Cascade Neural Networks Algorithms for Segmentation in Medical Imaging

Iaroslav Bespalov Konstantin Gavrilchik Mikhail Sidorenko Nikita Kotelevskii Valeriya Pronina Ksenia Yagafarova Ekaterina Serkova

Abstract

We provide our solution in the context of the Medical Segmentation Decathlon challenge [1], which measures tumour segmentation performance in ten different samples for CT and MRI images of human organs. This solution can be found in our GitHub repository.

1. Project Description and Goals

Medical Decathlon is a contest held in 2018 devoted to semantic segmentation of medical images. The task of this challenge was to segment human organ and tumour/cancer on MRI or CT images of ten different human organs. For our project, we focused on working with the dataset for CT images of the pancreas, since it was the trickiest one for all of the participants (details in section 2).

The practical importance of our project is connected to the necessity of pancreatic cancer detection as early as possible as it is one of the most hardly treated cancers (survival rate is only 5% last few years [2]). Moreover, correct cancer localization requires high proficiency of the specialist. Even experienced specialist could use our solution, paying more attention to regions which were segmented by a model and was not selected by him/her.

The main goal of our work was to implement segmentation of 3D (CT) pancreas images to localise organ and tumour inside it. Our goal was to develop our segmentation algorithm, which is comparable with state-of-the-art (SOTA) architectures and achieves as high as possible position on the leaderboard.

2. Data Description and Analysis

The data we used in our experiments was published on the official competition website [3]. The data (training + testing) consist of 10 datasets for ten different human organs. These datasets were obtained via different methods: MRI and CT.

[1] http://medicaldecathlon.com/

[2] https://en.wikipedia.org/wiki/Pancreatic_cancer

[3] http://medicaldecathlon.com/

Since the competition was finished and the final results were published, we noticed that the most difficult (in terms of obtained scores), and therefore, the most challenging organ was Pancreas. That is why we decided to choose it as our prime goal to fight (fig. 1).

Figure 1. Example of a dataset description for the Pancreas Tumour

Our training dataset (Pancreas Tumour) consists of 282 3D CT images (fig. 2(a)) and masks with three classes - 'background', 'organ' and 'tumour' (fig. 2(b)). The dataset was difficult to achieve a high score due to several reasons. The first one is that the amount of examples is known to be less than enough to train a deep neural network from scratch. To fight the problem, we used augmentation a lot, which was one of the largest and important parts of our project. The second one - different patients' CT images have a different number of 2D slices (volumetric dimension), but to be able to feed batches with different images to network, images should have same shapes. Here our first preprocessing idea came. We decided to uniformly resample this 'volumetric' axis to have the same number of slices. To avoid a lot of slices being resampled without organ and/or tumour, the resampling was done only in the [0.2, 0.8] interval of the relative position of slices, where the major CT images have an organ and/or tumour according to histograms presented in fig. 3. Using statistical insights of our dataset (typical example's 'volumetric' length), we selected the resample constant

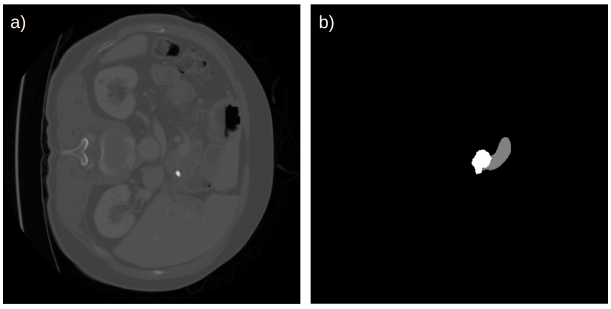

Figure 2. (a) One slice of CT 3D image, (b) 'organ' and 'tumour' segmentation

to be equal to 64. It means that each sample which has more than 64 slices will be downsampled, and upsampled in the opposite case. Here we sacrifice some information for the sake of simplicity of the algorithm. (fig. 4).

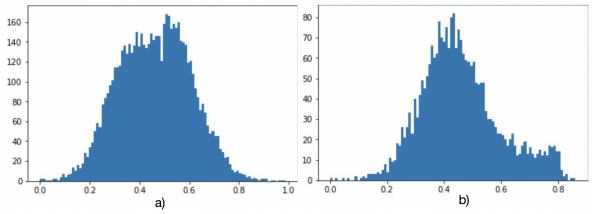

Figure 3. (a) Distribution of relative position of slices with organ, (b) 'Distribution of relative position of slices with organ and tumour

To know more about our data and aiming to find some helpful insights, we plot a histogram of pixel values. We figured out that it has two peaks. (fig. 5).

