U-Net and Transfer Learning Approaches to Liver Tumor Segmentation of MRI Images

Longlin Wang
wang-ll16@mails.tsinghua.edu.cn

Yurou Tang
tangyurou@gmail.com

January 2020

1 Introduction

1.1 Description of the Task and the Data

composed by Longlin

This reports describes our work on the liver tumor segmentation of MRI images, specifically on those of the artery phase. (The generalization to the images of other three phases is trivial.)

Automatic liver tumor segmentation is an emerging field of practical importance. According to the World Health Organization, liver cancer was the second most common cause of cancer-induced deaths in recent years. In addition, the liver is also a common site for secondary tumors. Therefore, a fast and accurate liver tumor segmentation applicable to medical images (CT and MRI images) would greatly help liver therapy planning and save lives from liver cancer.

However, the variability in the tumors' shape, appearance and localization has not only made manual segmentation challenging, but also impeded the easy application of some traditional machine learning algorithms, because no simple construction of features is available. It is with the application of convolutional neural networks, that such task began to have nice solutions. For example, in the 2017 LiTS challenge, all top scoring automatic methods used fully convolutional networks.

Previous work on liver tumor segmentation has largely focused on CT images, while MRI image segmentation is relatively less explored. CT is of lower cost and can be done fast, but with the disadvantage of worse soft tissue visibility and exposure to radiation. The abundance of public CT image datasets have spawned a large body of relevant models, but little has been done on MRI images, which is the gap this paper tries to fill.

A brief account must be given on the MRI images and the dataset of this project. MRI scan will generate 3-dimensional images of the patient's body, or, in other words, a series of 2-dimensional images stacked in spatial order. The main procedure of the MRI can be divided into 4 phases, among which the "artery phase" gives the brightest manifestation of the tumor. In this phase, the pre-injected contrast agent concentrates into the artery blood.

Because the tumor is rich in blood veins, the scan in this phase can distinguish it from the surrounding tissue because of the higher level of contrast agent in the tumor.

With the arterial phase images being our main focus, the dataset of this project contains 41 images of MRI scan, FZMC 1 $\sim$ 40+ FZMC 42. In the data cleaning, FZMC005, 006, 016 and 039 are removed because the dimension of the "artery.nii" and "label.nii" do not match, and FZMC039 is deleted because it has no "label.nii". Thus we are left with 36 images, each of shape $320 \times 260 \times 72$.

1.2 U-Net: a Neural Network Approach to Image Segmentation

composed jointly by Longlin and Yurou

Image segmentation is in some sense a combination of classification and image reconstruction. Intuitively, we would expect the network to first extract some features or properties from the raw image (2D or 3D), and use these features to reconstruct an image-shaped array, giving classification for each pixel (or voxel in a 3D image) of the image. Of course, segmentation differs slightly from classification, because one pixel may belong to several segments. For example, a pixel in the liver tumor may belong to both the liver segment and the tumor segment.

U-Net is one of the popular and successful ways of doing this task. It was first proposed in 2015 as a 2-dimensional image segmentation model (Ronneberger et al., 2015). It consists of a contracting path and an expansive path, much like an auto-encoder. The contracting path/the encoder is essentially a traditional convolutional neural network, while the expansive path/the decoder uses transpose convolution to perform up-sampling. The traditional design of the network, borrowed from the original paper of Ronneberger et al. (2015), is put below.

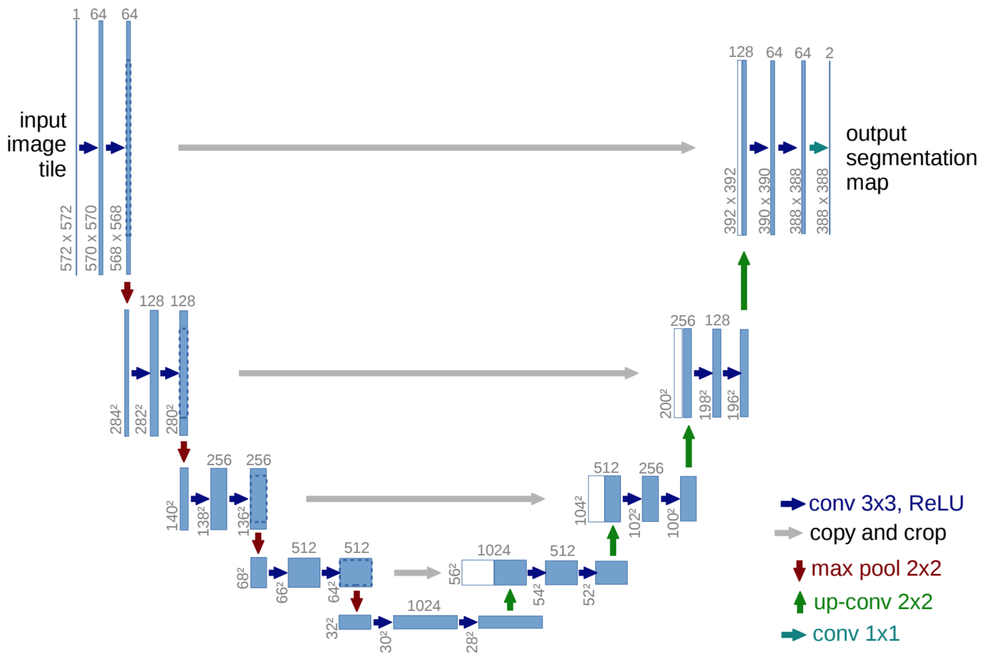

Figure 1: the original U-Net

One thing particularly interesting about U-Net is the use of “jump links”, which are indicated in Figure 1 with gray arrows. This partly explains the success of the network, as it increases the resolution of the segmentation.

To be more precise, the contracting path is the process of summarizing the information of the original image. The tensors flowing down the contracting path can be viewed to contain detailed and high-resolution information. At the very bottom of the “U”, the features in the input image is highly compressed. In the expansive path, the compressed information is used to expand the classification of the whole image, but the details may be lost, which is illustrated in the picture below. By using such links, the high-resolution features are again injected into the model, and boosts the performance.

