Extending 2D Deep Learning Architectures to 3D Image Segmentation Problems

Alberto Albiol[1(✉)], Antonio Albiol[1], and Francisco Albiol[2]

[1] iTeam, Universitat Politècnica de València, Valencia, Spain
{alalbiol,aalbiol}@iteam.upv.es
[2] Instituto de Física Corpuscular (IFIC), Universitat de València, Consejo Superior de Investigaciones Científicas, Valencia, Spain
kiko.albiol@ific.uv.es

Abstract. Several deep learning architectures are combined for brain tumor segmentation. All the architectures are inspired on recent 2D models where 2D convolution have been replaced by 3D convolutions. The key differences between the architectures are the size of the receptive field and the number of feature maps on the final layers. The obtained results are comparable to the top methods of previous Brats Challenges when median is use to average the results. Further investigation is still needed to analyze the outlier patients.

Keywords: Brain segmentation · Brats · 3D inception · 3D VGG · 3D densely connected · 3D Xception

1 Introduction

Brain tumor segmentation is an important problem which has received a considerable attention by the research community and particularly since the advent of deep learning.

Glial cells are the cause of gliomas that are the most common brain tumors. Gliomas are usually classified into low-grade gliomas (LGG) and high grade gliomas (HGG) which are malignant and more aggressive.

Brain tumors are usually imaged using several Magnetic Resonance (MR) sequences, such as T1-weighted, contrast enhanced T1-weighted (T1c), T2-weighted and Fluid Attenuation Inversion Recovery (FLAIR) images. From a pure pattern recognition point of view, these modalities provide complimentary information and can be used as different feature input maps. In other words, image modalities play a role similar to color planes of RGB natural images.

The Multimodal Brain Tumor Segmentation Challenge 2018 provided a set of MR sequences for training and evaluation of brain tumor segmentation algorithms. Ground truth for all the scans have been manually provided by expert

A. Crimi et al. (Eds.): BrainLes 2018, LNCS 11384, pp. 73–82, 2019.
https://doi.org/10.1007/978-3-030-11726-9_7

board-certified neuroradiologists, so that every voxel is categorized into these classes [11]:

- Label 0: background.
- Label 1: necrotic and non-enhancing tumor.
- Label 2: edema.
- Label 4: enhancing tumor.

We did not pay much attention on the medical details of this problem. Our main contribution was to extend some of the recent approaches used for 2D image classification: VGG, inception, Xception, densely connected models to be used with 3D data in a real segmentation problem.

2 Methods

Our approach uses an ensemble of deep neural networks with different architectures. The idea is that the ensemble provides a more robust solution with less variance compared to individual methods. Also, some architectures may compensate for other architectures weaknesses and thus improve the global performance. The idea of using an ensemble with multiple architectures was also used by the winning method of the last Brats competition [9].

This section describes the different architectures used in our approach. All the architectures have in common that every voxel is independently labeled using a deep neural network architecture. We are aware that better results could had been obtanied if some post processing that considered the spatial constraints had been used, similar to the CRF proposed in [10].

The key differences between the architectures are the number of parameters, the number of feature planes and the size of the receptive field associated to each voxel. These hyper-parameters were chosen as a trade-off usually limited by the memory of the GPU. More specifically, we mixed four different architectures in our final ensemble: VGG-Like, inception-2, inception-3 and densely connected. These models are described in detail in the following subsections.

2.1 VGG-like Model

This model is inspired on the well known VGG model proposed by [12]. The differences between our approach and the original VGG are:

- 2-D convolutions are replaced by 3-D convolutions.
- Maxpool layers are not used.
- The network is replicated in a convolutional way so that every pixel is labeled independently.

Table 1 describes in detail the layers used in this model. Note that all convolutional layers are preceded by batch normalization and followed by a ReLU activation function, except the last layer which is followed by a softmax activation function.

Table 1. Description of our VGG-like architecture.