Trying to understand the nature of this bimodality, we replaced the negative peak by the highest (brightest) values on the picture, and we obtained the following images: (fig. 6). In the leftmost plot, different (consecutive) slices of initial CT image of one patient are represented. In the next column, there are masks of the corresponding images here yellow states for the organ, and blue states for the tumour. One can notice here how small the tumour is. The third column shows the result of replacing. It figures out that negative-most values were responsible for the tube of the CT equipment. The rightmost picture is just an assembling of the initial image and two obtained masks to form an RGB image. Let us now switch to the most important difficulty in our project, which is the class imbalance. The detailed data analysis (fig. 7) showed that the ratio of pixels/voxels which belong to the class 'organ' is only 0.0018, and to the 'tumour' class - 0.0003. The remain pixels/voxels belong to the background. To tackle this data imbalance, we tried different techniques, including modifications in neural network architecture, data preprocessing, robust loss functions, etc., which will be discussed in details in sections 3 and 4.

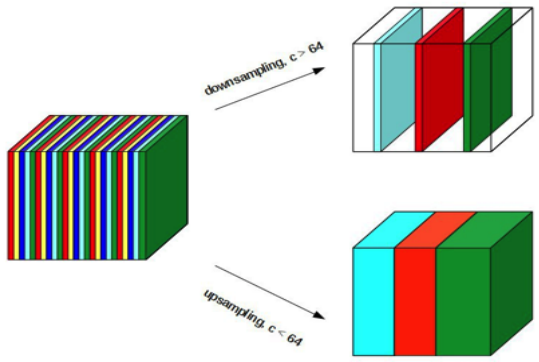

Figure 4. Scheme for channel resampling

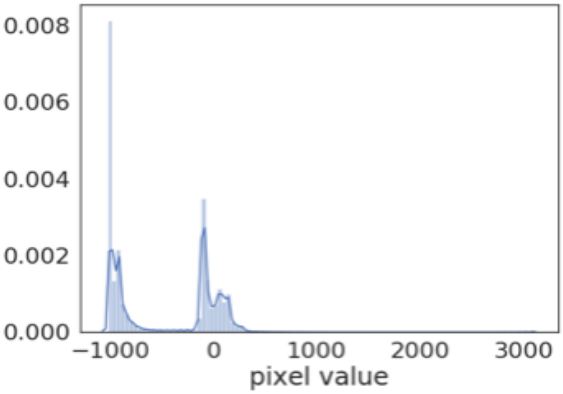

Figure 5. Diagram of the pixel values distribution

3. Methods

3.1. Network architectures

To handle the problem, we decided to start with the approaches which had already recommended themselves as a 'silver bullet' for biomedical segmentation tasks. Mainly, we started with a consideration of vanilla UNet architectures (both 2D and 3D) and then switched to their modifications. Let us firstly start with the description of the vanilla UNet, and then dig into details of our modifications.

Vanilla version

UNet is an encoder-decoder convolutional neural network, which was introduced specially for biomedical segmentation problems, mainly for cell segmentation in the original paper. Among other segmentation networks (which are encoder-decoder networks as well), the characteristic peculiarity of

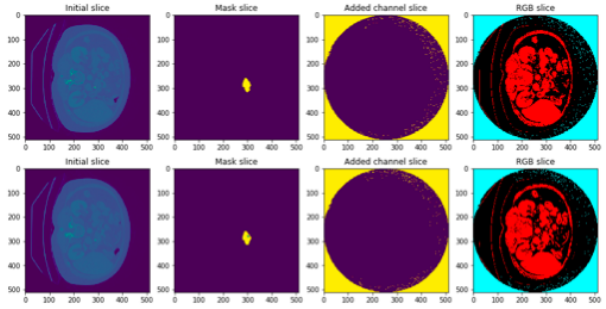

Figure 6. Visualization of pixels value distribution

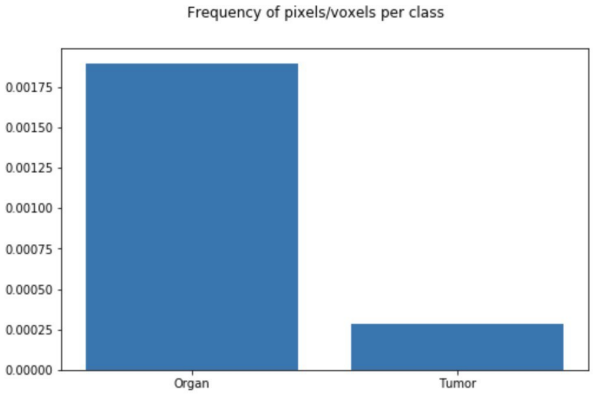

Figure 7. Frequencies of pixels/voxels per 'organ' and 'tumour' classes in the dataset

the network is the usage of crop-concat connections, which transfer information from encoder part (the first branch of the net) to the second one (the second branch is decoder) (fig. 8). Let us summarize the properties of UNet:

- UNet is symmetric in the sense that the encoder branch is almost the same as decoder branch.