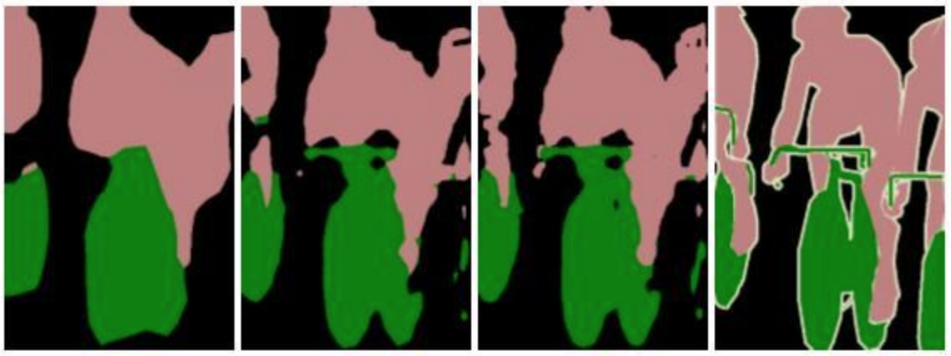

Figure 2: a picture originally depicting outputs of FCC networks. The original problem is about the design of the network to generate high-resolution segmentation, but the point is the importance of preserving details in the input.

The U-Net we built largely follows the traditional practice, while some compromise has to be made because of very limited computing resources and time.

We first traied the 3D U-Net, which can be seen as an extension to the original U-Net in 2D. Given that the given datasets are volumetric, 3D U-Nets would be a more appropriate choice of method than 2D U-Nets. However, we were faced with the challenge of limited computational power. A basic implementation of a downsized version of 3D U-Net (Çiçek et al., 2016) with just three layers and up to 256 channels easily requires more than 30Gb of memory. Faced with the difficulty of finding a machine with sufficient computational capacity, our final implementation looks at the performance of 2D U-Nets on volumetric data. We would expect the performance to be less than ideal, but it could still be worth taking a look at just how much 2D U-Nets can achieve.

1.3 Exploration of Transfer Learning in This Task

composed by Longlin

The attempt to use transfer in this project is actually inspired by the hint given in the slides. This part gives a brief introduction to transfer learning and how it is done in our project.

The origin of transfer learning is actually not so closely related to neural networks. The spirit of “transfer”, is to let the model pre-trained on source domain data to generalize well on target domain data. To put it more concrete, in our example, the source domain may be liver tumor CT images from the LiTS challenge, which contains 131 CT images, a much larger number than the size of our MRI dataset. We hope the model trained on the large CT dataset could bear some advantage over a model trained from scratch on MRI images.

The performance of transferring relies on finding domain-invariant features, which is in some sense the feature that can be generalized from the source domain to target domain. For example, the CT and MRI images are both scans of human body, and the layout of organs could be an invariant feature of the two models. However, MRI and CT contains different levels of details of soft tissue texture, and such details may not be easily generalized from the source domain to the target domain. The ultimate purpose of transfer learning, in general, is to let the model capture the invariant feature of the two domains, and achieve better performance on the target domain because the prior knowledge of source domain has been properly leveraged.

However, the conventional ways of doing transfer learning has some disadvantages. The common one is called catastrophic forgetting, which means that the pre-trained model trained on target domain actually forgets what it has learnt on the source domain, and is not different in performance with a model trained from scratch. Another concern is suggested in a recent paper (Raghu et al., 2019) specifically in terms of medical images. The paper suggests that despite the popularity of transfer learning in medical tasks, its performance is highly doubtful. The transferred models does not differ significantly with the models trained from zero, and the difference usually takes place with the model is really large, such as ResNets. However, such models are themselves over-parametrized for the usually small datasets of medical images, and thus this cannot be viewed as an advantage.

This part of the report tries to explore the performance of the transferred model trained on the MRI images. Despite the existence of more advanced transition training techniques (for example, progressive networks and the Bayesian approach in Kirkpatrick et al. (2016)), we stay with the traditional ones due to the limitation of time and computing resources. One model is first trained on LiST dataset, and then used as initialization to fit the MRI images, and the other model fixes the first two layers of the pre-trained model to avoid forgetting, and only trained the rest part of the model. Comparison is made among the pre-trained model, the model trained from scratch (normal initialzation), the model using the pre-trained as initialization, and the model using the pre-trained as initialization and fixing the first two layers.

2 Contributions of Each Author

This part talks about how the work in this project is acribed to Longlin Wang and Yurou Tang.

In the project, we planned to do two experiments. The first one was to implement the 3D U-Net and be trained on the MRI dataset from scratch to see its performance. The second one had a focus on transfer learning, and wished to compare the performance of naively trained model with the transfered model.

The first experiment is done by Yurou Tang. It was found that 3D U-Nets are too memory-consuming, far beyond the capacity of our computational resources. Due to the late beginning of this experiment, the switch to 2D U-Net was not succussful, and did not output anything meaningful by the deadline either.

The second experiment is done by Longlin Wang. This experiment used 2D U-Net from the start and yielded some interesting results. This report has to be based mainly on the analysis of this particular experiment about transfer learning.

In terms of the report writting, each of the two author finished the corresponding parts to their experiment, and other parts are generally joint work by Yurou and Longlin.

In terms of the code packaged with this report, only the code of the second experiment is submitted.

3 Methodology and results

3.1 3D/2D U-Net Trained on Artery-Phase Images

composed by Yurou, based on her work of the first experiment

This experiment originally used 3D U-Net, which did not work out due to its hugh memory comsumption. As a result, 2D U-Net was chosen instead of 3D. Due to time limitation, the network trained by Yurou herself had not worked out either, but an example of 2D U-Nets' preformance can be seen in Section 3.2 about transfer learning, in which one "control" group used a non-transferred 2D U-Net trained from random initialization to segment the MRI images.

We preprocessed the raw .nii datasets into arrays with dimension 36 * 320 * 260 * 72. In other words, there are a total of 36 sets of data, and for each dataset, there are 72 slices of 2D images of height 260 and width 320. To fit the given datasets in a 2D U-Net model, instead of performing convolution on each set of volumetric data, each slice of 2D image in the volumetric data is considered to be an independent data, and 2D convolution on each data. As such, for each volumetric dataset, there are 72 slices of independent data, with 36 volumetric datasets in total, we have a total of 2592 slices of 2D data images. Since each slice of 2D image data also has a corresponding label, it is possible to train a 2D U-Net based on the given data.

