.

For Pix3D we used the following loss metrics:

We used the same losses mentioned in the ShapeNet model and in addition to the standard mask-RCNN loses

Training method

Training options

as we described in the cubify section an untrained backbone and voxel branch can cause memory explosion in the GCN model in order to address this issue we've decided to provide the option to train the backbone as a standalone model and to train a simpler model which terminates after voxel prediction(omitting refinement stages).

Shapenet

We trained the ShapeNet model on 1500 samples of airplane images, with the Adam optimizer , with weight decay of 0 and learning rate of 10^{-4} , on batch size 3 , with cubify threshold of 0.2, and the weights we gave the losses are $l_{voxel} = 1, l_{cham} = 1, l_{norm} = 0, l_{edge} = 0.5$,with residual for 25 epochs.

Pix3D

We trained the Pix3D model on 950 samples of desk images, with SGD as optimizer, with weight decay of 10^{-4} , on batch size 4, with cubify threshold of 0.2 , linearly increasing the learning rate from 0.002 to 0.02 over the first 1K iterations, then decaying by a factor of 10 at 8K and 10K iterations. We initialize from a model pretrained for instance segmentation on COCO, and the loss weights are $l_{voxel} = 3, l_{cham} = 1, l_{norm} = 0.1, l_{edge} = 0.5$ for 60 epochs.

During training on Pix3d we filter the features that come out from the backbone by calculating its correspondent bounding box iou with the ground truth bounding boxes, and pass only the best scoring one.

Data parallel

as a result of the large memory consumption of the models and because of server availability issues we sought to speed our training process using data Parallelism. because our models have non standard inputs/outputs such as graphs and lists and dictionaries. custom functions where implemented in order to each data Parallelism. this resulted in significant speedup which enabled us to produce our results.

Evaluation metrics

To evaluate the models we wrote scripts that calculate some evaluation metrics on the output of the model on test set, the evaluation metrics we used are the following:

For ShapeNet:

We sample 10k points uniformly at random from the surface of predicted and ground-truth meshes, and use them to compute Chamfer distance

Normal distance, and F1τ at various distance thresholds τ , which is the harmonic mean of the precision at τ (fraction of predicted points within τ of a ground-truth point) and the recall at τ (fraction of ground truth points within τ of a predicted point). Lower is better for Chamfer distance; higher is better for all other metrics, we computed $f1^{0.1}, f1^{0.3}, f1^{0.5}$

For Pix3d:

we use the same mesh metrics and add the mask-rcnn specific metrics:

AP^{box} – the average percentage of predicted bounding boxes with iou greater then 0.5 with the ground truth bounding boxes.

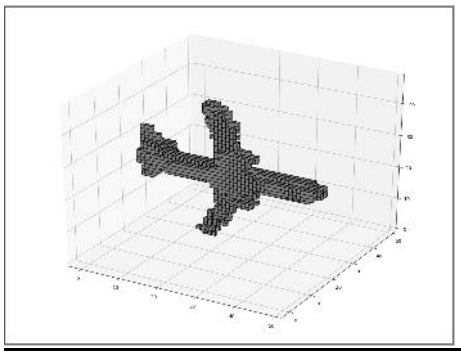
AP^{mask} – the average percentage of predicted masks with iou greater than 0.5 with the ground truth masks.

AP^{mesh} $-$ it is the mean area under the per $-$ category precision
$-$ recall curves for $f1^{0.3}$ at 0.5

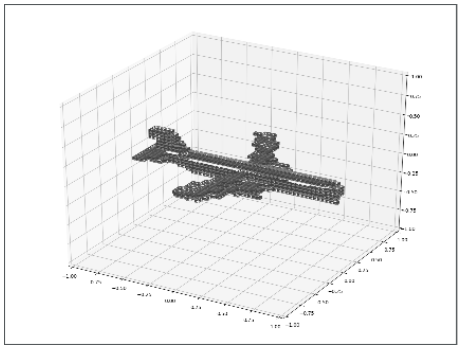
In our experiments the $f1^{c}$ scores are redounded because we trained our models on one class in the dataset do to not enough resources on the servers and time.

Results

On the right a shapnet image of an airplane and on the left its voxelized prediction from our trained model:

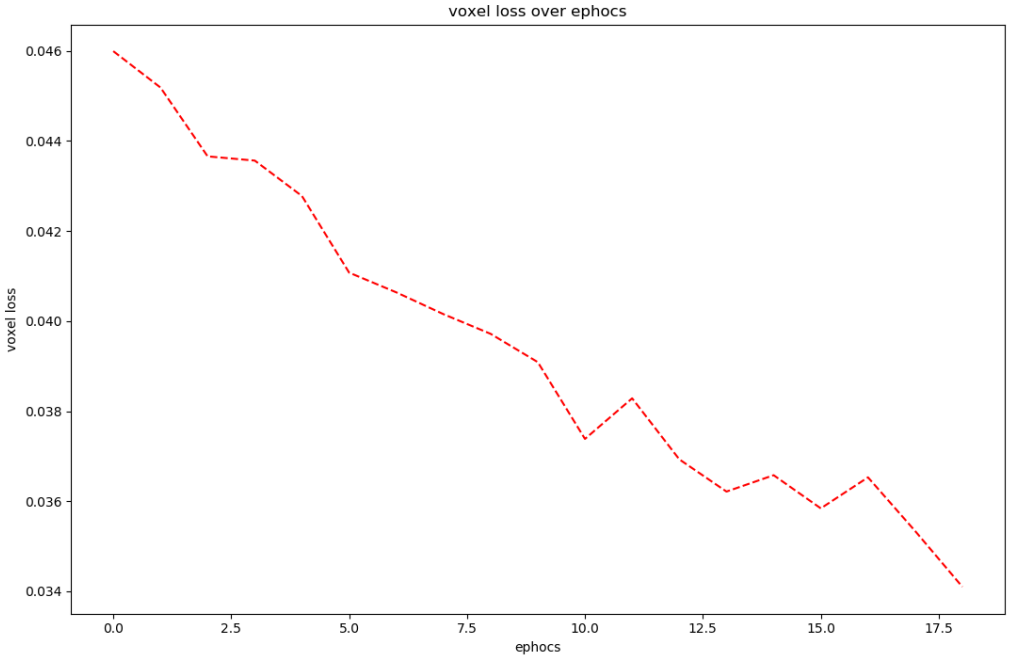


The final mesh prediction:

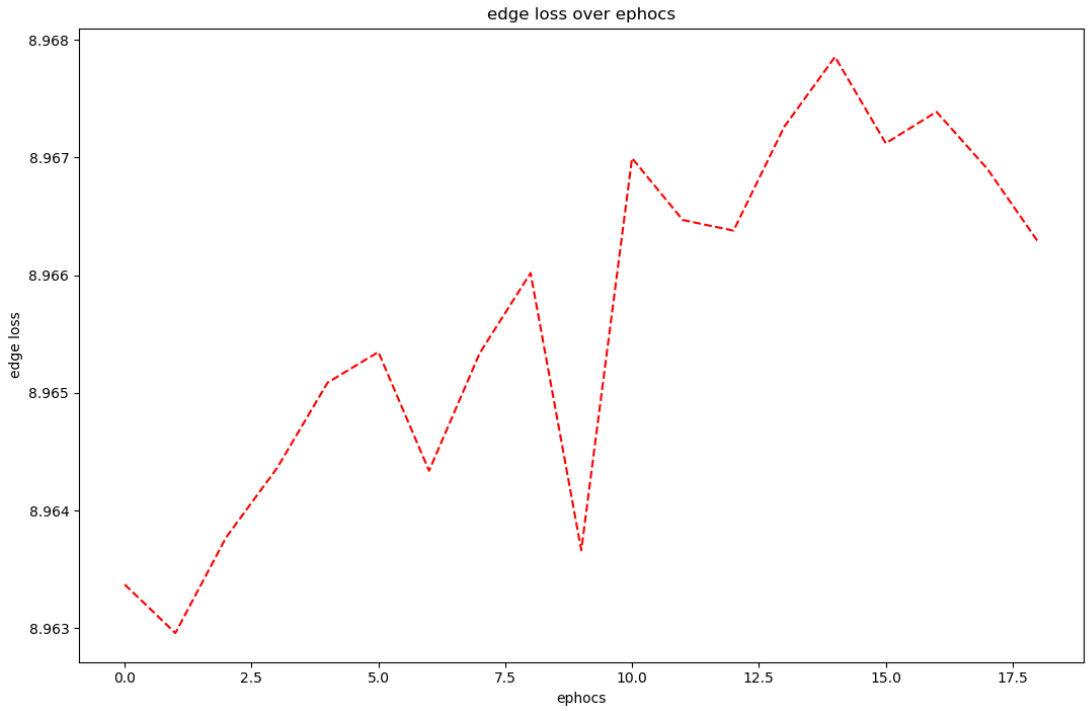


ShapeNet

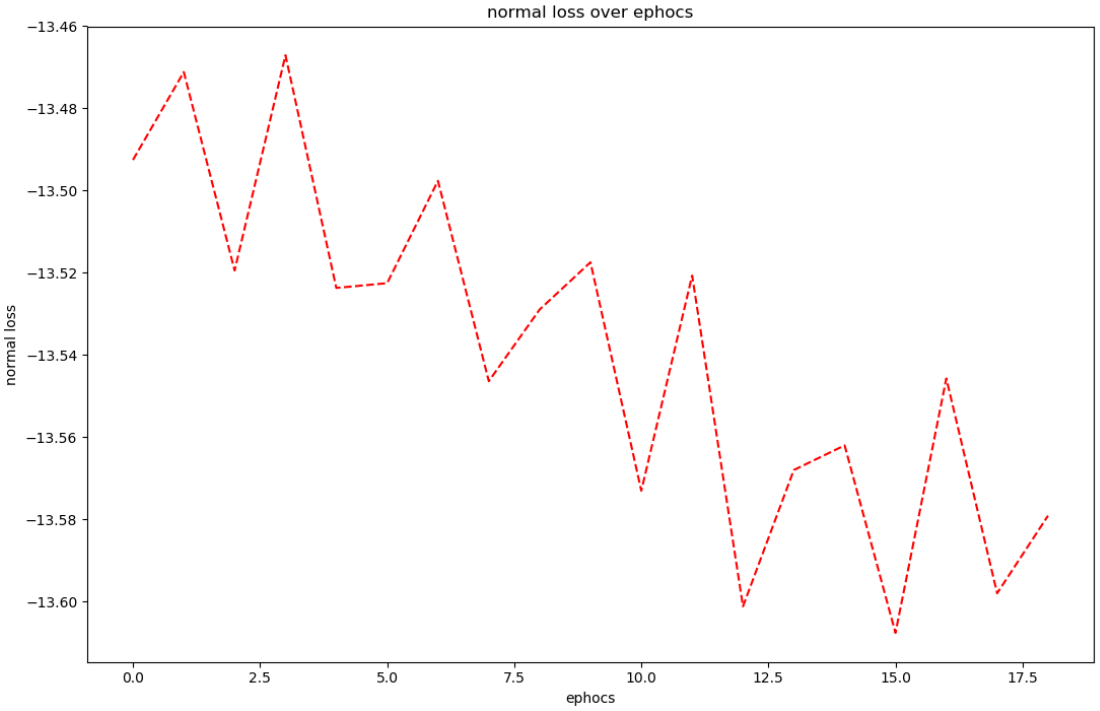
The shapeNet model we trained preformed well and produced very accurate voxels and mesh prediction as seen in the example above this is largely because the ground truth meshes where produced by our own Cubify layer which is used in our model.
the loss metrics as shown below have little to no changes and are displayed in a manner that emphasizes the small incremental changes.
in this way it reminds us of GANs in the way that even if the loss does not change visual improvements are still being made by the model.