- There are the skip connections between the downsampling path and the upsampling path, which apply a concatenation operator instead of a sum. Skip connections are used to combine low-level feature maps with higher-level ones, which enables precise pixel-level localization and transfers encoded different levels information to the final layers.

- For some reasons (probably connected with the architecture) it is known to perform very well on biomedical data.

UNet architecture is separated into three parts: downsampling path, bottleneck and upsampling path.

The downsampling path is constructed to capture the context of the input image. This contextual information will then be transferred to the upsampling path using skip connections. Then the purpose of the upsampling path is to enable precise localization combined with contextual information from the

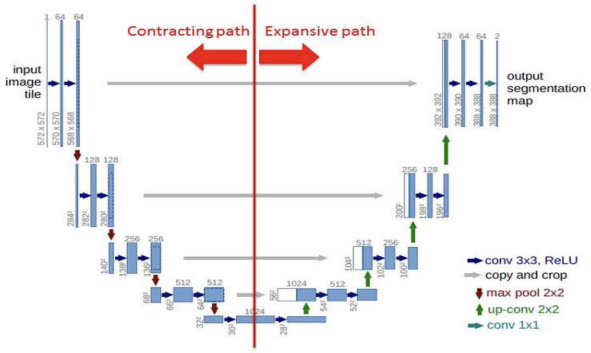

Figure 8. Scheme for UNet architecture (Ronneberger et al., 2015)

downsampling path.

2D and 3D UNet with ResNet34 encoder

Since UNet has encoder-decoder architecture, we can easily replace both encoder and decoder, preserving the feature with skip crop-concat connections. We implemented such modified UNet with ResNet34 network as the encoder and implemented these skip-connections as in the original network. The obtained results and corresponding reasonings are presented below. Also we tried to use a lot of different encoders such as densenet, resnet34, resnet50 and so ones. After a few iterations, we perceived that this is a bad approach for our task, because we have little data, and complex encoders lead to very fast overfitting.

2D and 3D UNet with attentions

Channel-wise attentions or so-called Squeeze-and-Excitation (SE) blocks are expected to provide better representations by explicitly modelling the interdependencies between the channels of the convolutional features of the network. The mechanism, proposed by (Hu et al., 2018), allows the network to perform feature recalibration and emphasize informative feature while suppressing less useful ones.

The structure of the SE block is presented on the fig. 9. The feature map **U** is passed through a squeeze operation, that is to produce a channel descriptor by aggregating feature maps across their spatial dimensions. In this case, with average pooling operation, one channel descriptor for one feature map was created, forming an embedding of the channel-wise feature responses. Then the squeeze operation was followed by the excitation operation. In this operation, the embedding from the previous step was passed through the simple self-gating mechanism, producing a set of per-channel weights. In the general approach, these weights are applied to the feature map **U**.

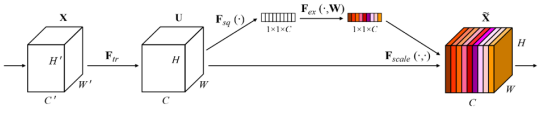

Figure 9. Squeeze-Excitation (channel-wise attention) block (Hu et al., 2018)

In the current work, we implemented 2D and 3D UNet architecture with the addition of SE blocks in the connection path (Fig. 10). The idea is the following: output from skip connection and the previous layer of the decoder part of the UNet are passed through the SE block, filtering both feature maps. After that, the results are summed and passed through the SE block again. Resulting weights are applied to the original output from the decoder layer and concatenated to the current decoder layer.

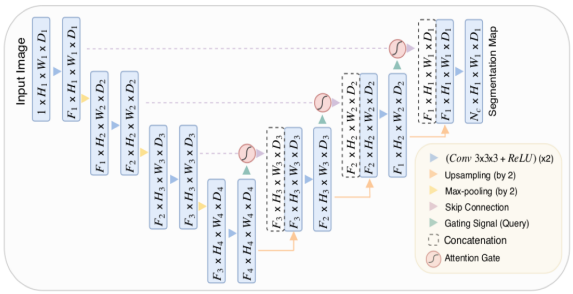

Figure 10. Block diagram of the Attention U-Net segmentation model (Oktay et al., 2018)

Another type of attention, spatial attention, was also implemented in the current work. The main difference of this type of attentions from SE is that instead of creating a channel descriptor, a feature descriptor is obtained with convolutions with the kernel of size 1. The scheme of the spatial attention block for 3D UNet is presented on fig. 11. Here g is the input from the connection and x^l is the input from the previous layer of the decoder. After the processing in the spatial attention block resulting weights are applied to the original output from the decoder layer and concatenated to the current decoder layer as showed on the fig. 10.

