Advanced Topics for 3D Computer Vision Exercise 4 - 3D Deep Learning

Kagan Kücükaytekin
MSc. Robotics, Cognition & Intelligence
Technical University of Munich
Munich, Germany
kagan.kuecuekaytekin@tum.de

I. Introduction

When deep learning is mentioned, computer vision is the first application domain one can think of. Because of the widely available datasets and affordable computation, deep learning is widely applied in the 2D domain. For 2D, state of the art already started to mature, and we can see common architectures like U-Net, ResNet or VGGNet achieving great results in a wide variety of tasks. We see one common building block in all of these architectures: the convolutional layers (on a side note, transformers are also very successful, but they are not as widely used as the convolutional networks).

In contrast to 2D deep learning, 3D deep learning is still a growing domain, in which there is no consensus about "the best type of architecture". To make this problem even more complicated, there is also no consensus about "the best representation". A 3D environment can be represented in various ways, with the most popular ones being meshes, point clouds, voxels, surfels, signed distance fields and occupancy grids. All these different representations require different constraints on the neural network architectures. In this exercise I work with 3 types of representations: Point clouds, voxels (as plain occupancy grids) and spectral embeddings. I only consider the locations of the 3D geometries and do not consider color information.

II. Point Clouds

Point cloud is one of the most light-weight representations. Large environments can be represented using sparse point clouds. The biggest constraint in working with point clouds is the order independecy: the order at which the point clouds are processed should have no effect on the outcome. PointNet++ is the most widely used backbone for this data representation. It relies on furthest point sampling and grouping of points around these points to down-sample the data while extracting higher dimensional features. For this task, I have used a much simpler architecture with no grouping and with no extra sampling strategy. The architecture is inspired from T-Net; a building block of PointNet, the predecessor of PointNet++.

III. Voxels

Voxels can be considered as 3D versions of pixels. Many operations done on pixels have been generalized for voxels as well, such as convolutional operations. However, working with voxels is not as simple as working with pixels. 3D objects occupy a larger space and small convolutional filters usually fail to catch enough detail. For this reason advanced concepts such as sparse convolutions are used to cover a larger space. Voxel representations are also expensive to store, because n^3 voxels are needed to save a 3D geometry. Advanced methods such as voxel hashing or OcTrees try to address this issue, but they also add additional complexity in the pre-processing step. In this exercise I focus on simple occupacy grid voxel representation and use standard 3D convolutions to process this data.

IV. Spectral Embeddings

Spectral Embedding is a technique commonly used for non-linear dimensionality reduction. It relies on adjacency graphs and Laplacian Eigenmaps, which preserve the local geometry. As a result, the points that are close to each other remain close to each other in the embedding space representation. The adjacency graph helps to encode the interrelation between point neighborhoods to the embeddings. This property makes it a great tool for enriching the 3D data in the pre-processing step. For this exercise I use the spectral embeddings without reducing dimensions. I augment 3D point clouds with their 3D sepctral embeddings and process it with a similar neural network I used for the point cloud.

V. Fused Model

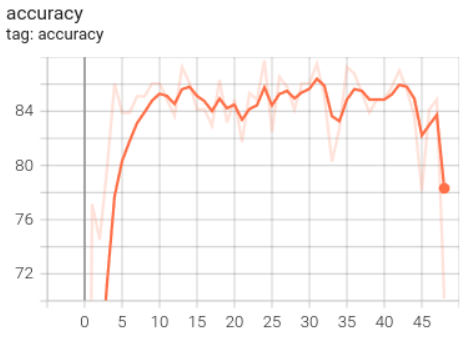
Finally, for the bonus part I fuse the models I used for point cloud and voxel processing. I combine the features extracted from the voxel network and the point cloud network using two fully connected layers. I also add dropout to prevent overfitting. Please refer to the code for the architecture.

VI. Comparison of Models

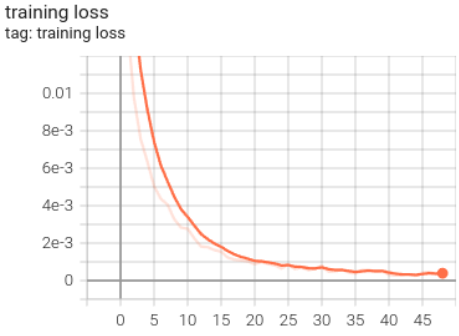
We can see that the voxelized architecture performs worse than the point cloud architecture. This is an expected result because useful information is disregarded during voxelization process. But as we can see from the results of the bonus, combining both modalities yields a more robust and better performing model. We do not observe overfitting here, but it is largely due to the lack of regularization in other models. Spectral embeddings help to increase the performance of the point-cloud network and increase its mean accuracy by about 2%, but its training was not as stable as other models.