It is a pity that 3D U-Net was not tried, because we reasonably believe it would outperform 2D U-Nets on the given data. 2D convolution is essentially different from 3D convolution, in a sense that the relationship between different layers in a volumetric data is ignored. In other words, treating layers as independent data would mean that we have essentially abandoned the information of spatial correlation in the 3D images. In fact, it is very common for layers beside each other to share much similarity (since they are visually almost the same), and hence relationship between layers can definitely be utilized to produce more accurate results.

3.2 Transfer Learning Approach and its Performance

composed by Longlin, based on his work of the second experiment

3.2.1 A Closer Look at the Data

A brief account has been made about the MRI dataset we are provided with in this project, but this part also involves the use of LiTS dataset, which is much larger.

The raw LiTS dataset contains 130 pairs of .nii files, 'volume-(i)' the raw images and 'segmentation-(i)' the segmentation, each of which is the output of a CT scan. The images are all 512×512, unlike the 320×260 of the MRI images. Another important difference is on the third dimension. The CT images is of ragged shapes, unlike the MRI images which are all $320 \times 260 \times 72$. For example, "volume-0.nii" contains image of $512 \times 512 \times 75$, while "volume-4.nii" is $512 \times 512 \times 841$.

The further processing of the data requires the planning of the network architecture. Here we regret to say that due to the large size of the LiST dataset and the limitation of computing recourses, we can only opt for simpler models which can be trained within a few hours or one day. In our search for inspirations, one simple and successful model from a Nature paper caught our eyes (Chlebus et al., 2018). The network uses U-Net with fully convolutional architecture, and takes in only axial image slices of the model (namely, $512 \times 512 \times 1$ slices) and achieved state-of-the-art performance in the LiTS challenge. The network is small but powerful, and the model used in our experiment largely inherits the structure of this model.

Since we have decided on a model which takes in only slices of the 3D model, the source domain data is further processed to only preserve slices of the model, and ignore the spatial ordering of the slices in the 3D images. However, the dataset yielded in this way will still be enormous, and also has the problem of severe imbalance, because only a small percentage of the whole dataset contains the liver, and an even smaller percentage the tumor.

To tackle the problem of imbalance and the huge size of the data, we take a slightly different approach from the original paper (Chlebus et al., 2018). In each of the CT images, only a small proportion of all the axial slices contains the segment of the liver and the tumor. For example, in "volume-4.nii", which is $512 \times 512 \times 841$, only the 347th to 596th slices contains the liver. To ensure the generalizing ability of the model, reduce the imbalance and shrink the dataset, we extracted 30 slices from the slices with liver segment and the adjacent 20 slices without liver segment, used only the first 1100 slices.

One justification needs to be made here about the decision we have made to only use axial slices as input. Of course, the original 3D images contains the spatial information of the images. For example, for a single slice of the image, if the segmentation of the two adjacent slices are known, its own segmentation can be inferred easily. However, 3D networks also exerts great pressure on the computational resources. In the LiTS challenge, the most successful images used 2D or "2.5D" networks, rather than 3D images. ("2.5D" refers to the networks using only a few adjacent slices to segment the middle one.)

3.2.2 Architecture of the Model, Pre-Training Procedure

Figure 3 depicts the structure of the network adopted in our transfer learning experiment. It largely follows the design of the network in Chlebus et al. (2018), but with slightly shrunk numbers of channels to make the network trainable within a few hours. The output "may" have one or two channels, because in the LiTS task we want the model to do both liver seg-

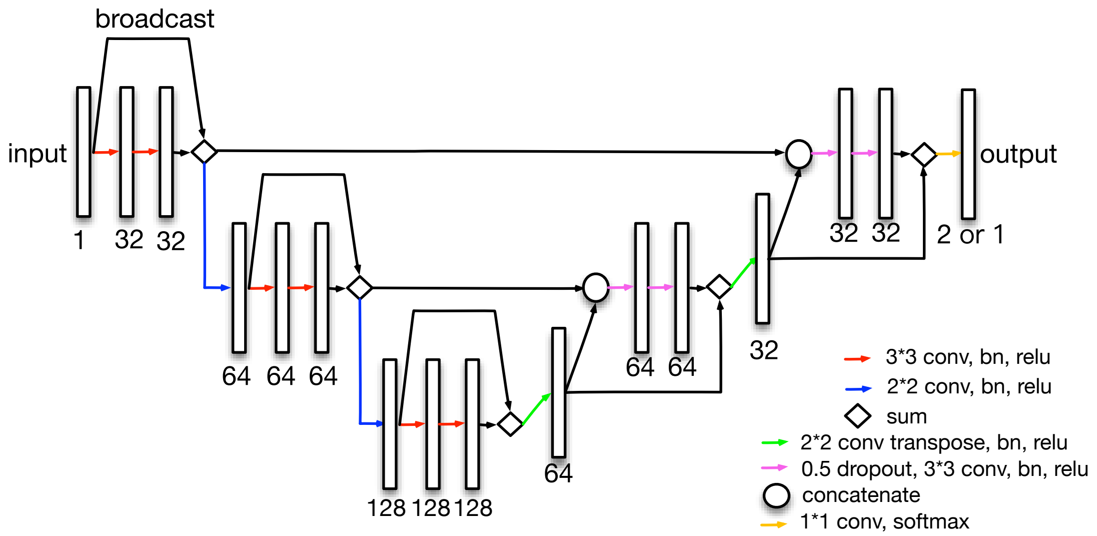

Figure 3: the architecture of my network

mentation and liver tumor segmentation, but in the target domain only tumor segmentation is needed.

Just like the traditional design of the U-Nets, our model has longskip connections passing the tensors from the contracting path to the expanding path to maintain the high-resolution details which are lost in the spatial downsampling. What is more, a new innovation is adopted from Chlebus et al. (2018) to use short skip connections to have better distributed parameter updates and to speed up training. Each convolutional layer uses 3×3 filter size and is followed by a batch normalization and a ReLU activation function. We used dropout ($p = 0.5$) before each convolution in the upscaling path to prevent the network from over overfitting.

The pre-training procedure is nothing more than fitting the model on to the LiST dataset we have just constructed. As has been said, the dataset contains 1100 slices of 512×512 images, with a balanced proportion of images containing liver segments and images without any segment. We used the last 100 observations as testing set and trained only on the first 1000 observations.

