Deep Learning in segmentation of medical image data

Tai-Jung Chen
Virginia Tech
Blacksburg, VA 24061-0002
taijungchen@vt.edu

Benny Antony Amaraselvam
Virginia Tech
Blacksburg, VA 24061-0002
bennyantony@vt.edu

Abstract

This project aims to implement five algorithms, which are EM algorithm, U-net, AutoEncoder, ViT AutoEncoder, and Variational AutoEncoder, to carry out the liver CT image segmentation task raised in MSD (Medical Segmentation Decathlon) challenge. The power of deep learning method comparing with the traditional statistical method, EM algorithm in this project, is discussed and demonstrated. The deep learning algorithms are fine-tuned, and U-net turns out to give a competitive dice metric (0.9923 in training and 0.9675 in testing) after 80 epochs of training.

1. Introduction

Medical image segmentation is a crucial task in the field of medical imaging, where it plays a critical role in the diagnosis, treatment planning, and monitoring of various diseases. Liver image segmentation is a particularly challenging task due to the complex shape and structure of the liver, as well as the presence of various surrounding organs and tissues.

Traditional segmentation methods rely on handcrafted features and heuristic algorithms, which are often time-consuming and require domain-specific knowledge. In recent years, teh computer vision community has been using deep learning algorithms, such as convolutional neural networks (CNNs), to show promising results in medical image segmentation tasks, including liver image segmentation. These algorithms can automatically learn features from the input images and perform accurate and efficient segmentation without the need for manual feature engineering. In this context, the use of deep learning algorithms in liver image segmentation has the potential to improve the accuracy and efficiency of liver disease diagnosis and treatment planning.

In this project, we aimed to implement five segmentation algorithms to carry out the medical image segmentation task in the Decathlon challenge [ref Antonelli] on liver CT scans. In particular, we are using one traditional statistical method, the EM algorithm, and four deep learning algorithms, which are U-net, ViT Autoencoder, Variational Autoencoder, and VarAutoEncode.

2. Methodologies

In this project, we implemented five different methodologies to carry out the medical image segmentation task. The methodologies can be categorized into the traditional statistical learning method, EM algorithm, and deep learning method, U-net. We aimed to compare the performance of these two categories, and also the performance within the deep learning category in the discussion section.

2.1. Expectation-Maximization (EM) algorithm

The Expectation-Maximization (EM) algorithm is a statistical method used to estimate the parameters of a probability distribution when some of the variables are unobserved or missing [3]. It is an iterative algorithm that alternates between two steps: the expectation step and the maximization step. In the expectation step, the algorithm estimates the missing or latent variables by computing their expected values given the observed data and the current estimates of the parameters. In the maximization step, the algorithm updates the estimates of the parameters based on the expected values of the latent variables.

EM algorithms can be used for image segmentation tasks [11, 2, 12]. Medical image segmentation involves partitioning an image into multiple regions or objects of interest. One way to perform image segmentation using the EM algorithm is to model the image intensities of the different regions using Gaussian mixture models (GMMs). The GMMs represent the probability distributions of the pixel intensities in each region. The parameters of the GMMs can be estimated using the EM algorithm, where the observed data are the image intensities and the latent variables are the labels or segmentations of the image. In the expectation step, the algorithm estimates the posterior probabilities of each pixel belonging to each region based on the current estimates of the GMM parameters. In the maximization step, the algorithm updates the estimates of the GMM parameters based

on the posterior probabilities. This process is repeated iteratively until convergence. The final segmentation of the image is obtained by assigning each pixel to the region with the highest posterior probability.

2.2. U-net

U-Net is a convolutional neural network architecture designed for semantic segmentation tasks, such as segmenting biomedical images [9]. Figure 1. shows an illustration of the structure of U-net. The architecture consists of a contracting path, which captures the context of the input image, and an expansive path, which enables precise localization of object boundaries. The contracting path is a typical convolutional network that consists of several convolutional and pooling layers to extract features from the input image. Each pooling layer halves the spatial resolution of the feature maps and doubles the number of channels, which reduces the computational cost of the network and increases the receptive field size. The expansive path uses transposed convolutions to upsample the feature maps and concatenate them with the corresponding feature maps from the contracting path, which allows the network to recover the lost spatial resolution and combine the low-level and high-level features. The final layer of the network is a pixel-wise sigmoid activation function, which produces a binary mask indicating the presence or absence of the object in each pixel. U-Net has shown state-of-the-art performance on various segmentation tasks [13], and its architecture has been extended and modified in many ways to improve its performance and adapt to different applications.

