Convolutional Neural Networks in Medical Imaging

Mitchell Finzel
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
finze008@morris.umn.edu

ABSTRACT

Over the past 5 years there has been an increase in the use of convolutional neural networks in a broad variety of medical imaging applications. This is due in part to the increase in their popularity since their success in the 2012 ImageNet competition, but is also due to their adaptability across a range of medical imaging applications. These applications vary greatly; from the segmentation of knee cartilage to the detection of Alzheimer's disease in MRIs and much more. In this paper we will go over some of the cutting edge techniques being used specifically for the tasks of brain segmentation; classifying with both binary segmentation on brain lesions and hierarchical segmentation with tumors. The results are proving to be quite promising with many of the described techniques outscoring previous state-of-the-art systems.

Keywords

Convolutional Neural Networks, Medical Imaging, Biological Segmentation

1. INTRODUCTION

In 2012 convolutional neural networks or CNNs, were used to great success improving dramatically over the previous state-of-the-arts in the ImageNet computer vision competition [4]. Since their success in image recognition CNNs have seen a rise in popularity, finding their way into more complex computer vision challenges such as medical imaging. In the past 5 years there has been an increase in the use of CNNs in biological segmentation tasks. These tasks extend across a wide variety of human anatomy. For example, CNNs have been used for the automated detection of lymph nodes [7], the segmentation of knee cartilage [6] and Alzheimer's detection [5] to name a few. For this paper we will be focusing on two specific examples of CNN use in medical imaging segmentation: [2] by Havaei, et al. and [3] by Kamnitsas, et al. These two papers show different approaches to CNN architectures applied to the segmentation of MRIs. While the results obtained by both approaches are not directly comparable, we will go through what differs between their approaches and where there is some overlap. After discussing their features we will take a look at their state-of-the-art results on a few different segmentation challenges.

UMM CSci Senior Seminar Conference, April 2017 Morris, MN.

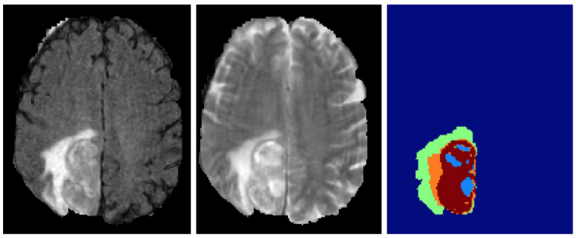


Figure 1: An example of tumor segmentation adapted from [3]

2. BACKGROUND

Before delving into the specifics of the network architecture approaches that are being employed we will look at some of the fundamentals of CNNs and segmentation in general. When discussing segmentation in the field of medical imaging we are discussing the ability to classify different parts of a medical image. This segmentation is fairly loosely defined and can be used to describe classification through a variety of granularities. On a more coarse granularity we might have an x-ray of a leg where we desire to differentiate and label the different bones in the image. On a finer granularity we might be interested in being able to identify and label different parts of tumors in an MRI of a brain much like in Figure 1. Currently most of this segmentation is done by hand by medical professionals. This is where convolutionl neural networks come in. The CNN is trained on a set of images that have been broken into patches that have been properly labeled by medical professionals. This teaches it how to differentiate the different parts of the image on its own. The network then takes unlabeled image patches as input and uses its training to attempt to correctly label the image. The end goal is the network generating correct image labels e.g. is or isn't a tumor, much as though it had been labeled by hand. The following background sections draw from the tutorial created by Adit Deshpande [1].

2.1 Introduction to Neural Networks

At their most basic form neural networks are pattern recognizers modeled on the neuronal structures of the cerebral cortex, the part of the brain that takes in sensory data. As seen in Figure 3, networks are generally comprised of layers of nodes that activate when they recognize a certain input.

