CNN BASED SEGMENTATION OF MEDICAL IMAGING DATA

IE643 (DEEP LEARNING: THEORY & PRACTICE) COURSE PROJECT

TEAM – METANET: RAJ VHORA (180110058) & SHIVPRASAD KATHANE (180110076)

OUTLINE

- Introduction & Problem Statement
- Background & Related Work
- Important Past Works
- Dataset Understanding
- Solutions to the Problems
  - Network Architecture
  - Loss Metric
- Experiments/Tasks
- Results
- Conclusions & Challenges
- Project Status
  - Data Acquisition
  - Training
  - Architecture & Experiments
- Improvements
- References

INTRODUCTION & PROBLEM STATEMENT

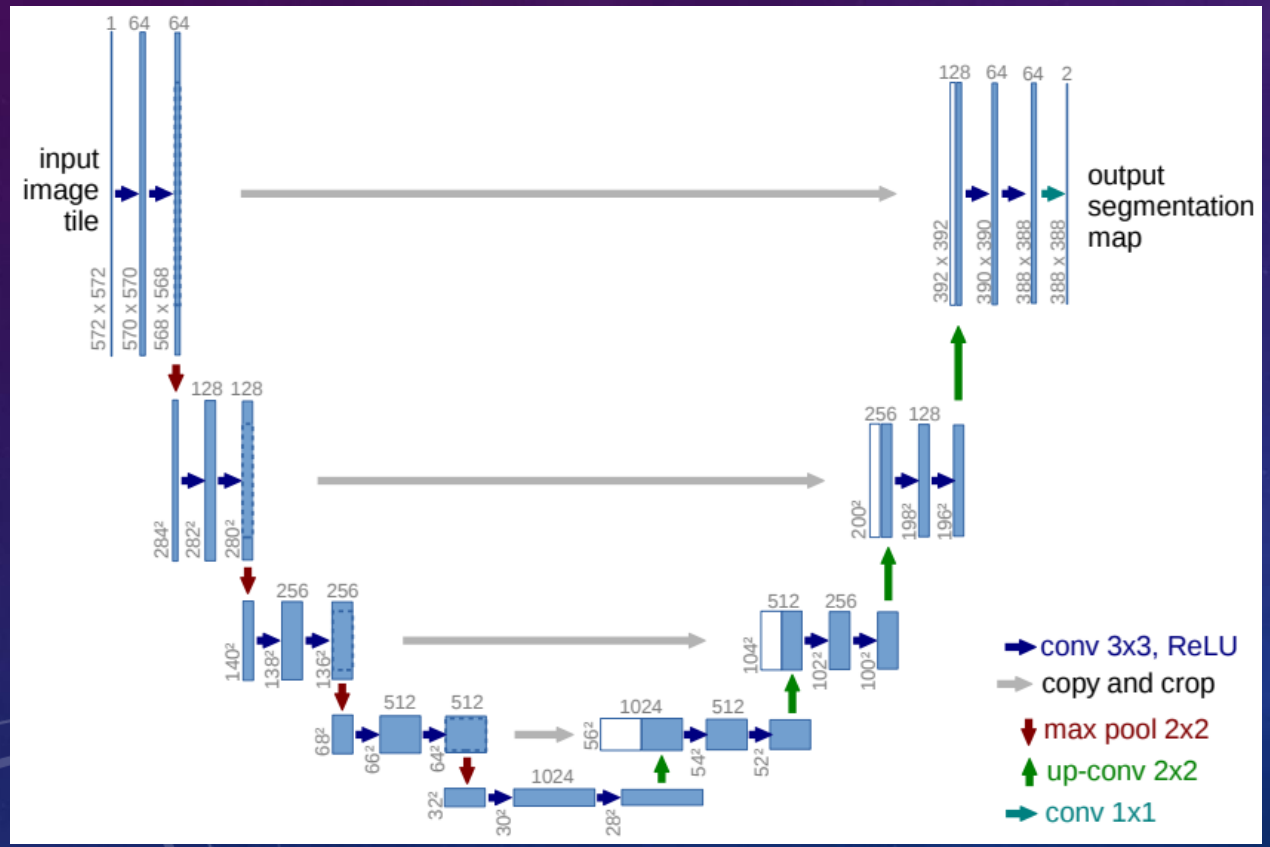
- Convolutional Neural Networks (CNNs) are used in wide variety of computer vision tasks. Recent advances in semantic segmentation have enabled their application in medical image processing.
- Medical images may come from a variety of imaging technologies such as computed tomography (CT), ultrasound, X-ray and magnetic resonance imaging (MRI). They are used in the depiction of different anatomical structures throughout the human body, including bones, blood vessels, vertebrae and major organs.
- As medical images depict healthy human anatomy along with different types of unhealthy structures (tumors, injuries, lesions, etc.), segmentation commonly has two goals: delineating different anatomical structures (such as different types of bones) and detecting unhealthy tissue (such as brain lesions).
- The paper given, discusses a CNN-based medical image segmentation method on hand and brain MRI, with the tasks of bone and tumor segmentation respectively.
- Certain aspects of medical images call for modifications on designs that are created for semantic segmentation, most notably the increased memory demands of 3D images and the imbalance between labels found in 3D ground truth data.