In medical image segmentation tasks, U-Net is a popular architecture. U-Net takes an input image and produces a binary segmentation map that indicates the presence or absence of the object of interest in each pixel. The network is trained on a dataset of labeled images, where each image is associated with a ground truth segmentation map. During training, the network learns to map the input image to the corresponding segmentation map by adjusting its parameters to minimize a loss function, Dice loss in our project, that measures the difference between the predicted and ground truth segmentation. Once the network is trained, it can be used to segment new images by simply passing them through the network and obtaining the output segmentation map. U-Net has shown promising results in various medical imaging applications, including brain tumor segmentation, retinal vessel segmentation, and lung segmentation, and has the potential to improve the accuracy and efficiency of medical image analysis.

2.3. AutoEncoder

Autoencoder (AE) is a type of neural network that can be used in medical image segmentation tasks. [7] The network consists of two main parts: an encoder and a decoder. The

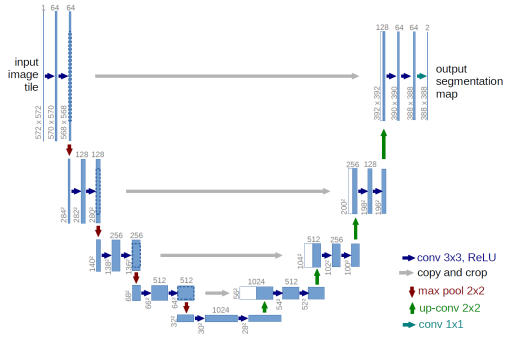

Figure 1: An illustration of the U-net structure. The left part can be viewed as an encoder and the right part can be view as an decoder.

encoder portion of the network learns a compressed representation of the input image, called a latent space, while the decoder portion learns to reconstruct the input image from the latent space. In other words, the network learns to encode the input image into a lower-dimensional representation and then decode it back to the original image.

In medical image segmentation, an AE can be trained on a large dataset of annotated medical images to learn a compressed representation of the images. Once the AE is trained, the encoder portion of the network can be used as a feature extractor to extract relevant features from the input image. These features can then be used to train a separate classifier to perform the segmentation task. Alternatively, the decoder portion of the network can be used to reconstruct the segmented image directly from the compressed representation. Overall, autoencoders provide a flexible and powerful tool for medical image segmentation and analysis.

2.4. ViT Autoencoder

ViT (Vision Transformer) Autoencoder is a recent development in deep learning architectures that combines the power of Transformers and autoencoders [5]. The Transformer model was originally designed for natural language processing tasks but has also shown promising results in computer vision. The ViT Autoencoder is based on the idea of unsupervised learning, where the model learns to reconstruct the input data without the need for labeled data. The ViT Autoencoder consists of an encoder and a decoder, where the encoder compresses the input data into a smaller latent space, and the decoder reconstructs the input data from the latent space representation.

The ViT Autoencoder has shown promising results in medical image segmentation tasks, where it can be used as a pre-training step before fine-tuning on the target segmentation task. By pre-training on a large dataset of medical images, the ViT Autoencoder can learn important fea-

tures that can be used for downstream segmentation tasks. The ViT Autoencoder can also be used in combination with other segmentation models, such as U-Net, to improve the performance of the segmentation task. The use of ViT Autoencoder on medical image segmentation tasks is still an active area of research, and further studies are needed to fully explore the potential of this architecture.

2.5. Variational Autoencoder

Variational Autoencoder (VAE) is a type of deep learning model that belongs to the family of generative models [8]. VAEs are based on the idea of learning a probabilistic distribution of the input data and generating new data samples from this distribution. The VAE consists of an encoder and a decoder, where the encoder maps the input data into a latent space representation, and the decoder maps the latent space representation back to the input data. Unlike traditional autoencoders, VAEs introduce a regularization term that forces the model to learn a smooth and continuous latent space representation.