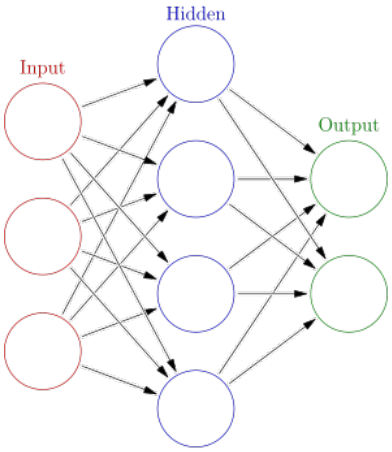


Figure 2: A simple artificial neural network. This network only has one hidden layer, but neural networks often have many hidden layers. https://goo.gl/mVAz5N

The result of these activations are then passed to neighboring nodes through weighted connections. After passing through the layers of nodes and connections the resulting data is sent out of the network as some form of output. What makes neural networks so powerful is their general usability across a wide range of data driven problems as well as the ability to alter the weights of the node connections to improve the accuracy of their output.

In our case we are interested in a type of neural network called a convolutional neural network. CNNs are based on many of the same principles, input, hidden, and output layers, of a normal neural network, but have an additional type of layer that has been found very useful when it comes to learning things about images. In our case the network inputs will be patches of an MRIs pixels with the output being segmentation labels.

2.2 Convolutional Layers

Convolutional layers are what differentiate convolutional neural networks from other neural networks, the first layer of all CNNs is a convolutional one. These layers are used to condense the input data into recognized data patterns, thus reducing data size and recognizing things of interest. The convolutional layer takes an array of values that represents either the pixels or voxels of a patch of the input image. The layer then uses what is interchangeably called a filter, neuron or kernel, which is another array representing some sort of feature. The kernel is then aligned to the upper left corner of the input, the area it covers is called the receptive field. The array contained within the receptive field is then multiplied with the array in the kernel using element-wise multiplication. The multiplications are then summed up and stored in the same relative position of what's called a feature map as seen in Figure 3. The kernel then slides over a specified distance on the input and performs the same operation, storing the result in the next position of the feature map. What we end up with after all of the possible convolutions of the kernel and the input is a completed feature map. The feature map is an array that contains all of the results of the convolutions between the kernel and the input. These feature maps can then be passed on as input to future layers.

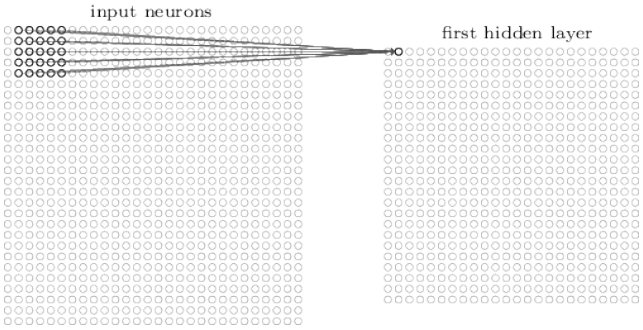
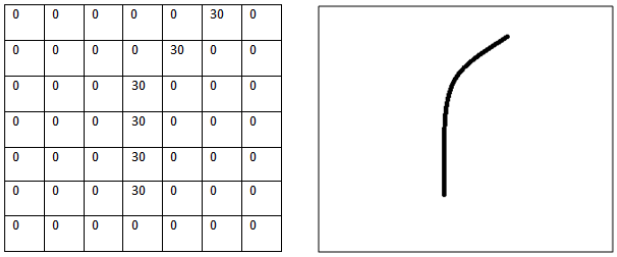


Figure 3: Visualization of kernel activations being stored in the feature map, also known as an activation map. Adapted from [1]



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Figure 4: An example of a kernel that recognizes a curve as a feature. Adapted from [1]

2.3 Kernels

Kernels, as described above, are arrays of values that are meant to represent features to be recognized. For instance a kernel could contain a feature such as a curve as shown in Figure 4. This might be represented by a pattern of numbers in the kernel. When the kernel is multiplied with the input the result will be a higher number if the feature in the kernel is similar to the feature in the receptive field. If the feature described by the kernel is not present in the receptive field then the result of the multiplication will be smaller. These recognitions of features are then stored in the feature map where they will likely be used as the input to the next layer. The more of these kernels there are, the more features the network can recognize. As these feature maps are used in future layers of similar operations a hierarchy of features is created with more complex features being represented in later layers.