voxel loss over epochs – as we can see the voxel loss avg goes down over the epochs witch corresponds with accurate voxel prediction we get from shapeNet, but as we can see the scale of the downfall of the loss is tiny , so we can infer from it that like in training a gan this loss metric doesn't effect how much preety is the voxel grid.

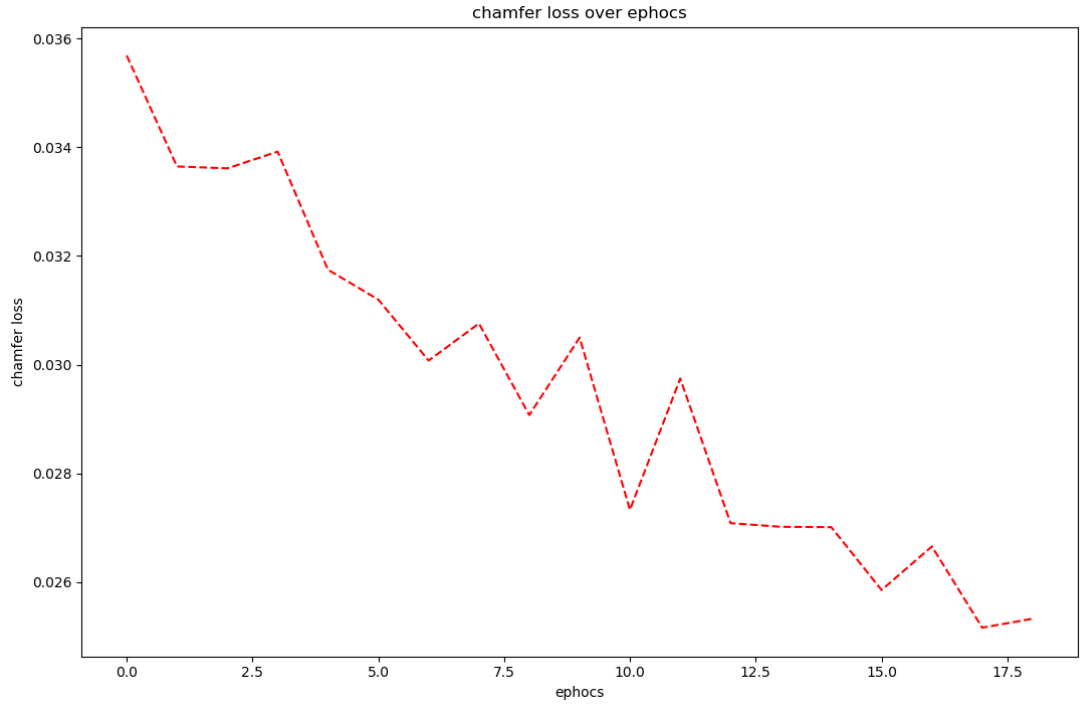


edge loss over epochs- as we can see the edge loss gets bigger over the epochs , that tells us that the predicted meshes are with longer edges each epoch, we can infer from it That the predicted mesh is bigger and intuitively more pretty.