VAEs have shown promising results in medical image segmentation tasks, where they can be used for both unsupervised and supervised learning. By pre-training the VAE on a large dataset of medical images, the model can learn important features that can be used for downstream segmentation tasks. The VAE can also be used for semi-supervised learning, where a small amount of labeled data is used to train the model, and the remaining unlabeled data is used for pre-training. In addition, VAEs can be used for data augmentation, where new synthetic data samples are generated from the learned distribution to increase the size of the training dataset.

However, VAEs have limitations in medical image segmentation tasks, particularly in cases where the boundaries between the organs or tissues of interest are not well-defined. In such cases, VAEs may generate unrealistic and blurry segmentation masks. Therefore, researchers are actively exploring ways to improve VAEs' performance in medical image segmentation tasks, such as combining VAEs with other segmentation models or introducing new regularization terms to improve the model's ability to capture fine details.

2.6. Dice metric

Dice coefficient is a popular metric used to evaluate the quality of the output of a deep learning segmentation algorithm [6]. It measures the similarity between the predicted segmentation mask and the ground truth segmentation mask of an image. The Dice coefficient ranges from 0 to 1, where a value of 1 indicates perfect overlap between the predicted and ground truth segmentation, and a value of 0 indicates no overlap. The formula for computing the Dice coefficient is $Dice = 2 \times \frac{predicted \cap groundtruth}{\sum \text{ pixels in predicted mask} + \sum \text{ pixels in ground truth mask}}$.

The interpretation of dice can be shown in Figure 2. As mentioned, high Dice coefficient values indicate that the algorithm is producing accurate segmentation that closely match the ground truth, while low values indicate poor segmentation performance. Therefore, Dice is used in this project to evaluate the performance of the model output.

3. Experiment results

In this section, the data description of the dataset we used in this project will be laid out. In addition, the process from data preprocessing, model construction, model training, to testing the trained data will be described. Finally, the experiment results will be displayed.

3.1. Data description

The dataset used in this project is adapted from the paper "A large annotated medical image dataset for the development and evaluation of segmentation algorithms" [10]. The paper presents a new dataset of medical images that can be used to develop and evaluate segmentation algorithms. The dataset consists of 1,000 3D CT and MR scans from a variety of imaging modalities, including abdominal CT, chest CT, and brain MR, with resolutions ranging from 0.5mm to 1.0mm. The images were acquired from various clinical sites and patient populations, ensuring a diverse range of imaging characteristics and pathologies.

The dataset is annotated with high-quality pixel-level segmentation masks for various anatomical structures and lesions, including the liver, lung, pancreas, spleen, kidney, and tumors. The annotations were performed by expert radiologists using a semi-automated annotation tool, ensuring high accuracy and consistency. The dataset also includes metadata such as patient age, sex, and imaging modality, which can be used for further analysis and stratification. The availability of this large and diverse dataset with high-quality annotations can greatly benefit the development and evaluation of segmentation algorithms in medical imaging and has the potential to improve the accuracy and efficiency

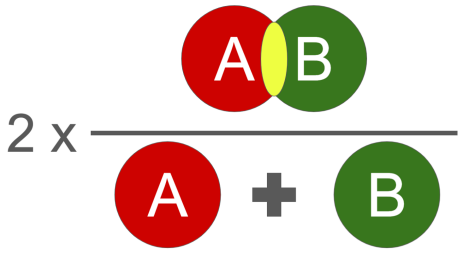

Figure 2: A graphical interpretation of how Dice is measuring the performance of segmentation algorithms.

of clinical diagnosis and treatment planning.

In addition, we were inspired by the medical image segmentation challenge raised in the paper "The Medical Segmentation Decathlon" [1]. Antonelli et al. note that while deep learning methods have shown promise in medical image segmentation tasks, the lack of standardized benchmarks and evaluation metrics has hindered progress and comparison of different methods. To address this issue, the authors propose a new challenge that consists of ten segmentation tasks across different anatomical structures and imaging modalities, including brain MRI, cardiac MRI, abdominal CT, and lung CT. We decided to carry out the segmentation of liver CT scan from one of the ten tasks.