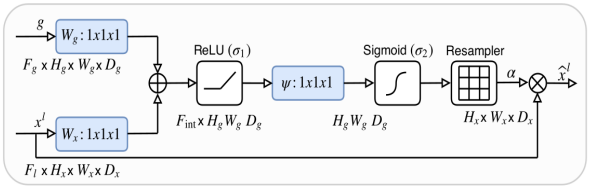

Figure 11. Block diagram of the Attention U-Net segmentation model (Oktay et al., 2018)

SE attention blocks and spatial attention blocks are expected to combine the most informative channel and spatial features from both global and local information to be passed further into the subsequent layers of the network.

UNet cascade

A cascade model is designed for datasets with large image sizes and aims to solve the data imbalance problem stated above. The idea of the cascade network is to use two networks independently and consistently (fig. 12). First of all, we perform a rough crop from the original CT image, where the organ is usually located (according to statistic throughout training dataset). Then these rough crops are fed to the first network, which is 3D vanilla UNet for segmentation 'organ' from 3D CT images. Then we perform cropping of segmented area from rough crops with some random shifting of 'organ' bounding box (our specific data augmentation) and feed these images to another 3D vanilla UNet for 'tumour' segmentation.

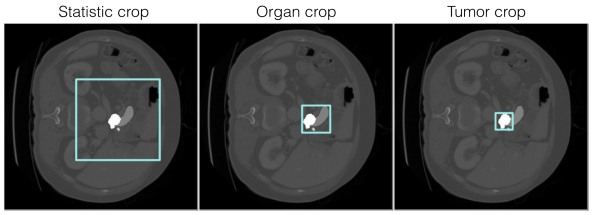

Figure 12. Scheme of cascade NN

Aside from vanilla UNet and its modifications mentioned above, we used some well-known SOTA segmentation algorithms as well:

3D Feature Pyramid Network

Feature Pyramid Network (FPN) (Lin, 2017) combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections. FPN riches semantics at all levels and is built quickly from a single input image scale, thereby without sacrificing representational power, speed, or memory. We adapted FPN for segmentation purpose from (Lin, 2017) for the 3D case. As a bottom-up path, we used the encoder as in our 3D UNet, then outputs on different layers were modified as shown on fig. 13.

2D PSPNet Well-known due to its performance, recently (2016) introduced PSPNet for semantic segmentation was also used and tested in our project. The main idea of the network is to consider a background context when predicting a segmented area. The textbook example is that two similar-looking objects (a boat and a car) usually have different background context: it is more probable to see a boat on water rather than a car. The context consideration is

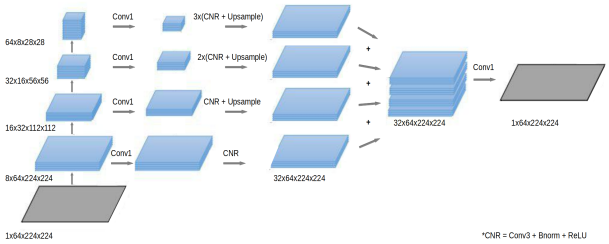

Figure 13. Scheme for channel resampling

taken into account via the usage of different average pooling kernel sizes. As it is shown in the fig. 14, using different kernels, we obtain feature maps of different spacial sizes, which means that we take a context from the background into account on different scales.

Our intuition is the following: we want to try a performance of 2D segmentation networks for 3D images, and we rely on the idea that the context is essential for 2D networks, which can only segment a 2D slice, but not a volumetric box at once. So we hope that the network will learn informative 'medical' interscale context (for example organ pairwise localization) and will take it into account on the inference. We use a pretrained ResNet34 as an encoder with trainable weights, because:

- it contains very good embeddings
- it does not require additional time for training the encoder

Schematic representation of PSPNet is presented on fig. 14.

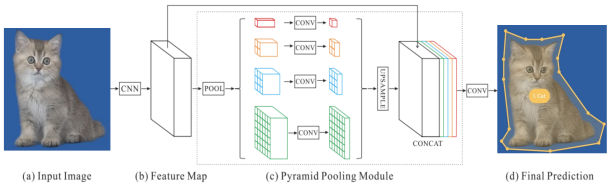

Figure 14. Scheme for channel resampling

All of the architectures mentioned above were implemented from scratch and are available in our GitHub repository.

3.2. Data preprocessing

Initially, all CT images had values in range [-1024, 3071] as shown in 5), so we convert all pixel/voxel values to [0, 255] interval, using the simple formula below (1).

$$\hat{X} = \frac{X - a}{b - a} \cdot 255, \tag{1}$$

where [a, b] is the original interval.