Layer name	Kernel size	Num filters
conv_1_1	$3 \times 3 \times 3$	30
conv_1_2	$3 \times 3 \times 3$	30
conv_2_1	$3 \times 3 \times 3$	60
conv_2_2	$3 \times 3 \times 3$	60
conv_3_1	$3 \times 3 \times 3$	120
conv_3_2	$3 \times 3 \times 3$	120
conv_4_1	$3 \times 3 \times 3$	240
conv_4_2	$3 \times 3 \times 3$	240
fc_1	$1 \times 1 \times 1$	400
fc_2	$1 \times 1 \times 1$	200
logits	$1 \times 1 \times 1$	4

2.2 Dense-Like Model

This architecture is inspired by the recent work [8]. The key difference between the original method and the one used in this paper, is that 2D convolutions are replaced by 3D convolutions. The advantage of densely connected networks (compared to VGG like models) is that features are reused on subsequent layers and each layer adds a few new features only. This allows to increase the number of layers and therefore the size of the receptive field associated to each voxel. This architecture also allows to combine features with relatively small receptive fields (first layers) with features with large receptive fields (last layers). This is particularly useful in segmentation problems, where large receptive fields provide context information and small receptive fields provide fine-grained information that helps to increase the precision of the segmentation.

Table 2 summarizes the architecture of our densely connected network. Note that each layer concatenates all the output features from the previous layers, for this reason the number of input feature grows steadily until layer conv_20. Then two fully connected layers similar to the VGG architecture are used.

2.3 Inception-Like Model

This architecture is inspired by some of the ideas proposed in [14] and [13]. The key idea proposed by the inception model is to replace convolutional layers by several parallel structures with different kernel shapes. This reduces the number of parameters (regularization) and forces diversity on the output features of each layer.

We took these ideas and adapted them to the problem of brain segmentation. The main limitation of inception layers is that they require much GPU memory because each layer is composed of several simpler sub-layers, for instance some

Table 2. Description of our Dense-like architecture.

Layer name	Kernel size	Num inputs	Num output filters
conv_1	$3 \times 3 \times 3$	4	8
conv_2	$3 \times 3 \times 3$	12	8
conv_3	$3 \times 3 \times 3$	20	8
conv_4	$3 \times 3 \times 3$	28	8
conv_5	$3 \times 3 \times 3$	36	8
conv_6	$3 \times 3 \times 3$	44	8
conv_7	$3 \times 3 \times 3$	52	8
conv_8	$3 \times 3 \times 3$	60	8
conv_9	$3 \times 3 \times 3$	68	8
conv_10	$3 \times 3 \times 3$	76	8
conv_11	$3 \times 3 \times 3$	84	8
conv_12	$3 \times 3 \times 3$	92	8
conv_13	$3 \times 3 \times 3$	100	8
conv_14	$3 \times 3 \times 3$	108	8
conv_15	$3 \times 3 \times 3$	106	8
conv_16	$3 \times 3 \times 3$	114	8
conv_17	$3 \times 3 \times 3$	122	8
conv_18	$3 \times 3 \times 3$	130	8
conv_19	$3 \times 3 \times 3$	138	8
conv_20	$3 \times 3 \times 3$	146	8
fc_1	$1 \times 1 \times 1$	154	400
fc_2	$1 \times 1 \times 1$	400	200
logits	$1 \times 1 \times 1$	200	4

inception layers use 1-D convolutions along each spatial dimension. In the case of 2D convolutions, this option doubles the number of layers and the required memory used to store intermediate results and gradients. In the case of 3D segmentation, this problem is even worse because the use of 1-D convolutions implies to use three times more memory.

For this reason, we created two simplified GoogLenet-like models with a few inception layers before the fully connected layers as detailed in Table 3.

Figure 1 shows the internal structure of the inception layers. As it can be seen, four different branches are used. The first layer extracts new features and reduces the dimensionality. The second and third branches introduce spatial convolution; the fourth brach is an average layer without pooling. This structure is similar to the structure of Fig. 5 in [13].

Table 3. Description of the inception architectures used in the final ensemble.