2.4 Dropout and Pooling Layers

Two more layer types used in convolutional neural networks are dropout and pooling layers. First up is the pooling layer. Pooling layers have a relatively straightforward purpose; they take clusters from the input feature map and reduce them down to a single feature. An example of a pooling operation is max-pool. A max-pool pooling layer will divide the input into clusters and place the highest value of each cluster in their corresponding place in the pooling layer's feature map. Pooling layers are used to drastically reduce the amount of spatial data by eliminating a large

portion of the input in one step.

Next is the dropout layer. Dropout layers, true to their names, randomly select activations in the input feature map and set them to zero. This helps make sure the network can correctly predict classifications even when some of the activations are dropped out. The redundancy provided by dropout layers is useful in the training process, but is not applicable to non-training data.

Both of these layer types are useful tools in addressing the issue of overfitting in CNNs. Overfitting is essentially when the network becomes too finely tuned to the training data set and doesn't generalize well to images it hasn't seen yet. There are many other methods for addressing overfitting, but these are the only two within the scope of this paper.

2.5 Fully Connected Layers

The fully connected layer is often the final layer in the network. It takes in as input a feature map from the prior layer and returns a vector of label probabilities for the center pixel of the input image patch. A CNN might e.g. be trying to segment an image into what is and isn't a tumor. The output is therefore a binary option, tumor or non-tumor. The fully connected layer looks at the features represented in the feature map and then provides a vector with two values, the probability that the input is a tumor and the probability that it is not a tumor.

2.6 Training

Now that we have gone through some of the components of a CNN we can get into the thing that makes it all work, the training process. Before going into the basic steps of the training process it is important to note that a training data set is required to begin the process. The training data set in our case would be medical images that are paired with their ground truth labels. These ground truth labels are the labels observed by a medical professional in manual segmentation. This way we have images that the network can try to segment, checking its results against the ground truth labels provided.

The kernels in the network originally start off randomized and therefore the output probabilities should all be approximately equal. On the forward pass an image patch from the training data is sent through the network and the output probability is compared to the true label probability provided with the test image. This comparison is put through a loss function to quantify the inaccuracy. At the start of the training process the loss will likely be very high, the goal being minimizing the loss as much as possible. The loss function is then used in the next step of the process called the backward pass. In the backward pass you progress iteratively back through the network from output to input, evaluating which kernels contributed the most to the total loss and calculating the weight adjustments that would minimize said loss. After the backward pass is complete the final step, weight update, is performed. This final step takes the loss minimizing weight changes from the backward pass phase and implements them. Applying this four-step process for every image in the training data set is considered one epoch; training generally requires many epochs.

After completing the training process the network can be tested on a testing data set. The testing data set, much like the training data, contains images and their true labels. This data set allows for the evaluation of the networks performance before using it on unlabeled data. The data in the testing set cannot contain images from the training set because of the inherent bias the network has towards the images it was trained on.

There are a few things that are important to note about the training process. In general the more images in the training data set the better. This can be a hurdle for medical image applications, due to the difficulty of gathering appropriate images and the time involved with labeling them. There are some methods of augmenting the data set to increase the size of the training image pool, such as application of rotations, translations and jitter to images in the training set. Truly different images are preferable.

3. METHODS

In this section we will go over two separate approaches for brain MRI segmentation taken by Havaei, et al. and Kamnitsas, et al. We will break these approaches down into their core components, addressing some of the costs and benefits of these choices. In discussing these approaches we will also discuss some of the similarities and differences between them.