4. Training procedure

4.1. Augmentation

Augmentation played an important role in our project, as the data is quite small, and therefore, it was necessary to prevent overfitting. 3D samples were huge, so we reduced them by resize: initial images have spacial shapes 512 by 512, and the different number of slices in the volumetric axis. We resized spatial dimensions to be 224x224, and used uniform resampling for z-axis as it was described earlier (fig. 4). As follows from the description of the cascade, in order to balance the classes we cut out the area from the original image, after that we segmented organ and crop this mask. We implemented the cascade of cropping. Then we added augmentation: we shifted the cropped area randomly in a narrow interval along spatial axes. This trick greatly stabilized our training procedure, yielding both regularizing and augmenting effects. Additionally, we added horizontal flips (vertical flips have no physical sense in the problem, because, as we noticed, all patients (given samples) have the same orientations). Also, we used other augmentation techniques like the addition of Gaussian noise, elastic deformations, etc. However, it contaminated our score.

4.2. Learning rate

We use Adam optimizer with reducing learning rate on a plateau for all experiments (Fig. 15). The initial learning rate was 3×10^{-4} for starting training. Whenever validation's score did not improve by at least 15 epochs, the learning rate was reduced by factor 0.5. Also, we restricted the learning rate from below 1×10^{-6}.

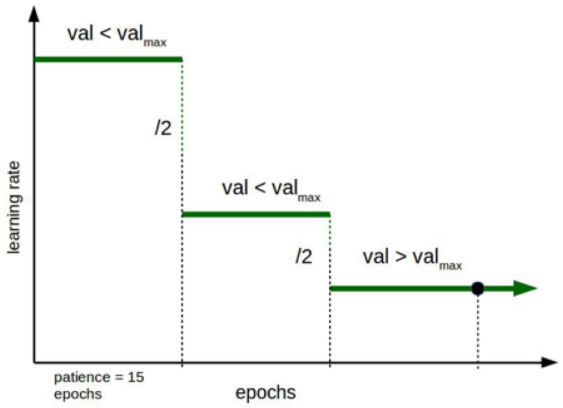

Figure 15. Learning rate scheduler

4.3. Loss functions

There are two main approaches for tackling class imbalance in medical images segmentation tasks: re-sampling of the data samples from the minor class (Havaei et al., 2017), (Lai., 2015) and deriving of appropriate and more robust loss functions (Brosch et al., 2015), (Milletari et al., 2016) , (Ronneberger et al., 2015). The main drawback of the first approach is the risk of reducing the variability in training. So we decided to concentrate on the second approach and make experiments with loss functions which are not sensitive to data imbalance.

$$L_{CCE}(y^{(i)}, \hat{y}^{(i)}) = -w^{(i)} \sum_{j=1}^{C} y_j^{(i)} \log \hat{y}_j^{(i)} \tag{2}$$

We started with Weighted Categorical Cross Entropy (2), which was also used in (Ronneberger et al., 2015). This loss function penalizes misclassifications of samples from different classes with different strength - samples from the minor class have higher weights, and it makes the cost of misclassification on these samples higher. Weight of a sample can be inversely proportional to the frequency of samples from the same class in a dataset or can be assigned manually depending on the cost of misclassification of objects from a class.

Another modification of standard Categorical Cross Entropy loss function is to combine it with Soft Dice Loss function (3). There are also some works where they used pure Soft Dice Loss without Cross Entropy term (Milletari et al., 2016), (Drozdzal & Michal, 2016). The authors claimed that the using of such loss function yields better results compared to Weighted Categorical Cross Entropy.

$$L(y^{(i)}, \hat{y}^{(i)}) = L_{CCE}(y^{(i)}, \hat{y}^{(i)}) + L_{Dice}(y^{(i)}, \hat{y}^{(i)}) \tag{3}$$

$$L_{Dice}(y^{(i)}, \hat{y}^{(i)}) = -\frac{1}{C} \frac{\sum_{j=1}^{C} \sum_{k=1}^{N} y_{kj}^{(i)} \hat{y}_{kj}^{(i)}}{\sum_{j=1}^{C} \sum_{k=1}^{N} (y_{kj}^{(i)} + \hat{y}_{kj}^{(i)})} \tag{4}$$

In (Sudre & Carole, 2017) they proposed to train a neural network for semantic segmentation problem using so called Generalized Dice Loss (5) as a loss function. This idea is similar to introduction the weights in Categorical Cross Entropy Loss.

$$L_{Dice}(y^{(i)}, \hat{y}^{(i)}) = -\frac{1}{C} \frac{\sum_{j=1}^{C} w_j \sum_{k=1}^{N} y_{kj}^{(i)} \hat{y}_{kj}^{(i)}}{\sum_{j=1}^{C} w_j \sum_{k=1}^{N} (y_{kj}^{(i)} + \hat{y}_{kj}^{(i)})}, \tag{5}$$

where $w_j = \frac{1}{(\sum_{k=1}^{N} y_{kj}^{(i)})^2 + \epsilon}$.

