3D U-NET FOR PROSTATE MRI SEGMENTATION

Daniel Homola
dani.homola@gmail.com

1 Introduction

This mini project implements the fully convolutional 3D U-Net segmentation network [1] and train/evaluate it on the NCI-ISBI 2013 Challenge: Automated Segmentation of Prostate Structures. This dataset consists of 80 patients' 3D MRI scans from their prostate region. The model was implemented using core TensorFlow (i.e. without Keras) and Python. The main focus of this work was establishing preliminary benchmark performance along with developing a package that enables us to quickly iterate and try various model architectures (through hyperparameter tuning) and evaluation methods. Therefore, the project, in its current form is not intended to be a rigorous and thorough evaluation of the algorithm and it is merely a platform for future work and experimentation.

2 Data preparation

The dataset's train and leadership samples were combined to form a training set of 70 samples. The number of scans, their dimensions and various other relevant sample metrics were explored using a jupyter notebook. Several the scans and corresponding segmentation data were examined from several patients using tiled plots and animations.

Finally, the scans were preprocessed using the following steps:

- All MRI scans were rescaled to be between zero and one.
- Although the 3D U-Net is a fully convolutional network architecture (which is therefore dimension agnostic), the training scans were down-sampled to 128 x 128 size to reduce the required memory during training. The test dataset was not resized to ensure we measure test performance on the original resolution.
- To form batches of the data at training time, the number of scans (i.e. the depth of input tensors) had to be unified across patients. Therefore each patient's data was maximised to be no more than 32 scans. Then, at training time, patients with less scans are padded with zeros.
- An input size of [batch, 32, 128, 128, 1] was also desired as the network has shortcut connections between the analysis and synthesis paths, which necessitate compatible tensor dimensions at equal depths of the architecture. Maximum depth of 32 was chosen, as the network's three dimensional max-pooling and up-convolutional operations both shrink and expand the data by a factor of 2 respectively. Therefore, an input tensor dimension that is the power of two guarantees error free batches without masking.

3 Model architecture

Several crucial parameters of the model architecture can be easily changed in this implementation by creating a new params.json file. These parameters include: depth of the architecture, the inital number of filters to use whether to use batch normalisation or not

Due to the

4 Model performance

See awesome Table **??**.

	Mean IOU	
Model	Original	128 x 128
Base with BN	0.34	0.34
Base w/o BN	0.34	0.34
Deeper with BN	0.34	0.34

Table 1: Model performance

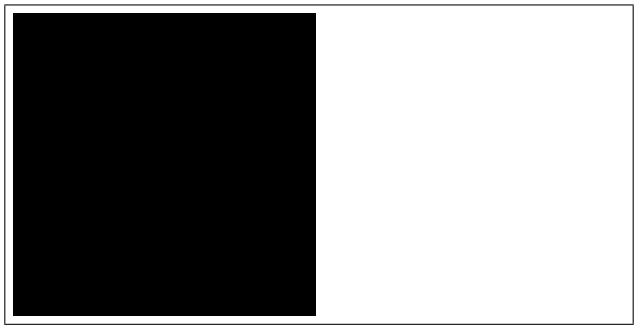

Figure 1: Sample figure caption.

4.1 Figures

See Figure 1. Here is how you add footnotes. [1]

5 Software engineering decision

6 Future work

6.1 Tables

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

References

[1] Ozgun Cicek et al. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”. In: *Lecture Notes in Computer Science* (2016), pp. 424–432. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46723-8_49. URL: http://dx.doi.org/10.1007/978-3-319-46723-8_49.

[1] Sample of the first footnote.