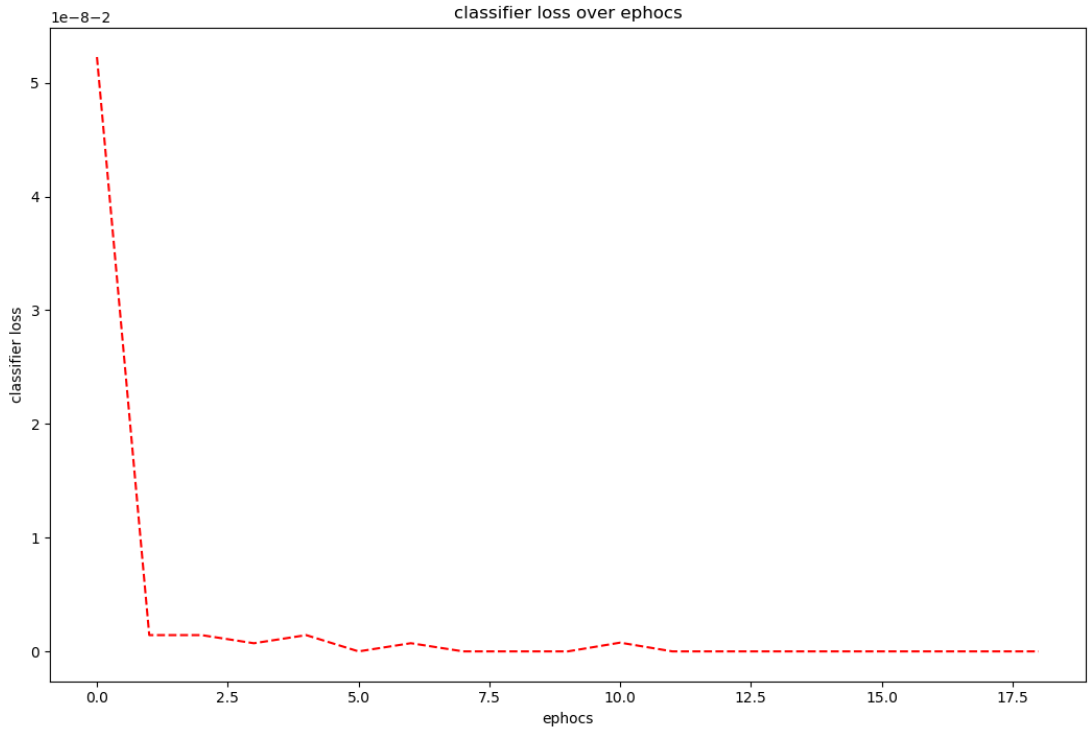


normal loss over epochs- As we can see the normal loss is getting smaller over the

Epochs that means the normal of the predicted and ground truth pointclouds are getting similar, but the decrease is small so we can infer doesn't effect how pretty the predicted pointcloud is but it does show a trend of improvement in the model.



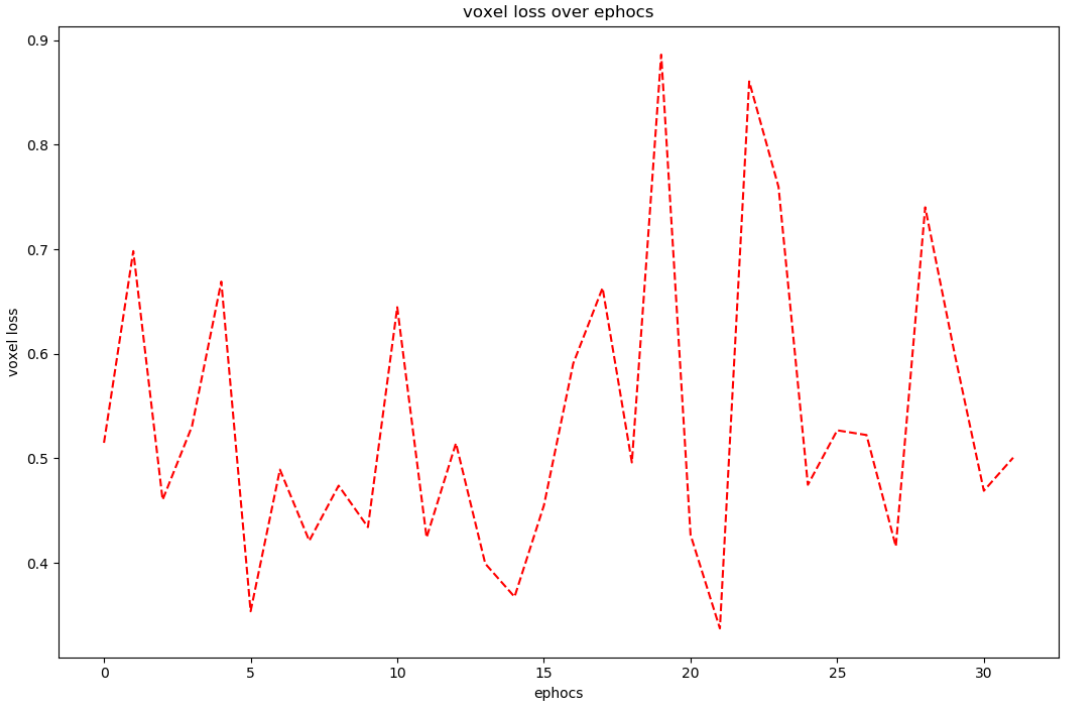
Chamfer loss- as we can see it decreases over the epochs and that signals that the predicted pointclouds are getting closer to the ground truth ones.