Layer name	kernel size	num filters
conv_1_1	$3 \times 3 \times 3$	30
conv_1_2	$3 \times 3 \times 3$	30
conv_2_1	$3 \times 3 \times 3$	60
conv_2_2	$3 \times 3 \times 3$	60
conv_3_1	$3 \times 3 \times 3$	120
conv_3_2	$3 \times 3 \times 3$	120
inception	see. Fig. 1	240
inception	see. Fig. 1	240
fc_1	$1 \times 1 \times 1$	400
fc_2	$1 \times 1 \times 1$	200
logits	$1 \times 1 \times 1$	4

Inception2

Layer name	kernel size	num filters
conv_1_1	$3 \times 3 \times 3$	30
conv_1_2	$3 \times 3 \times 3$	30
conv_2_1	$3 \times 3 \times 3$	60
conv_2_2	$3 \times 3 \times 3$	60
conv_3_1	$3 \times 3 \times 3$	120
conv_3_2	$3 \times 3 \times 3$	120
inception	see. Fig. 1	240
inception	see. Fig. 1	240
inception	see. Fig. 1	240
fc_1	$1 \times 1 \times 1$	400
fc_2	$1 \times 1 \times 1$	200
logits	$1 \times 1 \times 1$	4

Inception 3

2.4 Other Architectures Not in the Final Ensemble

We also made experiments with other architectures not included in the final ensemble for their lower performance on our training data using cross-validation.

The most innovative structure in this group was based in the Xception architecture presented in [6]. This architecture assumes that correlation in feature planes can be decoupled from spatial correlation, and therefore separability is applied. We implemented this separable 3D spatial filters from scratch in Tensorflow (the library only provides this feature for 2D images).

We also made experiments with other inception architectures similar to those presented in Figs. 6 and 7 of [13]. However, the results on our cross-validated training set were not good enough.

The main limitation of these other inception architectures and also the Xception layers is that they require more GPU memory compared to the simpler VGG architecture, for this reason total number of layers needs to be reduced so that the model fits into memory. The main advantage of these architectures in 2D images is that they require a smaller number of parameters which help to regularize the model. However, we found that overfitting was not the problem for any of our models (the training cost and training error was not negligible), and therefore models with many parameters (as the VGG) could be trained without overfitting.

Finally, we also made experiments with field bias correction of the input data [15]. In these experiments, we corrected the bias of the T1 and T1ce input modalities and compared the performance without the field bias correction and the same neural network architecture. The results with the bias correction were always worse compared to using the original raw data with the same model architecture, and for this reason we omitted field bias correction.

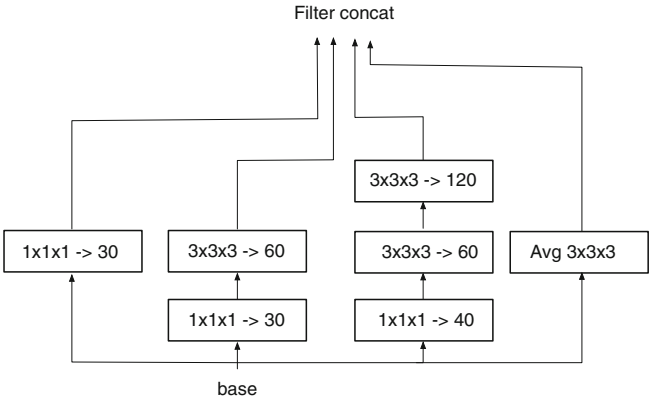

Fig. 1. Structure of the inception layers used in the models of Table 3. Each box shows the kernel size and the number of output features.

2.5 Number of Parameters and Receptive Field Size

Table 4 shows the number of parameters and receptive field size for the models included in our final ensemble. The model that requires more parameters is the VGG-like. This constraint limits the number of layers of the VGG model to avoid GPU memory problems. This is the reason why the VGG-like model has the smallest receptive field size.

The inception models halve the number of parameters (the latter layers are the ones with more parameters) and have a larger receptive field.

Finally, the densely connected model is the model with less parameters and largest receptive field.

The idea of our ensemble is to be able to combine models with large receptive field (more context), as the densely connected model, with very expressive models, i.e. models with many deep features (VGG-like) so that each model compensate for the weaknesses of the others.