The choice of the data and the reporting of the training results both needs to strike a balance between accuracy and time efficiency, because we really don't have much computing resources and are unable to run a model for too long.

batch size is set to be 10. We tried among the possible values of $5, 10, 20$. When the batch size is small, the learning curve is volatile because of the randomness in drawing the small batch, while a large batch size can cause the training to be really slow, which is the case of setting *batch size* $= 20$. Too large a batch size also has the disadvantage of affacting the generalizing abilities, because the randomness of small batches can also prevent the gradient descend procedure to be trapped in a local minima.

Adam is chosen to be the optimizer because of its robustness of the choice of the parameters, and the *learning rate* is set to be 10^{-4}. The paper Chlebus et al. (2018) used the number $5 \cdot 10^{-5}$, which is our first choice. However, with this learning rate, the model seems to be learning too slow and only gain little information in each update.

The *numbers of epochs* is set to be 5, and in each of the epochs the whole dataset is looped over thoroughly. Actually, the real time learning curve is output for every iteration, and after five epochs we tend to believe that the model has largely stopped learning anything. The curve of testing and training loss almost coincides, which reflects almost no overfitting due to the use of dropout layers.

The *loss function* is chosen to be the BCE loss, whose implementation in pytorch is capable of computing multi-class classification error. We did investigate many different loss functions, such as the **dice loss**, and the **generalized dice loss**. However, the paper Sudre et al. (2017) which compared many different loss functions in this context shows no significant advantage of any particular loss function. In one of the author's blog, he gave the suggestion to try BCE loss as the primary option and then try out dice scores. Therefore we decided to stay with the conventional choice.

The testing error reported in real time is not computed over the whole testing set of size 100, but only on a random draw of size 10. However, this gives actually very stable results.

The learning curve is presented below:

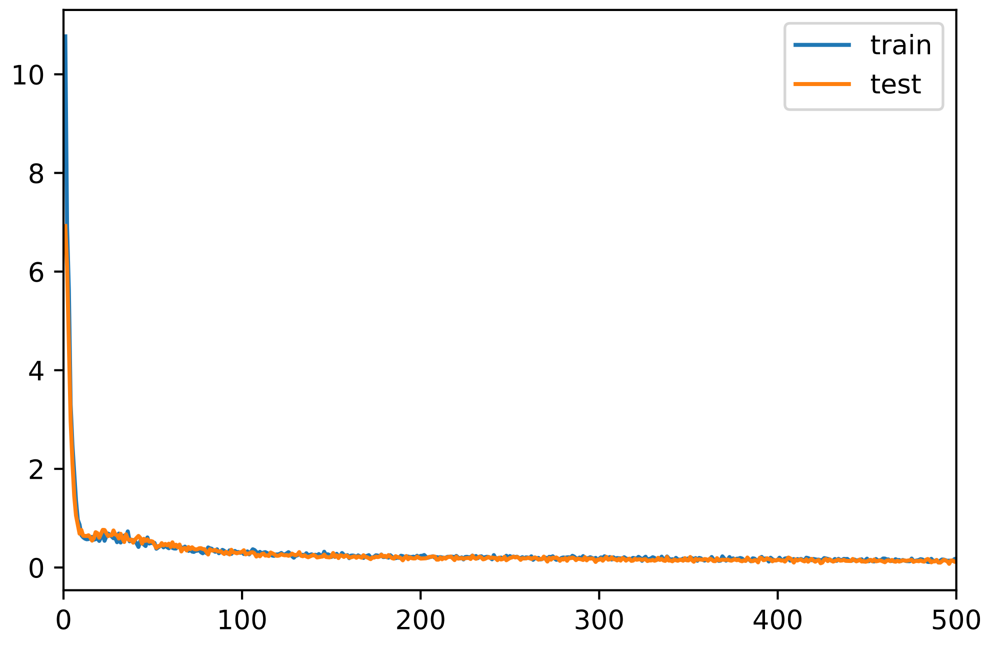

Figure 4: the architecture of my network

The network actually converges very fast and stays at a certain level of error. The time of the training and each epoch:

	epoch 1	epoch 2	epoch 3	epoch 4	epoch 5	whole process
time	23min	21min	20min	20min	21min	105min

Finally, we take a peek at the output of the pre-trained model on the source domain data in Figure 5:

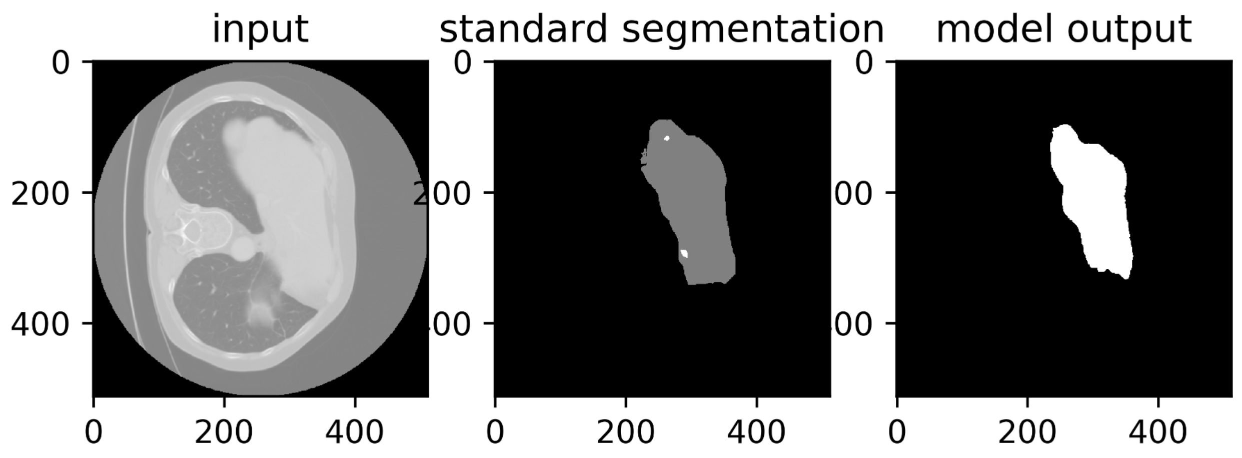

Figure 5: pre-trained model: output example 1

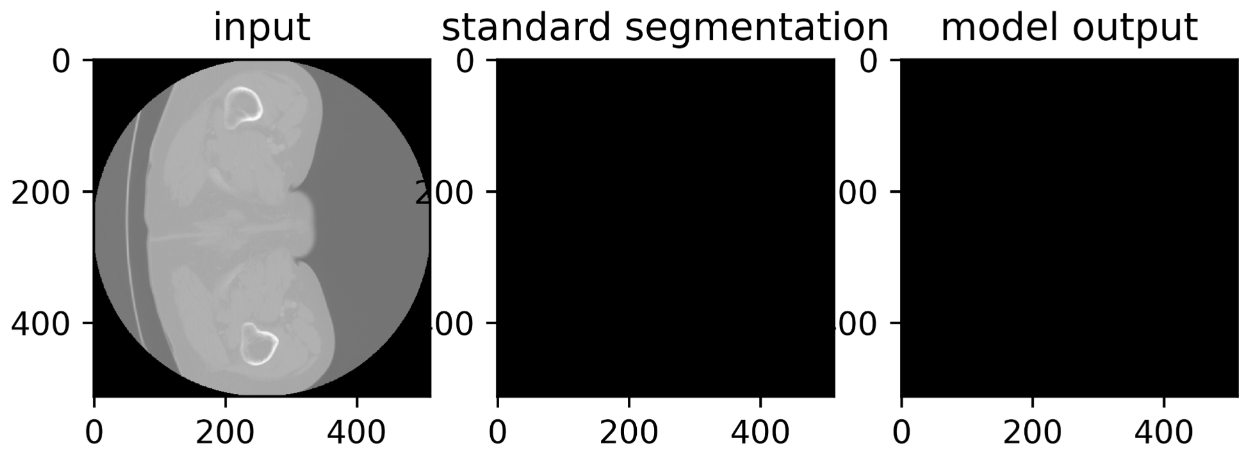

Figure 6: pre-trained model: output example 2

3.2.3 Training of the Transferred Models

The experiment of this report compared the following three models:

- (model A) the model of the above design and trained from scratch
- (model B) the model using the pre-trained model as initialization
- (model C) the model using the pre-trained model as initialization and fixing the first two layers of the pre-trained model.

(Note that the pre-trained model does both liver and tumor segmentation, while the MRI model only does the tumor segmentation. When using the pre-trained model as initialization, only the convolutional layer giving tumor segmentation is kept, and the other part is deleted.)

Before reporting the performance of these models, we first describe the training procedure of the three models. The choice of the parameters largely follows from those of the pre-trained model.

The dataset contains $72 \times 36 = 2592$ images in total, and the last four images (corresponding to $4 \times 72 = 288$ images) are used as testing set. The first $2592 - 288 = 2304$ images are processed with a batch size of 18. Each epoch loops over the entire training set, which account for 128 iterations. We set the number of epochs to 8.

The optimizer *adam* and learning rate $5 \cdot 10^{-4}$ is not changed. The loss function is still BCE loss.

3.2.4 Performance Comparison of the Transferred models

The experiment of this report compared the following three models:

- (model A) the model of the above design and trained from scratch
- (model B) the model using the pre-trained model as initialization
- (model C) the model using the pre-trained model as initialization and fixing the first two layers of the pre-trained model.

We first show the learning curves of the three networks:

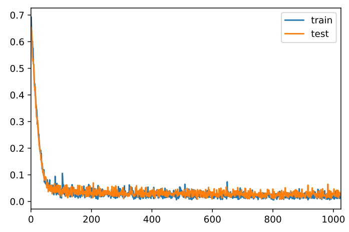

Figure 7: model A

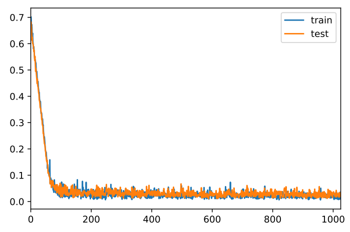

Figure 8: model B

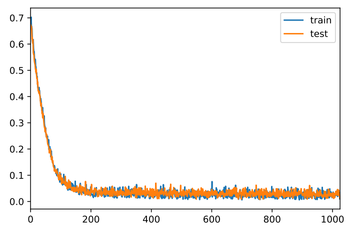

Figure 9: model C

Though the figures are somewhat small, it can be seen that none of them exhibits severe overfitting. Thus we make a clearer comparison by plotting the training loss in a single picture in Figure 10.

Viewing from the learning curve, it seems that the ultimate performance of the models are comparable. However, the curious phenomenon is that the convergence rate of the models decreases.

We defer the analysis of the curious learning curves to section 4, and move on to display the results of training time (in minutes).

	epoch 1	epoch 2	epoch 3	epoch 4	epoch 5	epoch 5	epoch 7	epoch 8	total
model A	14.25	14.74	15.02	14.31	14.30	14.26	14.10	14.17	114.93
model B	14.88	14.43	14.18	14.20	13.94	13.96	14.16	14.48	114.28
model C	12.56	12.20	12.13	12.23	12.15	12.21	12.24	12.25	90.01

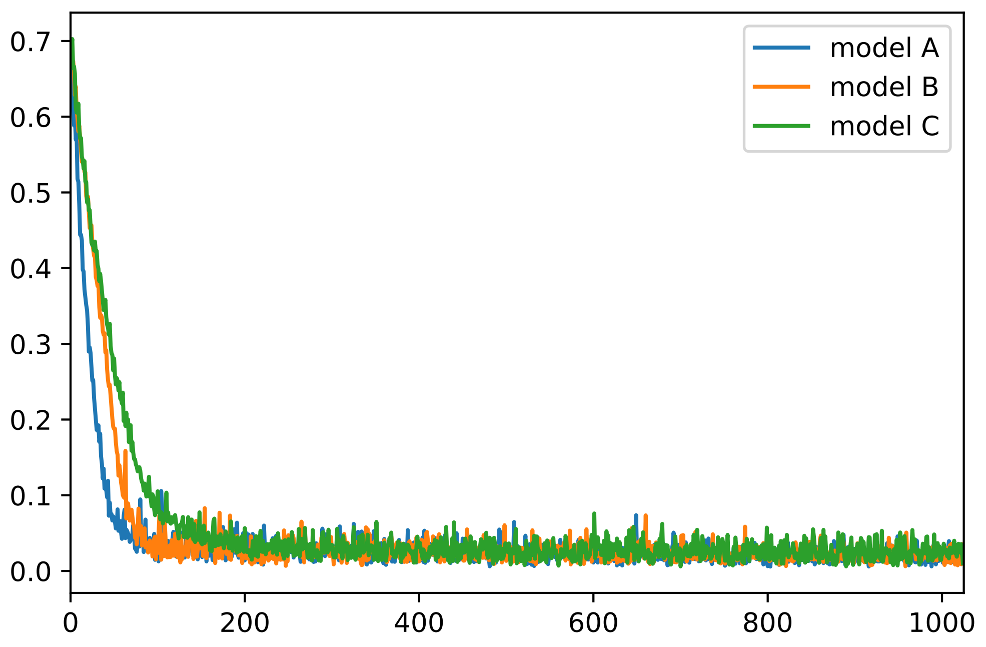

Figure 10: the learning curve of the three models

The result of training time is honestly not so informative, because the evaluation of the mini-batch, the back-propogation takes up basically the same time if the model's architecture is the same. In model C where the trainable parameters are actually fewer, the model naturally does the updates faster. No matter how fast the model does the computation, it has nothing to do with the outcome of training; this training time should not be confused with the rate of convergence of the model to some minima in gradient descend.