3.2. Data preprocessing

The first step in data preprocessing is image resizing. The U-net architecture requires the input and output images to be of the same size. Therefore, we resized the original liver CT scans to a fixed size of 512 x 512 pixels using the OpenCV library in Python. The resized images are saved in a new folder for further processing.

The next step is to normalize the pixel intensity values of the images. Medical imaging modalities may have different intensity ranges, and normalization can help to ensure that the U-net algorithm can effectively learn from the images. We performed min-max normalization, which scales the intensity values to a range of 0 to 1, using the NumPy library in Python. The normalized images are saved in another folder, which will be used as the input to the U-net algorithm.

In addition to resizing and normalization, data augmentation is also an important factor in training the U-net algorithm. Data augmentation involves applying random transformations to the input images, such as rotations, flips, and zooms, to increase the diversity of the training data and prevent overfitting. We performed data augmentation using the monai library in Python.

3.3. Results

The experimental results of the five algorithms we implemented will be displayed in this section. EM algorithm will be showed first. U-net, ViT autoencoder, Variational Autoencoder, and VarAutoEncode will follow along.

3.3.1 EM algorithm

Since the EM algorithm is using a mixture Gaussian to cluster the data, we further partition the dataset by how many labels in the CT scan into three categories, which are three-label samples, two-label samples, and one-label samples. This partition is prepared for the EM algorithm, so that we can feed the three-label samples to a mixture Gaussian model with 3 components, and feed the two-label samples to a mixture Gaussian model with 2 components. The number of samples in the three categories are summarized in table 1.

Table 1: Summary of the number of samples in the three categories.

	Three-label	Two-label	one-label
Training set	6009	10857	34313
Testing set	1174	1123	5162

Table 2: Performance of EM algorithm

	Mean dice	Standard Error	Max dice
Three-label set	0.1749	0.0046	0.6072
Two-label set	0.0652	0.0032	0.4336

The EM algorithm's segmentation result is shown in table 2. The configuration for the three-label model is using a three-component mixture Gaussian which represents the background, liver, and cancer. The configuration for the two-label model is using a two-component mixture Gaussian which represents either two of background, liver, and cancer. The maximum iteration for the two models are both set to be 100, and the tolerances are both set to be 1e-5.

3.3.2 U-Net

U-net was run under three different configurations. The difference between configurations are from the optimizer, and batch size. The configurations are summarized in table 3 along with the performance (table 4). Figure 3. also shows the loss to epoch graph and dice to epoch graph to monitor whether there are overtraining issues during the training. Figure 4. shows the testing performance of the U-net under the best configuration, which is configuration 3 in this case.

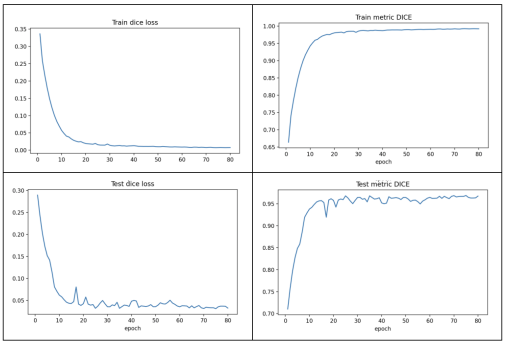

Figure 3: The loss to epoch and dice to epoch plot for U-net in configuration 3

Table 3: Configuration of the U-net models

Configuration 1	
Epoch	80
Batch Size	32
Channels	(16, 32, 64, 128, 256)
Optimizer	Adam
	(learning rate = 1r-5, weight decay = 1e-5)
Configuration 2	
Epoch	80
Batch Size	64
Channels	(16, 32, 64, 128, 256)
Optimizer	SGD
	(learning rate = 1e-3)
Configuration 3	
Epoch	80
Batch Size	48
Channels	(16, 32, 64, 128, 256)
Optimizer	RMSProp
	(learning rate = 1e-5, momentum = 0.9)

Table 4: The performance of U-net

Configuration 1	
Train epoch loss	Train epoch dice
0.0454	0.9546
Test epoch loss	Test epoch dice
0.0702	0.9298
Configuration 2	
Train epoch loss	Train epoch dice
0.1002	0.8998
Test epoch loss	Test epoch dice
0.1181	0.8819
Configuration 3	
Train epoch loss	Train epoch dice
0.0077	0.9923
Test epoch loss	Test epoch dice
0.0325	0.9675