Table 4. Number of parameters and receptive field size for the models used in our ensemble

Model	#parameters	Receptive field size
VGG-like	3270252	$17 \times 17 \times 17$
Inception2	1375872	$21 \times 21 \times 21$
Inception3	1611882	$25 \times 25 \times 25$
Densely connected	494220	$41 \times 41 \times 41$

3 Experiments and Results

3.1 Data

Our system was evaluated on the data from the Brain Tumor Segmentation Challenge 2018 (BRATS) [2–4,11]. As in previous editions, the training set consists of 210 cases with high grade glioma (HGG) and 75 cases with low grade glioma (LGG), for which manual segmentations are provided. The segmentations include the following tumor tissue labels: (1) necrotic core and non enhancing tumor, (2) edema, (4) enhancing core. Label 3 is not used. The validation set consists of 66 cases, both HGG and LGG but the grade is not revealed. For each subject, four MRI sequences are available, FLAIR, T1, T1 contrast enhanced (T1ce) and T2. The datasets are pre-processed by the organisers and provided as skull-stripped, registered to a common space and resampled to isotropic 1mm3 resolution. Dimensions of each volume are $240 \times 240 \times 155$.

3.2 Implementation Detais

We implemented everything in python. Input/output data for MRI scans was handled with the nibabel library [7] and neural networks were implemented using tensorflow [1]. The code used in this work has been dockerized and released to the challenge organizers so it will be available to the community.

We did not try any bias field correction of the input scans. The only intensity normalization that we used was z-score normalization of the input scans using the mean and standard deviation of the brain volume only (so the mean and std deviation are not dependent of the brain size).

Models were trained using crops of the original MRI scans. As in [10], the size of each crop was larger than the size of the receptive field. More specifically, the size of the crop is set $(9 + r_f) \times (9 + r_f) \times (9 + r_f)$, where r_f is the size of the receptive field. Thus, each crop contributes to the cost function with $9 \times 9 \times 9$ voxels. This approach increases the computational efficiency (reuses many computations) and we think that it also acts as a regularizer, forces the model to be smooth during labeling. For each mini batch, we increased the number of crops to fill the GPU memory (12Gb in our machine). These crops were randomly sampled using a uniform distribution among the four classes: healthy, oedema, core and enhancing core. During evaluation the size of the crops were increased and consecutive crops had some overlap to handle the reduced size of the network output (we used convolutions with only valid support).

Training was done using gradient descent with the Adam optimizer using a constant learning rate of 0.0001 for about 40k steps. We did not observed any overfitting during training, and for this reason we did not investigate into adding any L2, L1 regularization, learning rate decay.... Perhaps one of the reason why we did not observed overfitting is because we implemented a strong data augmentation that generated affine 3D transformations of the MRI scans on the fly.

3.3 Training Results

We split the training data in two random sets, so that one of the sets that contained 20% of the patients was used to evaluate the training progress. For each model architecture we generated two different training partitions using different random seeds, so that all training data was used by the models in the ensemble.

We ranked the model architectures using the Dice scores on our validation subset. Table 5 shows the Dice scores for our best models on our validation split. As it can be seen the differences among models are very small, however since the receptive field size and number of parameters is very different we think that the models might have captured complimentary information.

The last row in Table 6 shows the results on the Brats test set that we obtained on the challenge. The results on this set are clearly worse than those obtained for the validation set, this fact could be a clear symptom of some overfitting on the training and validation sets. However, we suspect that there could be also some differences due to other factors, such as different acquisition conditions because we did not made any model selection on the validation set and in that case we did not observed any difference with the results on our cross validation partition.

Table 5. Results of the selected model architectures on our validation split

Model name	Dice_WT	Dice_TC	Dice_ET
VGG-like	0.880	0.771	0.689
Inception2	0.882	0.792	0.685
Inception3	0.880	0.789	0.695
Densely connected	0.883	0.787	0.683

3.4 Results on the Validation and Test Sets

We submitted the predicted labels for each of the described models and also for the ensemble model for the validation set. There ensemble model averages the probabilities of 8 trained models (one for each architecture, and two random partitions of the training set).