3.1 Two Pathway Approach to Brain Tumor Segmentation

In [2], Havaei, et al. created a novel two path approach for a CNN trained on the Multimodal Brain Tumor Segmentation (BRATS 2013) challenge data set. This challenge is comprised of three data sets, 30 training images with ground truth labels, 10 test images with labels and 25 leaderboard images without ground truth labels. The segmentation task contains five different labels to be segmented; non-tumor, necrosis, edema, non-enhancing tumor and enhancing tumor. Their approach to this challenge can be broken down into three main components; the use of two pathways, the concatenation of one CNN's output into varying locations in a second network, and the use of a two-phase training approach.

To setup their two pathway CNN Havaei, et al. created an architecture with two streams, a *local pathway* with a 7 x 7 receptive field and a *global pathway* with a 13 x 13 receptive field, see Figure 5. By combining a localized and global perspective the architecture has the ability to detect visual detail around the centered pixel while also capturing data about the greater context of that pixel's location within the brain.

These two pathways are concatenated together after going through a series of convolutional layers, 2 layers for the local pathway and 1 for the global pathway. This final concatenation of the two pathways is then sent through the output layer to be interpreted as segmentation labels.

Next is an issue with traditional CNN segmentation systems that predict the segmentation labels independent of one another, ignoring the possibility of joint segmentation label models where different labels correlate. To address this issue Havaei, et al. propose three different cascaded CNN architectures, where the output of one CNN is concatenated into one of the layers in a second CNN. By using a cascaded architecture they allow the second CNN to learn from the labeling of nearby pixels.

The three different cascaded architectures implemented by Havaei, et al. are variations on their two pathway approach described above; both of the CNNs, the original and

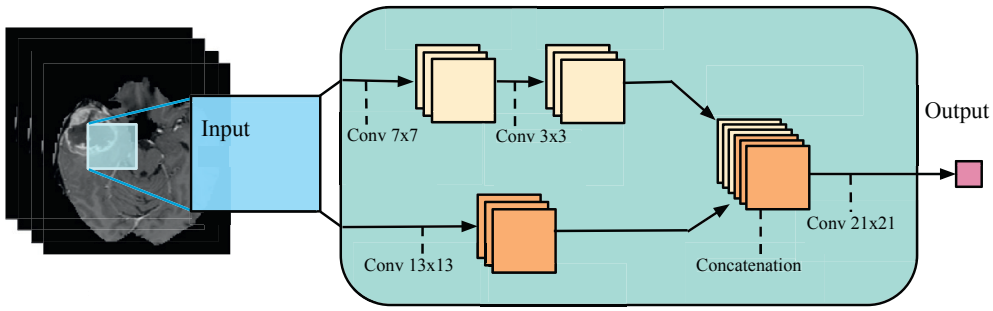


Figure 5: The base two-pathway architecture used by Havaei, et al. Adapted from [2].

the one being concatenated in later, use two pathways. The main variation between these three different architectures is the location of the concatenation of the first CNN with the second. Looking at the basic architecture seen in Figure 5, the first implementation, InputCascadeCNN, takes the output of the first CNN and directly concatenates it to the input of the second CNN. The second implementation, LocalCascadeCNN, takes the output of the first CNN and concatenates it to the first convolutional layer of the second CNNs local pathway. The final implementation, MFCascadeCNN, concatenates the output of the first CNN to the final layer of the second CNN, directly before its output.

The third main component in the research done by Havaei, et al. is their adoption of a two-phase training system. One of the large issues in training CNNs for segmentation is the relative abundance of healthy tissue as compared to the small quantities of tissue that fall under some other labels. This is especially true of brains where labels might represent a small percentage of the total image's composition e.g. small quantities of tumor tissue. To alleviate this problem Havaei, et al. first train the CNN on a data set of image patches where all of the labels are equally probable. They then retrain the final output layer taking into account the relative probabilities of the labels, thereby keeping the discriminatory capability of the previous layers intact while maintaining proper output probabilities.