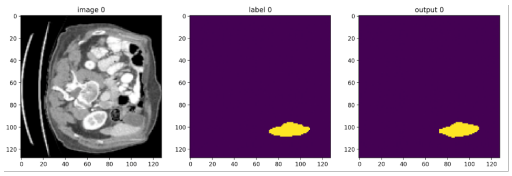

Figure 4: Slice image of the volume, label, and segmentation output by U-net in configuration 3

3.3.3 AutoEncoder

AutoEncoder was run under two different configurations. The configurations are summarized in table 5 along with the performance (table 6). Figure 5. also shows the loss to epoch graph and dice to epoch graph to monitor whether there are overtraining issues during the training. Figure 6. shows the testing performance of the U-net under the best configuration, which is configuration 3 in this case.

Table 5: Configuration of the AutoEncoder models

Configuration 1
spatial dims=3,
in channels=1,
out channels=2,
channels=(4,),
strides=(2,),
inter channels=(8, 8, 8),
inter dilations=(1, 2, 4),
num inter units=2
epoch = 500
Configuration 2
spatial dims=3,
in channels=1,
out channels=2,
channels=(2, 4, 8),
strides=(2, 2, 2),
kernel size=(3,5,3),
up kernel size=(3,5,3),
num res units=2
epoch = 500

Table 6: The performance of AutoEncoder

Configuration 1	
Train epoch loss	Train epoch dice
0.1311	0.8689
Test epoch loss	Test epoch dice
0.1224	0.8776
Configuration 2	
Train epoch loss	Train epoch dice
0.0780	0.9220
Test epoch loss	Test epoch dice
0.1106	0.8894

3.3.4 ViT Autoencoder

ViT Autoencoder was run under two different configurations. The configurations are summarized in table 9 along with the performance (table 10). Figure 9. also shows the loss to epoch graph and dice to epoch graph to monitor whether there are overtraining issues during the training. Figure 10. shows the testing performance of the U-net under the best configuration, which is configuration 3 in this case.

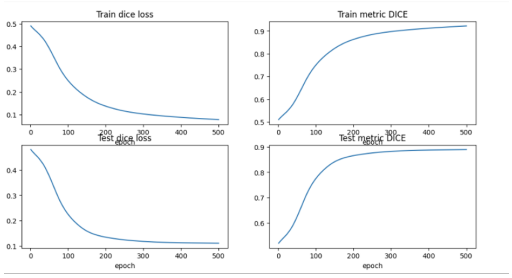

Figure 5: The loss to epoch and dice to epoch plot for AutoEncoder in configuration 2

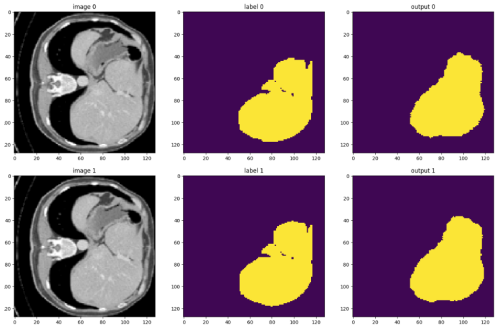

Figure 6: Slice image of the volume, label, and segmentation output by AutoEncoder in configuration 2

Table 7: Configuration of the ViT AutoEncoder

Configuration 1
in channels=1, patch size=(16,16,16), img size=(128,128,64), out channels=2, pos embed='conv', spatial dims=3 epoch = 10
Configuration 2
in channels=1, patch size=(16,16,16), img size=(128,128,64), out channels=2, deconv chns=16, hidden size=768, mlp dim=3072, num layers=12, num heads=12, pos embed='conv', spatial dims=3 epoch = 600

Table 8: The performance of ViT AutoEncoder

Configuration 1	
Train epoch loss 0.4299	Train epoch dice 0.5701
Test epoch loss 0.4148	Test epoch dice 0.5852
Configuration 2	
Train epoch loss 0.3992	Train epoch dice 0.6008
Test epoch loss 0.4003	Test epoch dice 0.5997