4.4. Evaluation metrics

To evaluate our models, we calculate the Dice score metric separately for organ and tumour as was proposed in Medical Decathlon challenge. General Dice score between two continuous sets A and B has the following form (6).

$$Dice = \frac{A \cap B}{|A| + |B|}, \tag{6}$$

where $|\cdot|$ is the cardinality of a set.

Due to the fact that reference and predicted masks are discrete, we use approximation of Dice score (6) as in many works (Havaei et al., 2017),(Brosch et al., 2015),(Ronneberger et al., 2015).

$$Dice = \frac{\sum_{i=1}^{N} p_i r_i}{\sum_{i=1}^{N} (p_i + r_i)}, \tag{7}$$

where p_i is predicted probability for i-th pixel/voxel to belong to the positive class, r_i is a reference mask, which contains ones for pixels from positive class and zeros otherwise.

4.5. Results and Discussion

We split segmentation task into two subtasks: organ segmentation and tumour segmentation, and used different network architectures, augmentations and losses for them. We tried out all losses described above, but we got decent results only for losses which include the cross-entropy term. Otherwise, our model was overfitted to the 'background' class, i.e. the dice score for the 'background' was equal approximately to 1 and 0 for 'organ' and 'tumour' classes. For all training procedures, we used learning rate scheduler, which adjusted learning rate, dividing it by a factor of 2, if dice metric value does not increase during 15 epochs on validation dataset (fig. 15).

2D UNet was trained for organ segmentation. Augmentations we used were random shifts of the box with localized organ (this localization was obtained by statistical analyze of our initial dataset), resize to 224x224 (spatial resolution), random horizontal flip and Gaussian noise with probability equal to 0.25. The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 16. We also tried to include deep supervision trick introduced in (Zhu et al.) in 2D UNet model, but it didn't give us considerable improvements.

2D PSPNet with pretrained ResNet34 encoder was also trained for organ segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224 (spatial resolution), random horizontal flip and adding of Gaussian noise (variance = 0.25). The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained

results can be found on fig. 17.

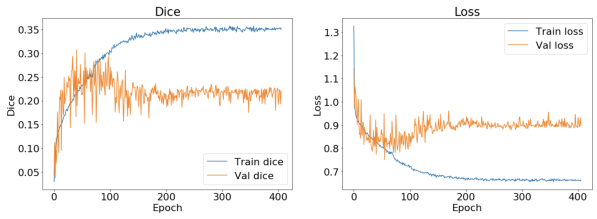

Figure 16. Dice metric and loss values for 2D UNet(organ)

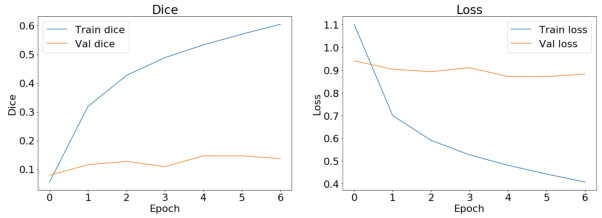

Figure 17. Dice metric and loss values for 2D PSPNet(organ)

However, the performance of the network is poor. It is probably connected with the usage of the pretrained model because there were no biomedical images in ImageNet. Since the process of 2D learning is much slower than 3D learning and since the fact that for 3D segmentation people usually use corresponding 3D models, we decided not to spent out time on it and switched to other methods.

2D UNet with SE blocks (Ronneberger et al., 2015) was trained for organ segmentation. Augmentations we used were random shifts of the box with localized organ, resize to 224x224 (spatial resolution). The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 18.

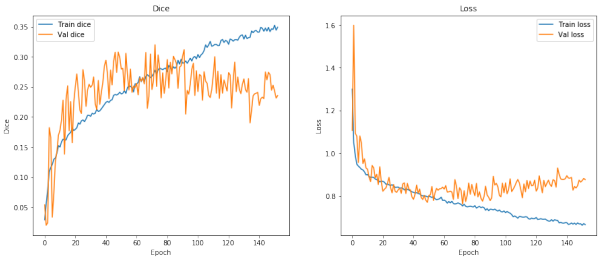

Figure 18. Dice metric and loss values for 2D UNet with SE (organ)

Adding of SE blocks into 2D UNet showed a slight improvement in both loss function and dice metric values, so we decided to continue with the implementation of this trick for 3D UNet.