Finally we show some examples of the outputs of the three models (Figure 11, Figure 12, Figure 13).

4 Discussion

4.1 On the Use of 3D/2D U-Net

composed by Yurou based on her work of the first experiment

The performance of the 2D U-Net on the data given is probably less than satisfactory, but the experiment has really shone a light on the critical importance hardware has on machine learning.

In fact, the issue of computational intensiveness is particularly significant in the area of Medical Image Segmentation. Medical image data to be analyzed are often 3D, for instance images produced by MRI or CT, and medical images often have high resolutions to allow accurate medical analysis. Therefore, various researchers have proposed different methods

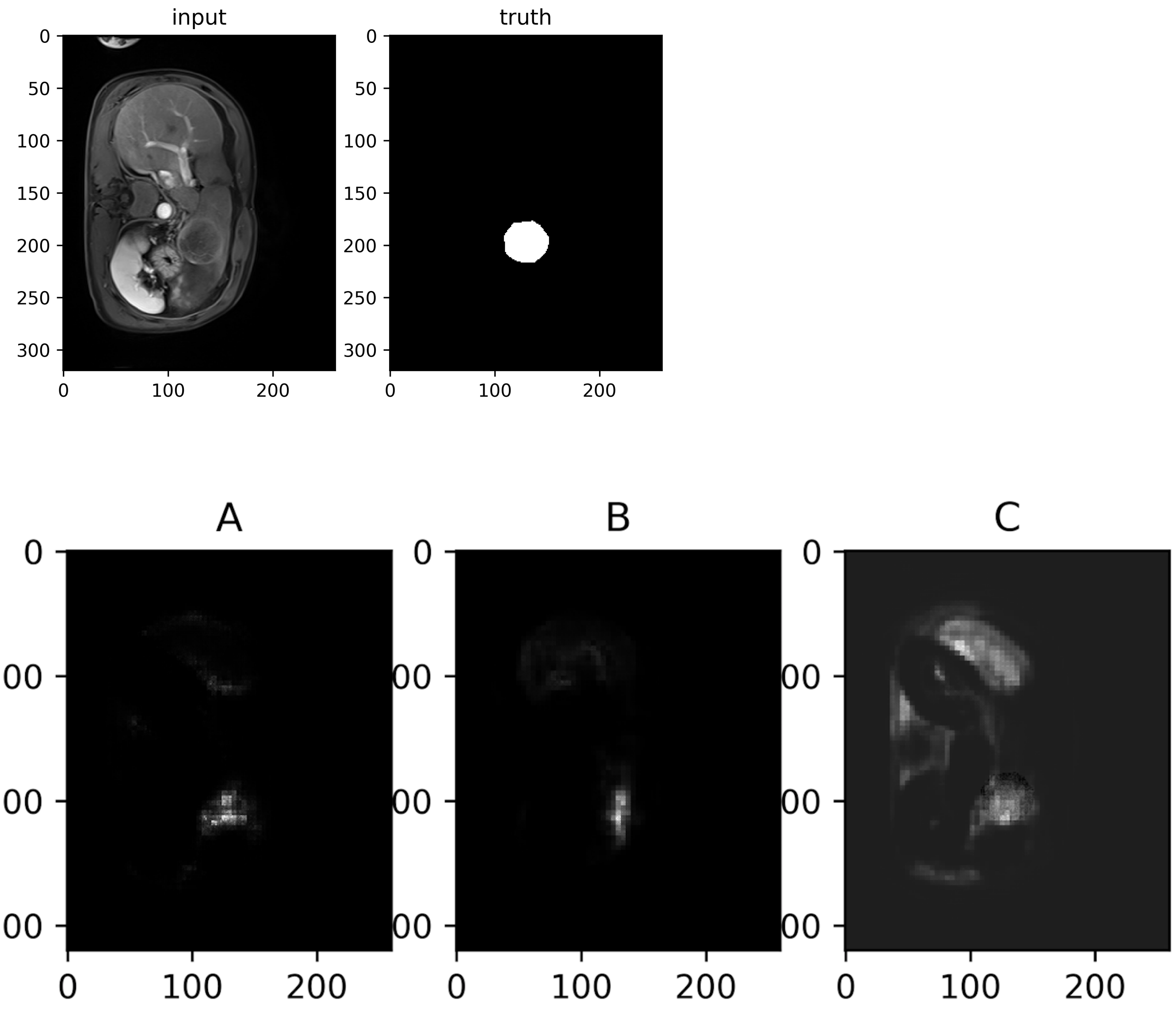

Figure 11: example 1 of the outputs of the three models and the ground truth

of reducing computational intensity of image segmentation. Hou et al. (2019) proposed a method of special partitioning which distributes the input and output of convolutional layers across various GPUs. It is seen as a better alternative to some other methods, including down-sampling and cropping, which could be dangerous in the field of medical imaging due to loss of data.