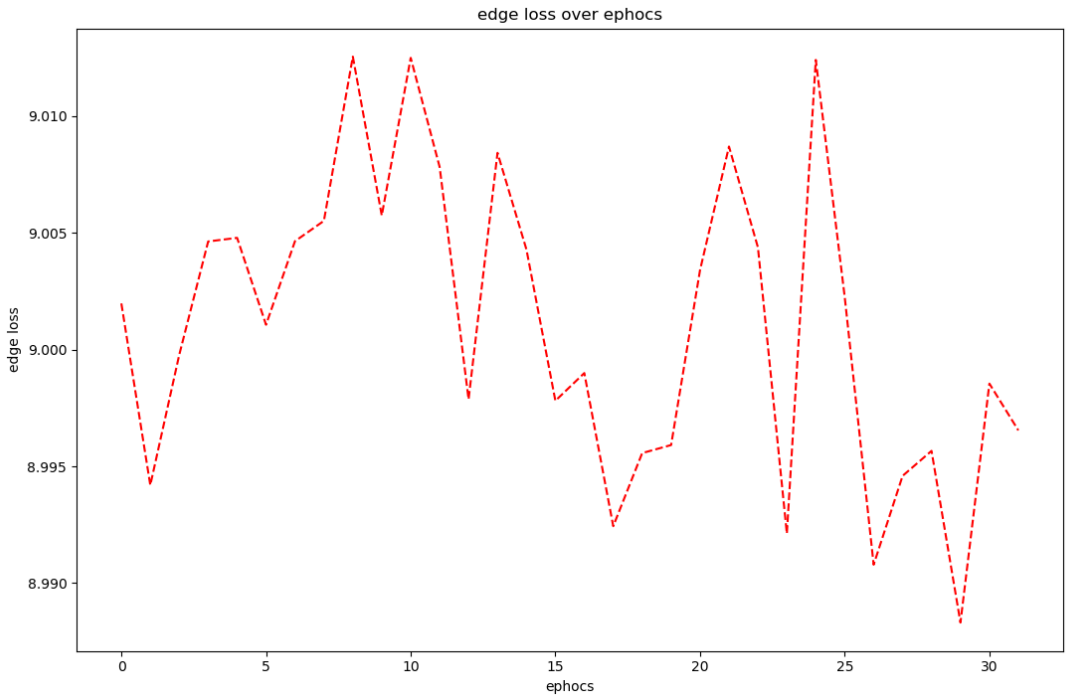
classifier loss over epochs – this loss decreases fast over the epochs but in our case it tells us nothing cause the model was trained over only one class.

Pix3D

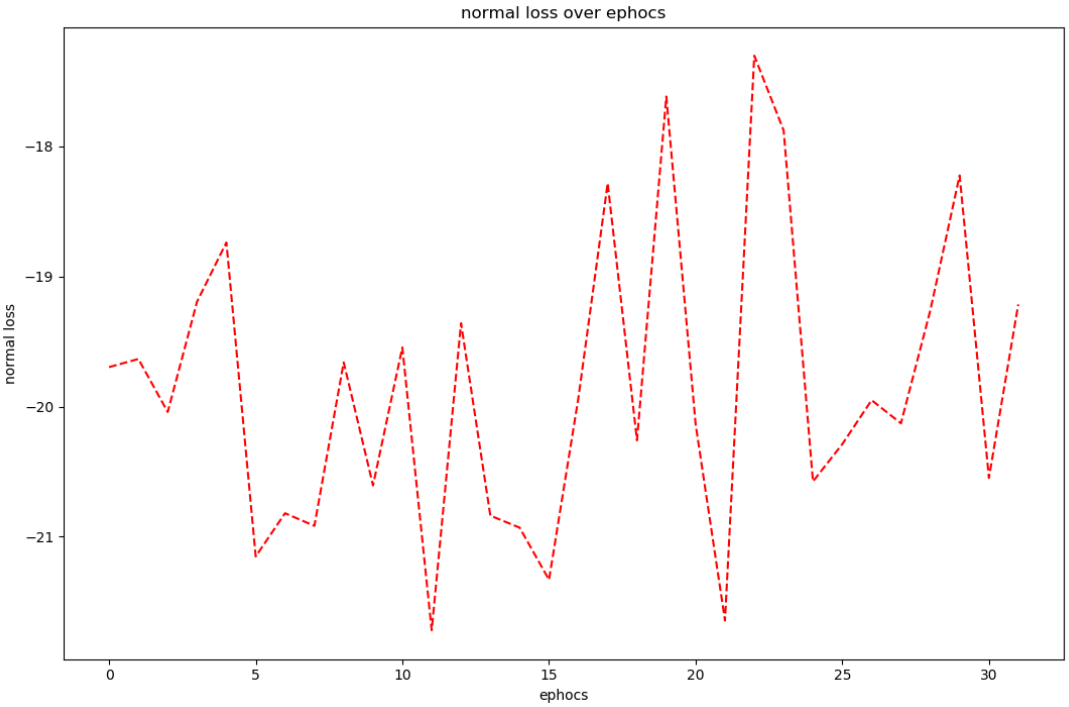
Pix3D model preformed less well then the ShapeNet model , the meshes he produces are not as accurate as they should be , but that can be explained because of the small batch size we had to use during his training because of memory constrains and because the time we had to train is relatively small to what it should have been, but mostly the complexity of the model that needs to operate on real world images not like ShapeNet that operates on rendered images, here a better backbone training on more vast size of examples effects the performance of the model greatly. As we can see in the loss graphs below from the model training the losses "bounce" around and don't have a certain trend, and that again is explained by the hard goal of this model that needs to preform on real world images with a very heavy and complex backbone that need way more training then we could give it in our time.