3.2 3D Multi-Scale Approach

In [3], Kamnitsas, et al. provide one of the most recent architectural approaches to CNNs in the field of medical imaging segmentation. Their work can be boiled down to a few main techniques drawing from a wealth of past research. These main techniques include the use of 3D CNNs, dense-inference for network training, 3D CRFs for the final processing of the network's segmentation, deep networks for better discrimination and two pathways using a multi-scale approach.

Much like the work done by Havaei, et al., Kamnitsas, et al. uses a two pathway approach to better capture local information while maintaining the broader context of the entire image. However, while Havaei, et al. accomplish this two pathway approach by using a different sized receptive field in each pathway, Kamnitsas, et al. downsample the image itself, shrinking the size of the image in the global pathway.

The network displayed in Figure 6 is a simplified version of Kamnitsas's full blown network, 'DeepMedic'. As can be seen in the figure the network has two pathways, one with the normal resolution image patch and the other with a down-sampled patch. These patches then go through a series of convolutional layers before two fully connected layers and the final classifier layer. DeepMedic has twice as many convolutional layers as the depicted figure, using a smaller 3^3 kernel size.

One of the biggest attributes of the work done by Kamnitsas, et al. is the use of 3D CNNs for the basis of their architecture. 3D CNNs are characterized by their use of 3D kernels and typically have more volumetric image patches as input. In the past, research has avoided the use of full blown 3D CNNs due to their increase in parameters and computational requirements. Hybrid approaches such as training on three orthogonal slices (coronal, sagittal and axial) have been used to cheaply approximate 3D CNNs [7]. The 3D CNNs used by Kamnitsas, et al. use three-dimensional kernels in the convolutional layers to create the feature maps. This can be thought of as a rectangular prism traversing the volume of the image at hand. The use of these 3D CNNs allow for better representation of the volumetric data and allows for the full exploitation of dense-inference. Dense-inference or dense-training, is a process used by Kamnitsas, et al. to help lower the additional computational costs associated with 3D CNNs. When the input image patch is larger than the receptive field of the CNN the CNN's classification layer can convolve across the patch and output multiple predictions rather than the traditional single prediction. The expected cost savings come from decreasing the number of overlapping patches sampled from the test data.

In CNNs it has been shown that deeper networks, networks with more consecutive layers, have greater discriminitive capability. This is due to their additional nonlinearities, meaning outputs disproportionate to inputs, and better defined local optima, meaning more finely defined features in kernels. While deeper networks have greater discriminitive power it is also true that they have greater computational costs, which is further exaggerated by the use of 3D convolutions. There is also an increase in the number of trainable parameters that is associated with the use of 3D CNNs. To combat these barriers Kamnitsas, et al. replace convolutional layers that have a standard 5^3 kernel size with multiple layers that use a smaller 3^3 kernel size. These smaller kernels require fewer parameters and lead to a significant decrease in element-wise multiplications per layer. [3]

Deeper networks also suffer from being harder to train. As the network becomes deeper it becomes harder to preserve