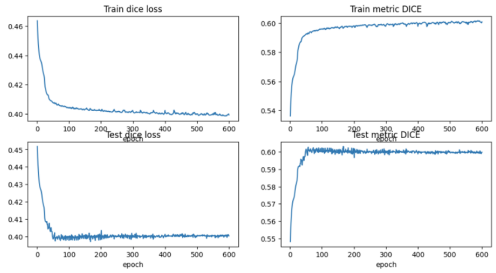

Figure 7: The loss to epoch and dice to epoch plot for ViT AutoEncoder in configuration 2

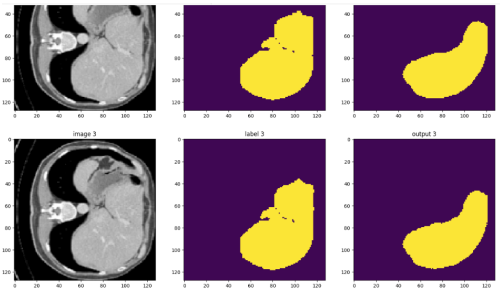

Figure 8: Slice image of the volume, label, and segmentation output by ViT AutoEncoder in configuration 2

3.3.5 Variational Autoencoder

Variational Autoencoder was run under two different configurations. The configurations are summarized in table **??** along with the performance (table **??**). Figure **??**. also shows the loss to epoch graph and dice to epoch graph to monitor whether there are overtraining issues during the training. Figure **??**. shows the testing performance of the Variational AutoEncoder under the best configuration, which is configuration 1 in this case.

Table 9: Configuration of the Variational AutoEncoder

Configuration 1
spatial dims=3, in shape=[1, 128, 128, 64], out channels=2, latent size=2, channels=(16,32,48,64), strides=(2, 2, 2,2), kernel size=(3,5,3), up kernel size=(3,5,3), num res units=3 epoch = 600 optimizer = SGD
Configuration 2
spatial dims=3, in shape=[1, 128, 128, 64], out channels=2, latent size=2, channels=(8,16,32,48), strides=(2, 2, 2,2), kernel size=3, up kernel size=3, num res units=0 epoch = 600 optimizer = Adam

Table 10: The performance of Variational AutoEncoder

Configuration 1	
Train epoch loss 0.4076	Train epoch dice 0.5924
Test epoch loss 0.4018	Test epoch dice 0.5982
Configuration 2	
Train epoch loss 0.4214	Train epoch dice 0.5786
Test epoch loss 0.4167	Test epoch dice 0.5833

4. Discussion

First of all, the results of EM algorithm and the deep learning methods, U-net, AutoEncoder, ViT AutoEncoder, and Variational AutoEncoder, are compared. The performance of EM algorithm is significantly worse than the four deep learning algorithms. Although the maximum dice for a particular sample can be comparable to the output of deep learning algorithm (0.6072), the average performance of EM algorithm is not satisfying. However, the time for EM algorithm to carry out the segmentation is significantly faster than the deep learning algorithms. It took around half an hour for EM to carry out the three-label segmentation

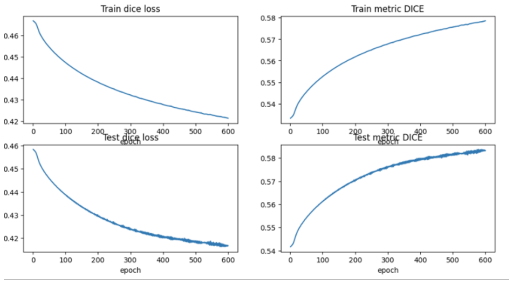

Figure 9: The loss to epoch and dice to epoch plot for Variational AutoEncoder in configuration 1

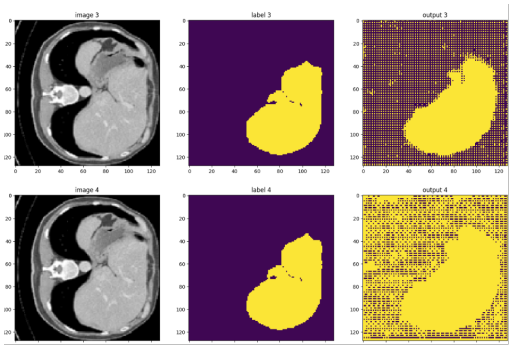

Figure 10: Slice image of the volume, label, and segmentation output by Variational AutoEncoder in configuration 1

task. Yet for U-net, it requires 6 hours to get the output. Another thing worth noting is that EM algorithm does not require training. Therefore, EM algorithm is a wonderful algorithm for simple data, but not so desired for medical image segmentation.