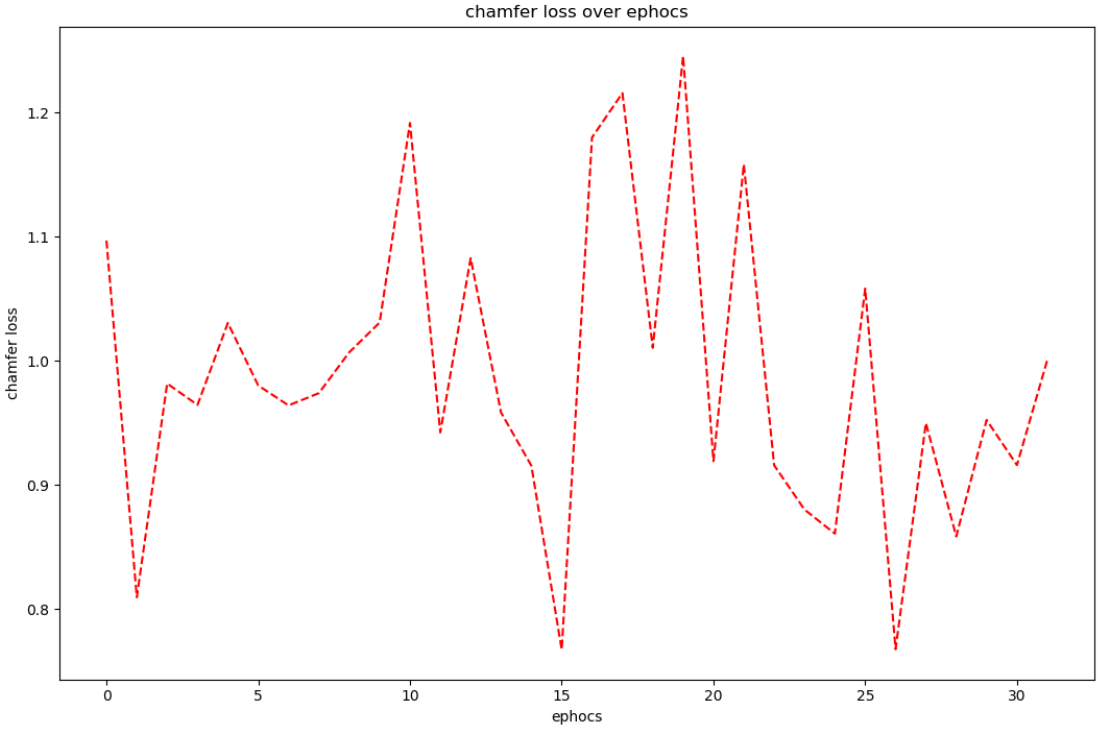
(voxel loss over epochs)



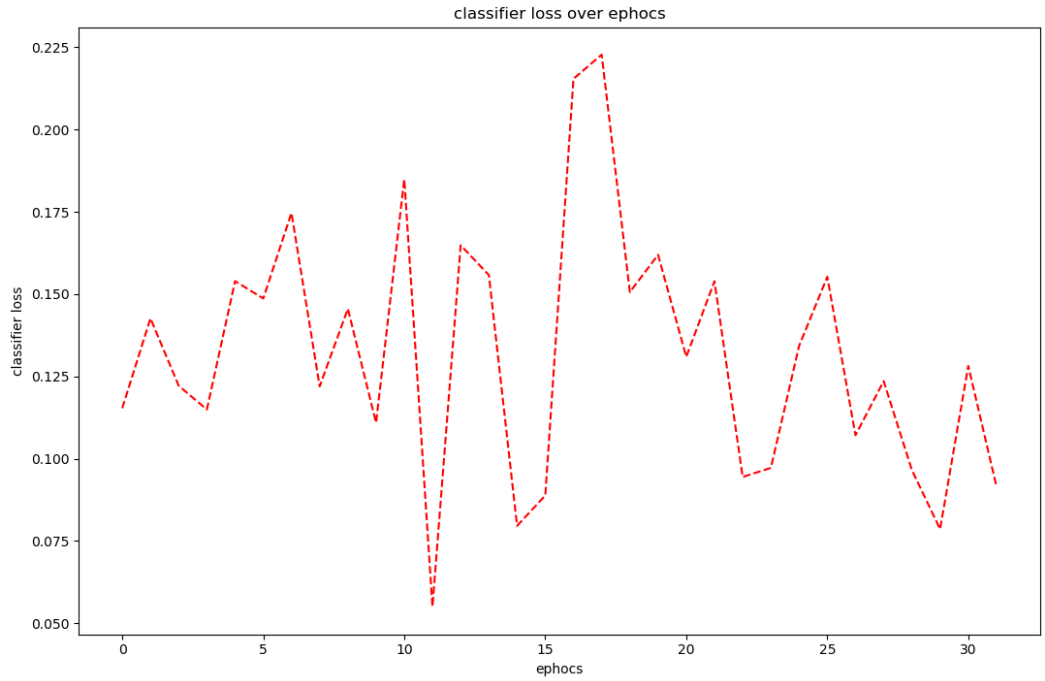
(edge loss over epochs)