Other experiments on the U-Nets have shown more of its possibilities. For instance, the V-Net is a extension of the U-Net that is able to give better results by learning from residuals at various stages (Milletari et al., 2016). Moreover, the RCNN and RRCNN (Alom et al., 2018)seeks to tap on the advantages of both the Residual network and U-Net to produce

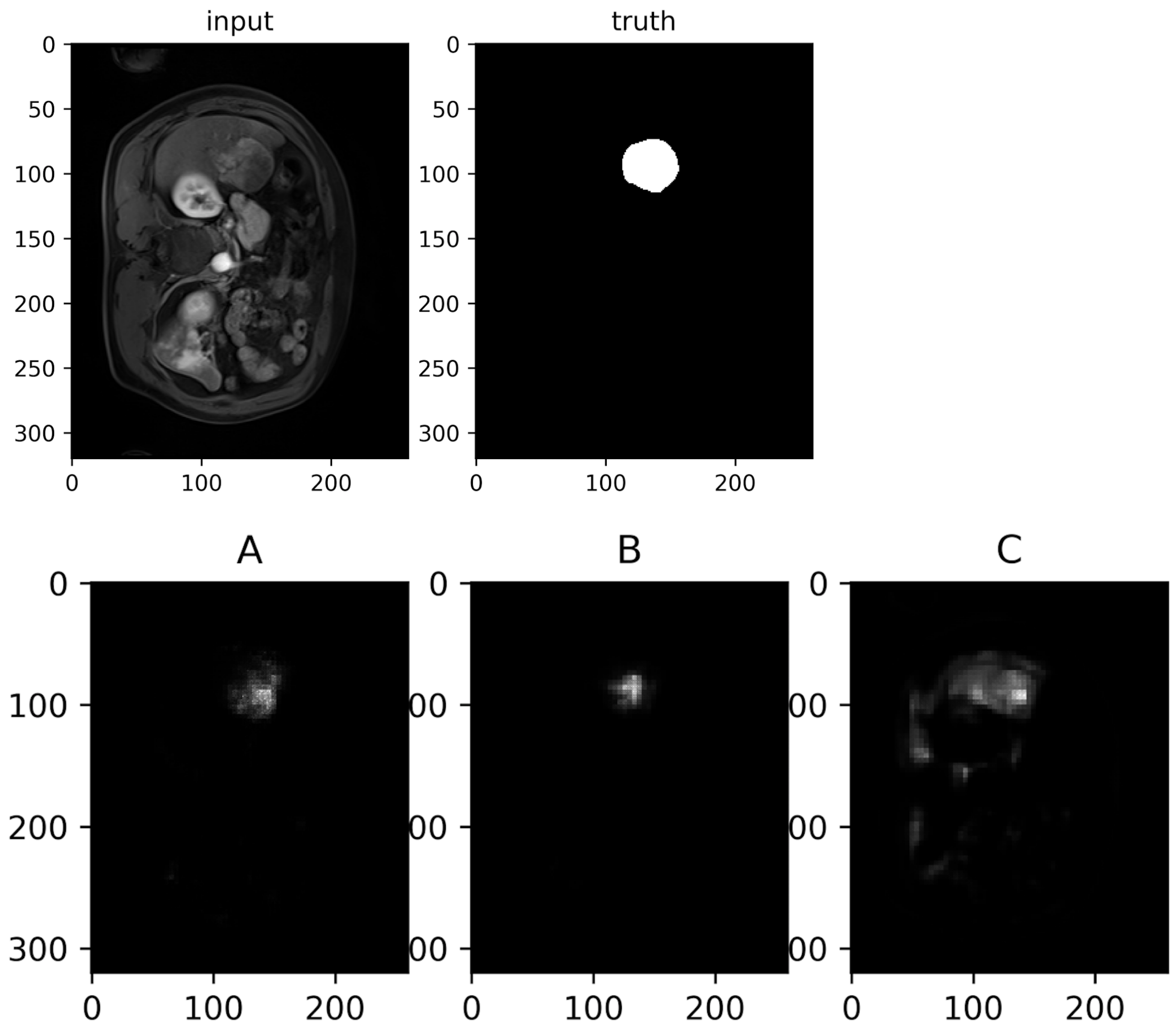

Figure 12: example 2 of the outputs of the three models and the ground truth

better results.

4.2 On Transfer Learning

composed by Longlin based on his work of the second experiment

This part analyses and comments on the results presented in the previous section about transfer learning. In general, the 2D U-Net used in this experiment is not so successful, but the results indeed shed light on the advantages of using transfer learning.

First of all, all the networks seem unable to present a very consistent result with the truth, viewing from Figure 11, 12, and 13. Compared with the ground truth, the area the models point out is blurry and vague. However, the output of the three networks does agree with each other, and reflects some knowledge about the location of the tumor.