Comparing within the category of deep learning models, U-net has the best performance. U-net's training time to get the optimal dice is also significantly shorter (80 epochs vs. 500 epochs) comparing to the second best algorithm, AutoEncoder configuration 2. Thus, U-net is the optimal deep learning algorithm in this project.

In addition, for U-net, the selection of optimizer can improve the dice metric around 8% (comparing configuration 2 to configuration 3). Therefore, the selection of optimizer is also crucial for optimizing U-net.

5. Contribution

In this project, Tai-Jung Chen carried out the implementation of EM algorithm along with partitioning the dataset into three-label, two-label, and one-label datasets. He also built U-net via Monai package, and fine-tuned U-net to acheive the optimal result.

Benny Antony Amaraselva was in charge of building and fine-tuning the AutoEncoder, ViT AudoEncoder, and Var AutoEncoder. He also provided background knowledge to this project since this is in his research area.

Overall, the contribution for this project are stated as below. First, the EM algorithm is implemented and compare to these modern deep learning methodologies. This ensures the power of deep learning methods again, and shows the importance of its application. In addition, although we do not have a chance to get the real testing label, our U-net performance is competitive to the teams in the MSD (Medical Segmentation Decathlon) Challenge. The top one team is having 0.95 dice for the liver segmentation, and we are having 0.9675 dice. Again, we only have the training labels (the 0.9675 is actually a validation set performance), but it's a good sign that our model should be competitive to the ones on the leader board.

Acknowledgements

Thanks to the tutorial created by FreeCodeCamp.org and Mohammed El Amine [4] to give a solid base for us to carry out the code in this project further.

References

[1] M. Antonelli, A. Reinke, S. Bakas, K. Farahani, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, O. Ronneberger, R. M. Summers, et al. The medical segmentation decathlon. *Nature communications*, 13(1):4128, 2022.

[2] G. Boccignone, P. Napoletano, V. Caggiano, and M. Ferraro. A multiresolution diffused expectation–maximization algorithm for medical image segmentation. *Computers in Biology and Medicine*, 37(1):83–96, 2007.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[4] M. A. Haddad. Liver segmentation using monai and pytorch. `https://github.com/amine0110/Liver-Segmentation-Using-Monai-and-PyTorch`, 2021. Accessed: [insert date here].

[5] J. Huang, H. Li, G. Li, and X. Wan. Attentive symmetric autoencoder for brain mri segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part V*, pages 203–213. Springer, 2022.

[6] S. Jha, R. Kumar, I. Priyadarshini, F. Smarandache, H. V. Long, et al. Neutrosophic image segmentation with dice coefficients. *Measurement*, 134:762–772, 2019.

[7] S. Karimpouli and P. Tahmasebi. Segmentation of digital rock images using deep convolutional autoencoder networks. *Computers & geosciences*, 126:142–150, 2019.

[8] C. Nash and C. K. Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. In *Computer Graphics Forum*, volume 36, pages 1–12. Wiley Online Library, 2017.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[10] A. L. Simpson, M. Antonelli, S. Bakas, M. Bilello, K. Farahani, B. Van Ginneken, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv preprint arXiv:1902.09063*, 2019.

[11] S. K. Warfield, K. H. Zou, and W. M. Wells. Validation of image segmentation and expert quality with an expectation-maximization algorithm. In *in Proceedings of Fifth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), Part I*, 2002.

[12] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE transactions on medical imaging*, 20(1):45–57, 2001.

[13] Z. Zhuang, N. Li, A. N. Joseph Raj, V. G. Mahesh, and S. Qiu. An rdau-net model for lesion segmentation in breast ultrasound images. *PloS one*, 14(8):e0221535, 2019.