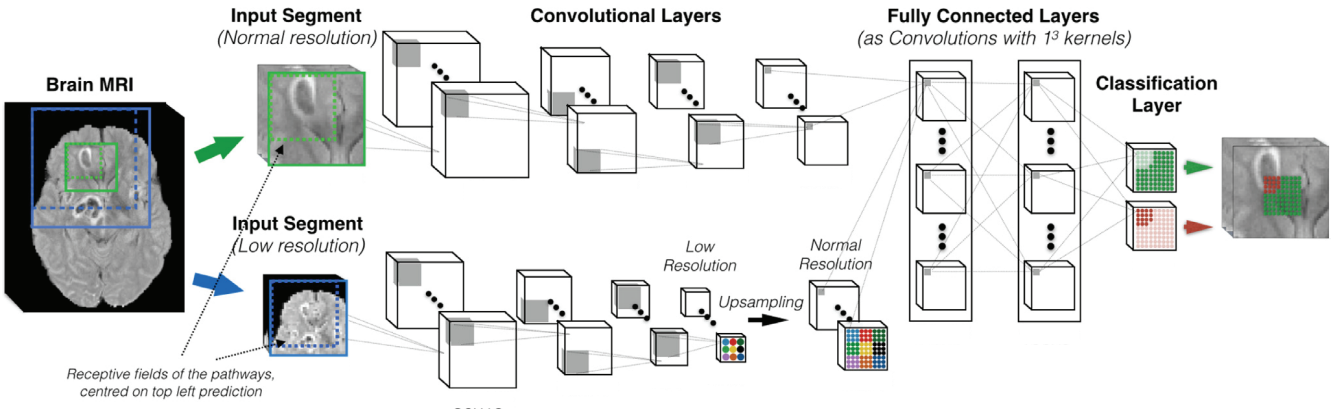


Figure 6: The basic neural network architecture, 'Deep Medic', used by Kamnitsas, et al. Adapted from [3].

the loss function. This is caused by the multiplication of the loss function's variance as it propagates through each layer. To alleviate this Kamnitsas, et al. initialize their kernel weights to the normal distribution $N(0, \sqrt{2/n_l})$ where n_l is the number of weights through which a neuron is connected in layer l to the input. They also use a technique called batch normalization to deal with the similar issue of covariate shift, but this is outside of the scope of this paper.

The final contribution from the work of Kamnitsas, et al. that we will discuss is the use of what they call the first fully 3D conditional random field, or CRF. CRFs are used in neural networks as a means of taking the predictions from the normal fully-connected layers and giving them the ability to make predictions given the context of the neighboring areas in the image as well. The result is a more precise segmentation than that provided by the more traditional output layer.

4. RESULTS

In the discussion of the results there are four different values being reported. The first is the Dice score which is a statistic measuring similarity calculated as

$$Dice(P, G) = \frac{|P \cup G|}{(|P| + |G|)/2} \quad (1)$$

In this case P is the networks predictions and G is the ground truth labels. Next up is specificity also known as the true negative rate; calculated as

$$specificity = \frac{TN}{(TN + FP)} \quad (2)$$

The sensitivity also known as the true positive rate; calculated as

$$sensitivity = \frac{TP}{(TP + FN)} \quad (3)$$

And lastly the precision also known as the positive predictive value calculated as

$$precision = \frac{TP}{(TP + FP)} \quad (4)$$

For these three equations T stands for true, F for false, P for positive and N for negative. Havaei, et al. report their data using specificity while Kamnitsas, et al. reports results using precision. Larger numbers are better and can be reported either as 0 to 1 or 0 to 100.

Name	Dice	Specificity	Sensitivity
InputCascadeCNN*	0.84	0.88	0.84
Tustison	0.79	0.83	0.81
Zhao	0.79	0.77	0.85
Meier	0.72	0.65	0.88
Reza	0.73	0.68	0.79
Cordier	0.75	0.79	0.78

Table 1: Comparison of Havaei, et al's. (highlighted) results on BRATS 2013 leaderboard set.

As mentioned before, the work done by Havaei, et al. was applied to the BRATS 2013 challenge. They also attempted to use their system on the BRATS 2014 data set, but were unable to get the system to work due to problems with the data set itself. For the BRATS 2013 Challenge contenders used an online evaluation tool that provided Dice, specificity and sensitivity scores for three categories; the complete tumor region, the core tumor region and the enhancing tumor region. As can be seen in Table 1 their highest scoring network configuration improved on the state of the art in both accuracy and speed.

The architectures proposed by Kamnitsas, et al. were applied to three different brain related challenges. The first was a challenge involving a database of MRIs from people who had suffered traumatic brain injuries, TBI. The second was on the BRATS 2015 dataset, which much like BRATS 2013, measured dice, specificity and sensitivity on the three categories of brain tumor hierarchies; the results can be seen in Table 2. In these results the top score by Ensemble was a combination of three similar DeepMedic networks; their output aggregated by averaging. The exact nature of these networks was undefined.