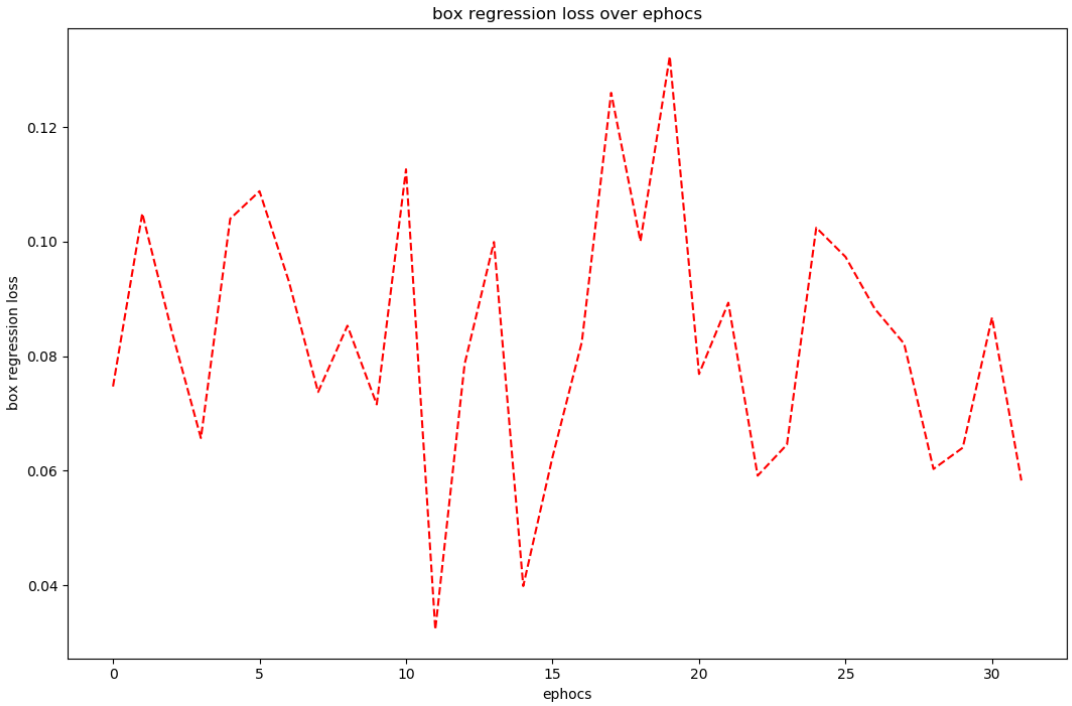
(normal loss over epochs)



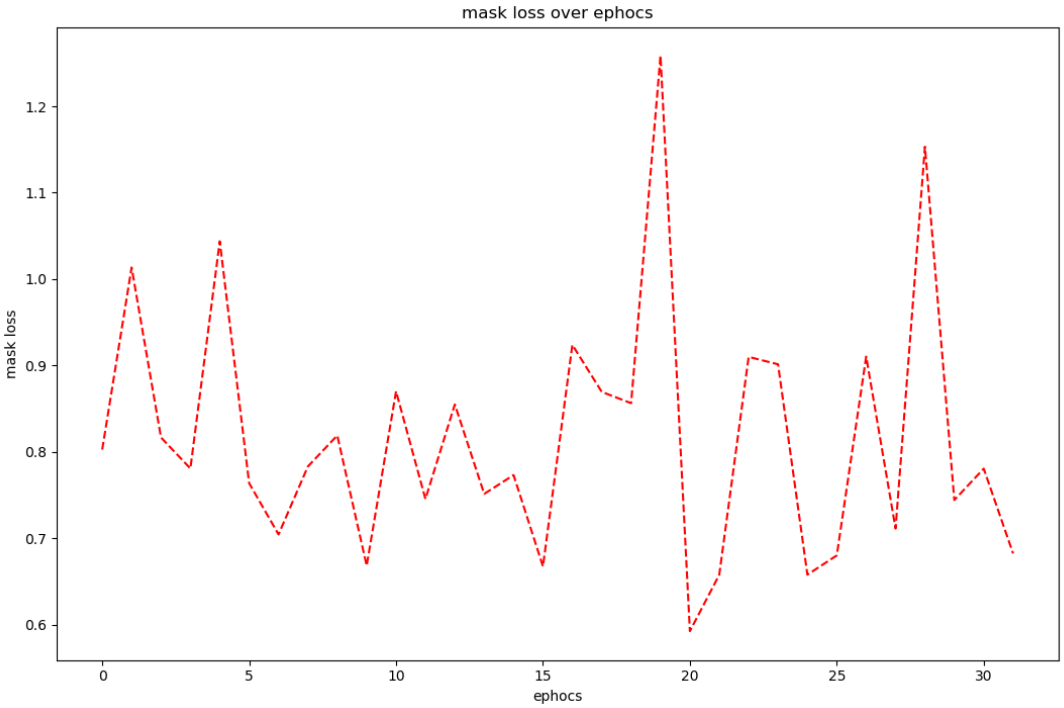
(chamfer loss over epochs)