VII. RESULTS

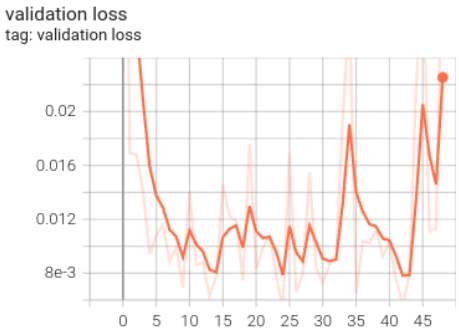
Below are the validation accuracy, training loss and validation loss curves of the models trained for the tasks. Results are listed from top to bottom in the following order: Point-cloud network, Voxel Network, Point Cloud Network with Spectral Embeddings, Fused Network composed of Point-Cloud and Voxel Network



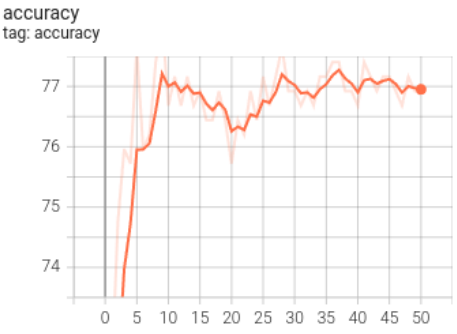
(a) Task 1:Accuracy



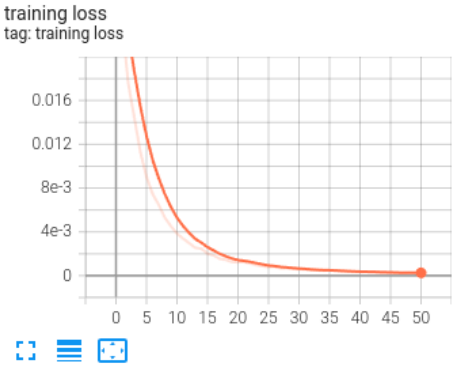
(b) Task 1:Training Loss



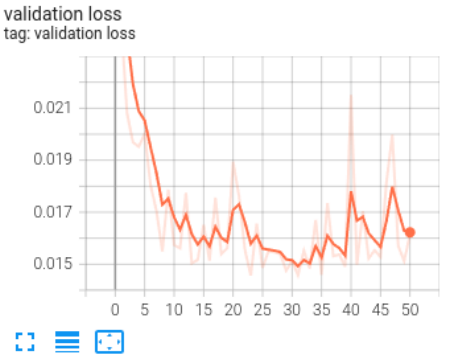
(c) Task 1:Validation Loss



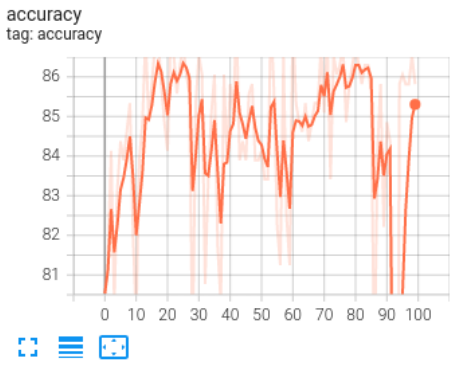
(d) Task 2:Accuracy



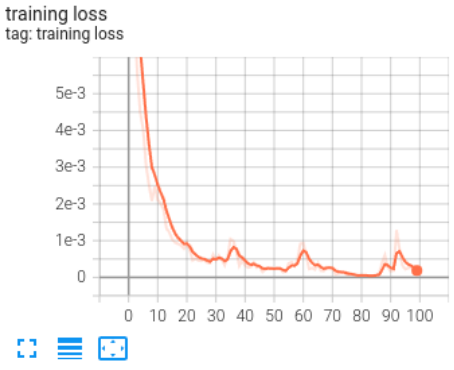
(e) Task 2:Training Loss



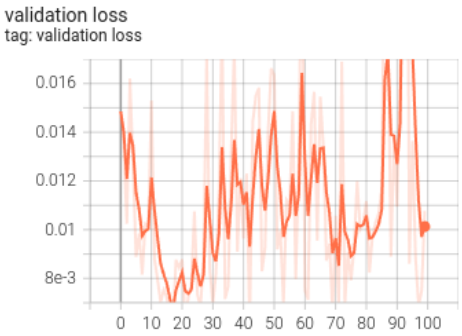
(f) Task 2:Validation Loss



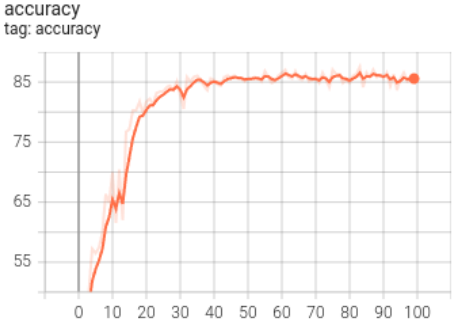
(g) Task 3:Accuracy



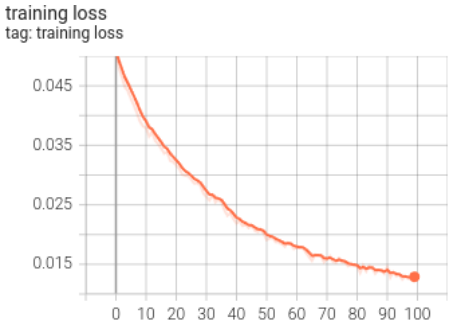
(h) Task 3:Training Loss



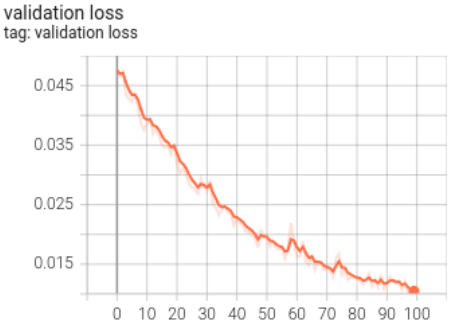
(i) Task 3:Validation Loss



(j) Bonus:Accuracy



(k) Bonus:Training Loss



(l) Bonus:Validation Loss