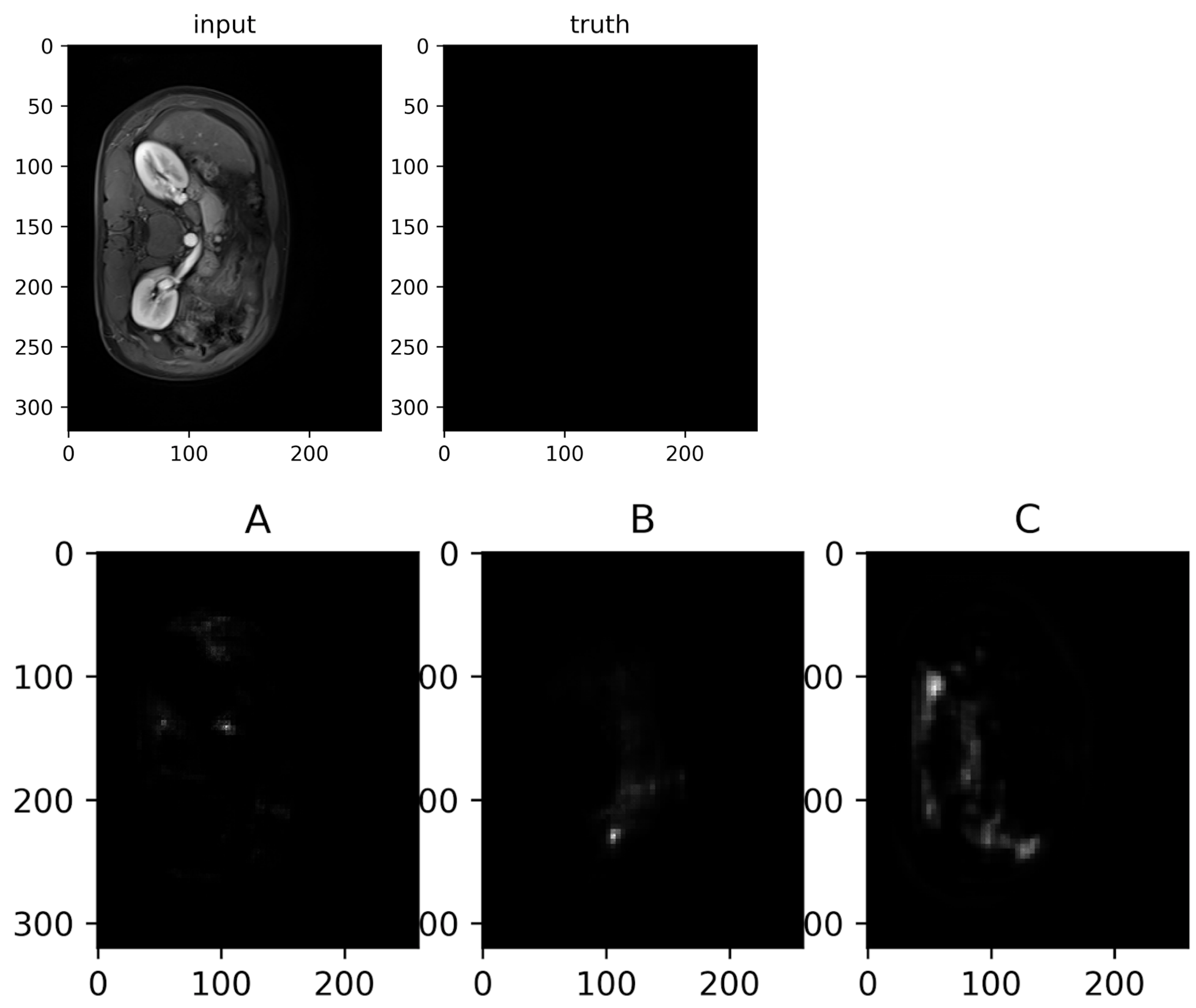

Figure 13: example 3 of the outputs of the three models and the ground truth

The possible reasons for the above results include the improper choice of the loss function or the design of the network.

The output of U-Net is a large tensor of probabilistic scores, each giving inference at a particular location. In the training data, the vast majority of the voxels are not tumor. Using simple BCE loss on the whole network, the network tend to be trained conservatively, to be less likely to give false positive (type I error) but also not able to identify the tumor (low power, prune to type II error). In the paper of Chlebus et al. (2018), the authors used the loss constrained on the liver region to increase the weights of the tumor in some sense, and trained another classifier to eliminate false positive. In our example, the labelling of the liver region has not been given, and other ways must be considered. For example, using Dice

Score could possibly reflect the network's accuracy in the tumor region more clearly. Also, the training data could be filtered first to lay more emphasis on the tumor segmentation.

Additionally, the network used in this model is a shrunk version of a top-scoring network, and may have the problem of under-parametrization.

Despite the blurry output of the networks, some interesting phenomenon about transfer learning can be observed, both from the learning curve and the output pictures of the models.

In terms of the learning curve, the ultimate loss of the three models are almost the same, while the rate of convergence to this minima follows model A ¿ model B ¿ model C. Recall that we did not observe overfitting in any of the models, and model A used Xavier normal initialization, model B used the pre-trained model as the initialization and model C used the pre-trained model and fixed the first two layers.

It seems that the prior knowledge carried in the pre-trained model does have an effect on the learning process, and makes the convergence slower. This contradicts our prior knowledge, that transfer learning can speed up the convergence. It is even more surprising that model C, the partially fixed model with the fewest parameters to be updated, turned out to be the slowest to converge. The reason behind this phenomenon is still beyond me.

In terms of the output, although only three examples are listed in this report, more have been checked by me, and it seems that the output of model A and model B are usually not so different, but output of model C does show a different pattern. In all three examples of model C's output, you can see the blurry outline of the liver.

The similarity of model A and model B could be caused by the aforementioned "catastrophic forgetting", meaning that the knowledge carried by the pre-trained model is destroyed by the training process. This is not surprising, if it has indeed happened, because catastrophic forgetting is not a rare problem, but a major topic in the research of transfer learning.

The interesting behavior of model C comes from the fact that model C has fixed the first two layers, and the knowledge of these layers cannot be easily forgotten. This is especially true for U-Nets, because the long skip connections concatenates the output of the first few layers with the end of the network. The blurry outline of the liver tells us that the network, aided by the prior knowledge in the pre-trained model, can detect the liver to some extent and base its search of the tumor on the region of the liver. Thus it tend to attach more weight to the region of the liver it has identified, causing the blurry outline to appear in the output. Of course, this outline also suggests that although the model has somehow focused on the liver, it still did not successfully identified the tumor within the liver region.

To sum up, in terms of the rate of convergence or the training speed, this experiment does leave us with some confusion, because using pre-trained model or fixed some layers strangely slows down the convergence. However, the behavior of model C gives us hope, in that there is indeed evidence that transfer training helps, and the prior knowledge in the pre-trained model plays it part. The blurry output of the three models is caused by some mistakes in choosing loss function, training data or network structure, by the effect of transfer learning is still verified to some extent. There is some future work to be done, but this last experiment is still a partial success.

References

Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2018). Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *CoRR*, *abs/1802.06955*. Retrieved from http://arxiv.org/abs/1802.06955

Chlebus, G., Schenk, A., Moltz, J. H., van Ginneken, B., Hahn, H. K., & Meine, H. (2018). Automatic liver tumor segmentation in ct with fully convolutional neural networks and object-based postprocessing. *Scientific reports*, *8*(1), 15497.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3d u-net: Learning dense volumetric segmentation from sparse annotation. *CoRR*, *abs/1606.06650*. Retrieved from http://arxiv.org/abs/1606.06650

Hou, L., Cheng, Y., Shazeer, N., Parmar, N., Li, Y., Korfiatis, P., … Song, X. (2019). *High resolution medical image analysis with spatial partitioning.*

Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., … Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. *CoRR*, *abs/1612.00796*. Retrieved from http://arxiv.org/abs/1612.00796

Milletari, F., Navab, N., & Ahmadi, S. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR*, *abs/1606.04797*. Retrieved from http://arxiv.org/abs/1606.04797

Raghu, M., Zhang, C., Kleinberg, J. M., & Bengio, S. (2019). Transfusion: Understanding transfer learning with applications to medical imaging. *CoRR*, *abs/1902.07208*. Retrieved from http://arxiv.org/abs/1902.07208

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, *abs/1505.04597*. Retrieved from http://arxiv.org/abs/1505.04597

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Cardoso, M. J. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *CoRR*, *abs/1707.03237*. Retrieved from http://arxiv.org/abs/1707.03237