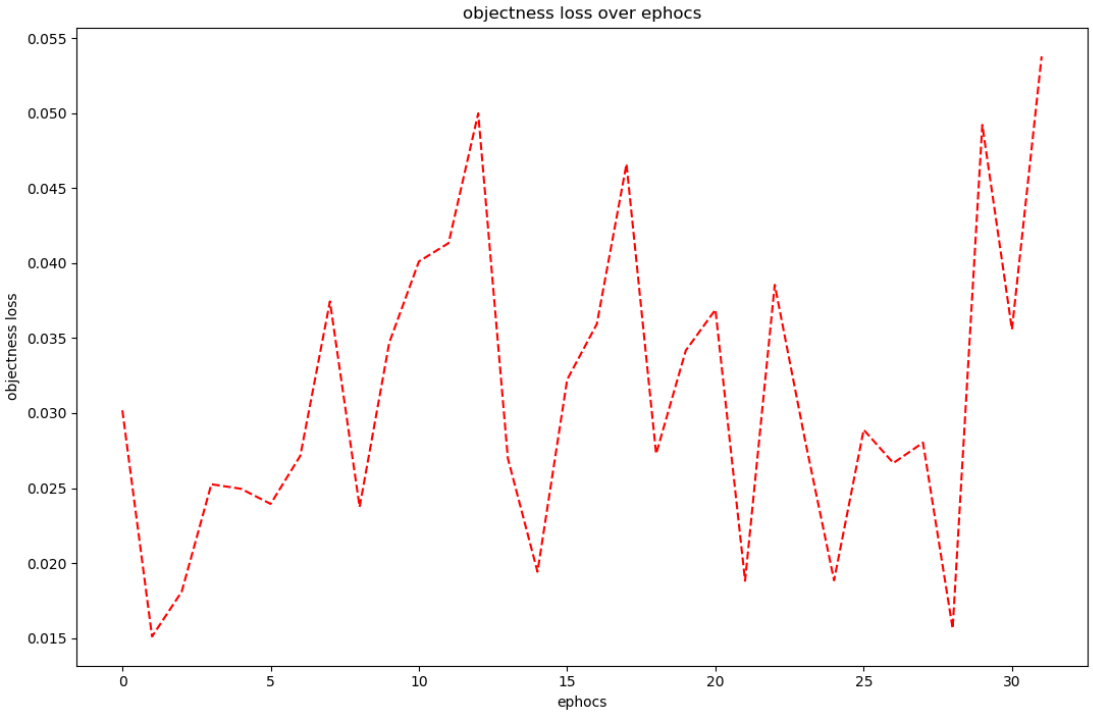
(classifier loss over epochs)



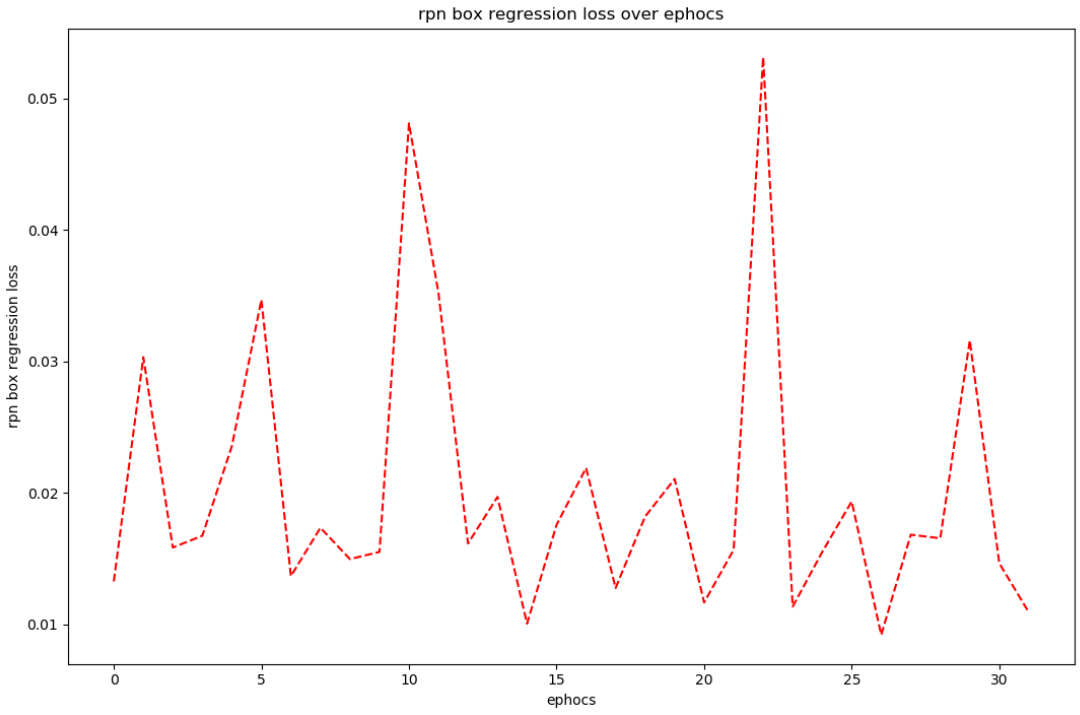
(bounding box regression loss over epochs)



(mask loss over epochs)



(objectness loss over epochs)



(rpn box regression loss over epochs)

<u>Conclusions and future improvements:</u>

because of wide spread server availability issues throughout the summer

we had to narrow our work and treat it as more of a proof of concept and implementation than as an accurate reproduction of the article.
the results that we have managed to obtain with our ShapeNet model lead us to believe that our work can achieve better results given more training time.
an obvious improvement is the generalization to multi class training