Table 6 shows the results provided by the Brats evaluation platform on the blind validation dataset. The results are quite consistent with the results shown on Table 5, and hence we can conclude that we did not overfit the training dataset and the models generalize quite well on new data. However, the evaluation on the Brats platform shows an interesting point, median values of the Dice scores are much larger than the mean values. This confirms the existence of image outliers. The last row in Table 6 shows the results on the contest test set, the results for all other contest participants can be found in [5].

Table 6. Results of the selected model architectures on the validation set

Model name	Set	Mean Dice_WT	Mean Dice_TC	Mean Dice_ET	Median Dice_WT	Median Dice_TC	Median Dice_ET
VGG-like	Validation	0.872	0.760	0.751	0.900	0.837	0.844
Inception2	Validation	0.877	0.773	0.7533	0.909	0.866	0.858
Inception3	Validation	0.873	0.776	0.781	0.907	0.852	0.858
Densely connected	Validation	0.874	0.755	0.729	0.903	0.837	0.846
Ensemble	Validation	0.881	0.777	0.773	0.912	0.873	0.860
Ensemble	Test	0.850	0.740	0.723	0.894	0.856	0.828

4 Discussion and Conclusion

In this paper, we have extended some well known architectures for 2D image classification to the problem of 3D image segmentation. This can be easily done by replacing 2D convolutions by their 3D counterparts and adjusting the number of layers and number of feature maps to more appropriate ranges so that models can be fitted in memory.

We selected four model architectures so that we had models with large/small receptive fields, many/less parameters. The idea is that different configurations can capture complimentary information and an ensemble model can outperform each separate model.

The results on the validation set, show that there no exist many performance differences between the different model architectures, however the ensemble model outperforms each model. These results confirms our hypothesis and are also consistent with the results that we had previously obtained on the training data. The results on the Brats test set are clearly worse, we think that the cause of this behaviour is that there are some differences in the image acquisition and our method is not robust enough to deal with these variations.

We also tried other models, not included in the final ensemble, such as the 3D Xception that assumes independence between spatial and feature dimensions. We also tried to use bias field correction however our results showed that this was not useful for our models.

Finally, it is worth to highlight that the obtained results shows the existence of image outliers that are not well segmented. This issue severely drops our global performance as shown by the huge difference of using the mean or median metrics. We need to make further research on the causes of these outliers.

References

1. Abadi, M., et al.: Tensorflow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/, software available from tensorflow.org
2. Bakas, S., et al.: Segmentation labels and radiomic features for the pre-operative scans of the TCGA-LGG collection. The Cancer Imaging Archive (2017)
3. Bakas, S., et al.: Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features. Nat. Sci. Data **4**, 170117 (2017)
4. Bakas, S., et al.: Segmentation labels and radiomic features for the pre-operative scans of the TCGA-GBM collection. The Cancer Imaging Archive (2017)
5. Bakas, S., Reyes, M., Menze, B.: Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. In: arXiv preprint arXiv:1811.02629 (2018)
6. Chollet, F.: Xception: deep learning with depthwise separable convolutions (2016), cite arxiv:1610.02357
7. NIPY developers: nibabel 1.0.2, August 2016. https://doi.org/10.5281/zenodo.60861
8. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
9. Kamnitsas, K., et al.: Ensembles of multiple models and architectures for robust brain tumour segmentation. In: International MICCAI Brainlesion Workshop (2017)
10. Kamnitsas, K., et al.: Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Med. Image Anal. **36**, 61–78 (2017)
11. Menze, B.H., et al.: The multimodal brain tumor image segmentation benchmark (brats). IEEE Trans. Med. Imaging **34**(10), 1993–2024 (2015). https://doi.org/10.1109/TMI.2014.2377694
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: arXiv: 1409.1556 (2014)
13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826, June 2016. https://doi.org/10.1109/CVPR.2016.308
14. Szegedy, C., et al.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015), http://arxiv.org/abs/1409.4842
15. Tustison, N.J., et al.: N4ITK: improved N3 bias correction. IEEE Trans. Med. Imaging **29**(6), 1310–1320 (2010). https://doi.org/10.1109/tmi.2010.2046908