3D Vanilla UNet was also trained for organ segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224 (spatial resolution), random horizontal flip. The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 19.

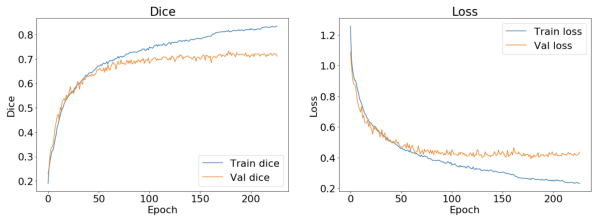

Figure 19. Dice metric and loss values for 3D Vanilla UNet(organ)

All implementations of 2D UNet with all modifications showed worse results than the implementation of even vanilla 3D UNet, so we decided to focus on the implementation of tricks for 3D UNet and not continue with attempts to improve results of 2D UNet.

3D UNet with SE (Ronneberger et al., 2015) was also trained for organ segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224 (spatial resolution), random horizontal flip. The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 19.

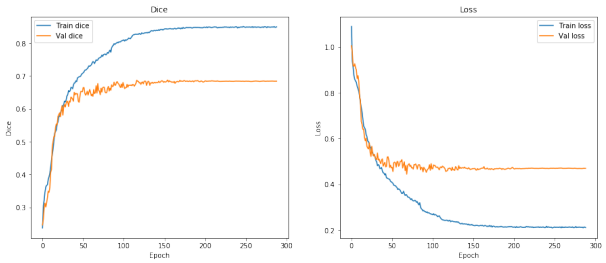

Figure 20. Dice metric and loss values for 3D UNet with SE (organ)

3D UNet with spatial attentions was also trained for organ segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224 (spatial resolution), random horizontal flip. The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 19. The results provided with UNet with spatial attentions showed worse loss decline than for UNet with SE, so it was decided to interrupt training and focus on other tricks.

As it can be seen from figures 19 and 20, although implementation of SE blocks provided better results for 2D UNet,

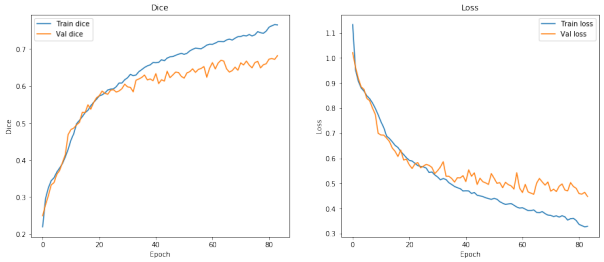

Figure 21. Dice metric and loss values for 3D UNet with spatial attentions (organ)

it showed worse results in organ segmentation for 3D UNet. This can be explained with the fact that when processing 2D slices independently, some of them provided better dice than others as they contained more information about organ. On the contrary 3D images were processed altogether analyzing information from all slices. Probably in the task of segmentation, there are not many features that can be selected with attentions for better segregation unlike, for example, tasks of face recognition. So this may be the reason of 3D vanilla UNet overperformance of the 3D UNet with SE.

3D Vanilla UNet was also trained for tumour segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224(spatial resolution), random horizontal flip and adding of Gaussian noise (variance = 0.25). The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 22.

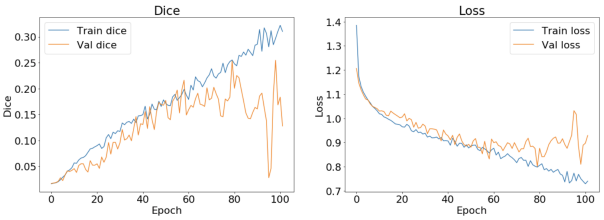

Figure 22. Dice metric and loss values for 3D Vanilla UNet(tumour)

3D UNet with SE (Ronneberger et al., 2015) was also trained for tumour segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224(spatial resolution), random horizontal flip and adding of Gaussian noise (variance = 0.25). The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 23.

Similarly to results for organ segmentation, 3D UNet with SE provided a similar result, but slightly worse than vanilla 3D UNet. It is more correct to say that the SE block does not improve the model.

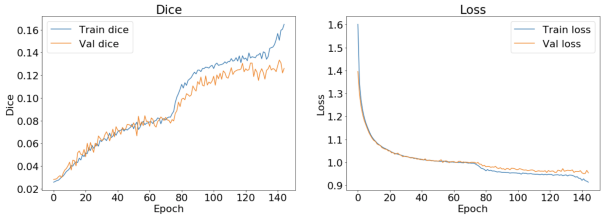

Figure 23. Dice metric and loss values for 3D Vanilla UNet with attentions(tumour)

3D FPN was trained for tumour segmentation. Augmentations we used were random shifts of the box with localised organ, resize to 224x224(spatial resolution), random horizontal flip. The loss function was Binary Cross Entropy(BCE)+DiceLoss. Obtained results can be found on fig. 24.