The last challenge Kamnitsas, et al. did was the ISLES 2015 challenge that dealt with the segmentation of brain lesions caused by stroke. In all cases the DeepMedic + CRF architecture they propose beat the scores of the previous state-of-the-art approaches. Additionally, in all cases the addition of the 3D CRF was found to have a statistically significant effect on performance.

Figure 7 shows an example [3] of DeepMedic's segmenta-

Name	Dice	Precision	Sensitivity
Ensemble+CRF	90.1	91.9	89.1
Ensemble	90.0	90.3	90.4
DeepMedic+CRF	89.8	91.5	89.1
DeepMedic	89.7	89.7	90.5
bakas1	88	90	89
peres1	87	89	86
anon1	84	90	82
thirs1	80	84	79
peyrj	80	87	77

Table 2: Average performance of Kamnitsas, et al. on the training data from BRATS 2015 compared to other teams. Highlighted cells are networks implemented by Kamnitsas, et al.

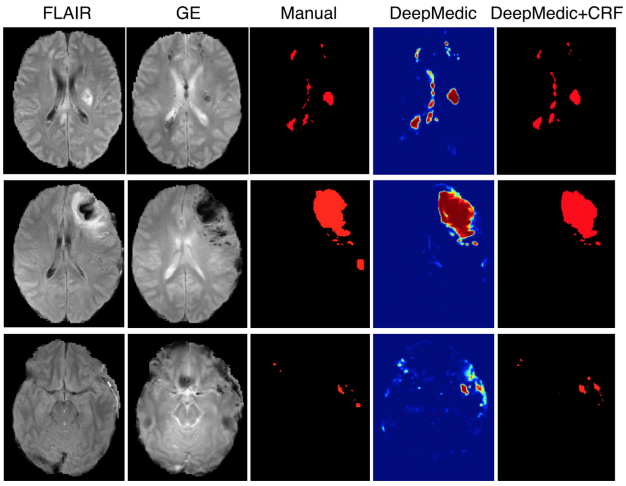


Figure 7: Three examples from the TBI dataset. The first two columns show the original MRIs with the third column showing the images segmented manually and the last two columns showing the segmentation performed by DeepMedic and DeepMedic with the CRF. Taken from Kamnitsas, et al. [3]

tion of three different MRIs. As can be seen in the figure the network does a good job of segmenting both small and large lesions. In the second row however, it undersegments a contusion, possibly mistaking it for background. The third row also shows one of the greater challenges of this segmentation task, where post-surgical sub-dural debris is mistaken as a relevant lesion. [3]

5. CONCLUSIONS

As demonstrated by the work of Havaei, et al. and Kamnitsas, et al. convolutional neural networks show considerable promise. The use of multiple pathways for local and global context was shown to be effective in both works. Havaei, et al. has further shown that cascaded architectures with two-phase training are also useful in improving the state-of-the-art. Meanwhile Kamnitsas, et al. have shown that 3D CNNs have become computationally feasible and that dense inference and 3D CRFs show promise as well.

These two works are unfortunately not comparable when it comes to the challenges they partook in, but nevertheless both show a breadth of space for further improvement when it comes to CNNs and medical imaging. Future work could combine some of the techniques used such as using a cascaded architecture with DeepMedic or attempting two-phase training.

Acknowledgments

Thank you to Elena Machkasova and Nic McPhee for their guidance through this process.

6. REFERENCES

[1] A. Deshpande. A beginner's guide to understanding convolutional neural networks, Jul 2016.

[2] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18 – 31, 2017.

[3] K. Kamnitsas, C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker. Efficient multi-scale 3d {CNN} with fully connected {CRF} for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61 – 78, 2017.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] A. Payan and G. Montana. Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *CoRR*, abs/1502.02506, 2015.

[6] A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, and M. Nielsen. *Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network*, pages 246–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[7] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, and R. M. Summers. A new 2.5d representation for lymph node detection using random sets of deep convolutional neural network observations. *CoRR*, abs/1406.2639, 2014.