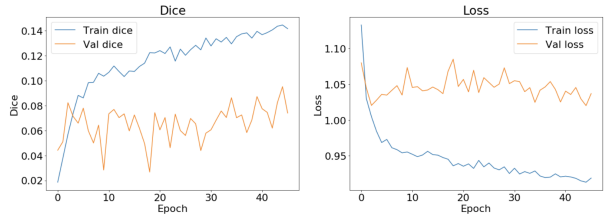

Figure 24. Dice metric and loss values for 3D FPN(tumour)

The predicted masks could be discontinuous and have unnatural form, therefore we decided to use some of the post-processing, which in our case is Conditional Random Fields (Lafferty et al., 2001), that allows us to avoid faults. This is a mathematical approach, and the result of can be seen on fig. 25, where CRF smoothed mask is more realistic and give additional 1.5% for dice metric.

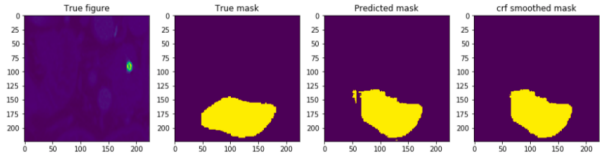

Figure 25. Learning rate scheduler

5. Conclusion

As a result, we got a cascade architecture that is different from the proposed in the articles. The competition medical decathlon **lasted for three months**, it was a challenge to start a project on segmentation of pancreas and pancreas cancer. The task involved a large number of problems at different stages. It is worth noting that we had only two weeks to work on the project and received a score corresponding

to the fifth place (see fig. 26). And during this time we have tried a large number of approaches and got a fairly good result.

№	Pancreas Dice	
	Organ	Cancer
1	0.80	0.52
5	0.74	0.25
11	0.56	0.15
19	0.48	0.04

Figure 26. Medical decathlon leaderboard with our results

References

Brosch, T., Yoo, Y., Tang, L., and et al. Deep convolutional encoder networks for multiple sclerosis lesion segmentation. *MICCAI*, pp. 3–11, 2015.

Drozdzal and Michal. The importance of skip connections in biomedical image segmentation. *eep Learning and Data Labeling for Medical Applications*, pp. 179–187, 2016.

Havaei, M., Davy, A., Warde-Farley, D., and et al. Brain tumor segmentation with deep neural networks. *Medical image analysis*, (35):18–31, 2017.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

Lai., M. Deep learning for medical image segmentation. 2015.

Lin, T. Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

Milletari, F., Navab, N., and Ahmadi, S. V-net: Fully convolutional neural net- works for volumetric medical image segmentation. *2016 Fourth International Conference on 3D Vision (3DV*, pp. 565–571, 2016.

Oktay, O., Schlemper, J., Folgoc, L. L., and et al. Attention u-net: learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

Ronneberger, O., Fisher, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, pp. 234–241, 2015.

Sudre and Carole, H. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pp. 240–248, 2017.

Zhu, Q., Du, B., Turkbey, B., Choyke, P. L., and Yan, P. Deeply-supervised cnn for prostate segmentation. In *2017 International Joint Conference on Neural Networks (IJCNN)*.

A. Contributions

During the project, the whole group worked in open space, so, if there were problems with some task, other guys could help, so the contribution to the project was about the same for all students. However, in any case, each student had their own tasks (a particular NN architecture also implies experiments with training procedures, like tuning the learning rate, batch size, different augmentations, etc.):

1) Konstantin Gavrilchik : Implemented the main part of the training pipeline, for which our team members adapted their models. Trained different encoders (Resnet, DenseNet). Proposed tricky approaches for cropping and non-classic augmentation.

2) Iaroslav Bespalov : Data analysis. 2D/3D Unet models. Setting the training process. Implementation Conditional Random Fields for masks postprocessing.

3) Nikita Kotelevskii : Data analysis. PSPNet. Debug of the cascade. Setting the training process. Parsing and save a lot of data and logs. Statistical approaches.

4) Ekaterina Serkova : implementation of augmentations and visualizations.

5) Valeriya Pronina : implementation of 2D/3D spatial and channel-wise attentions for our models.

6) Mikhail Sidorenko : Data analysis. Analysis of different loss functions, their implementation and testing with different models.

7) Ksenia Yagafarova : implementation of 3D FPN for our models.

B. Github links

Third party code usage listed in readme.md in our Git repository:

https://github.com/YaroslavBespalov/DL_project_nnUnet