BACKGROUND & RELATED WORK

- Typical classifiers like AlexNet, VGGNet, etc read in image and output a set of class probabilities. Moreover, to get a semantic segmentation of the image, classification needs to be done on every pixel.
- This segmentation goal can be achieved by applying classifiers to patches of input image in a sliding window fashion. However, this method suffers from redundant computation. Moreover, it learns features that are local in the patches and can't learn global features as the patch size goes down.
- Replacing fully connected layers with 1x1 filters allows for multiclass class prediction of multiple pixels in a single forward pass. This technique is efficient when the size of input is larger.
- Since the Convolutional layers and pooling layers decrease the size of input layers, the resolution of output is often lower. This can be tackled by padding of input images.
- Proposals for creating a CNN that generates a segmentation map for an entire image include up-sampling the down-sampled feature maps back to original size using deconvolution operations.
- Alternative way of up-sampling feature maps is by storing the indices of activations that were preserved by max-pooling layers and using them later for up-sampling.
- Coupling a CNN with a Conditional Random Field (CRF) to enhance segmentation results is also suggested.

U-Net: Convolutional Networks for Biomedical Image Segmentation

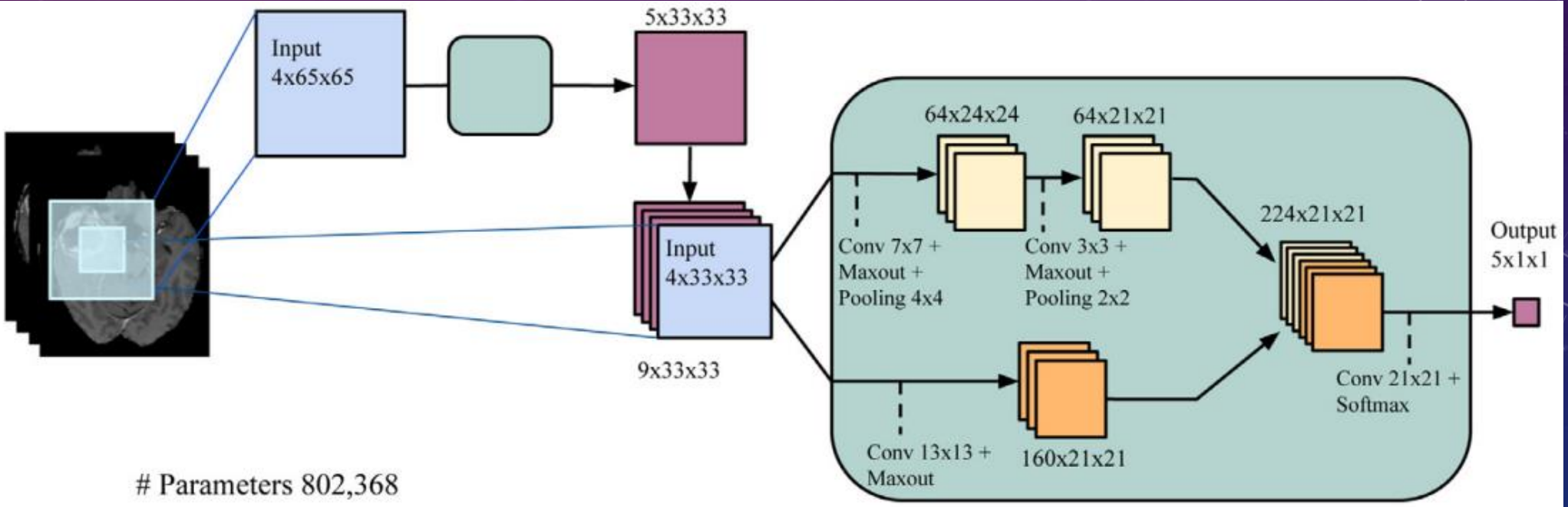
[3]



It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 upconvolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

Brain tumor segmentation with Deep Neural Networks

[2]



(a) Cascaded architecture, using input concatenation (INPUTCASCADECNN).

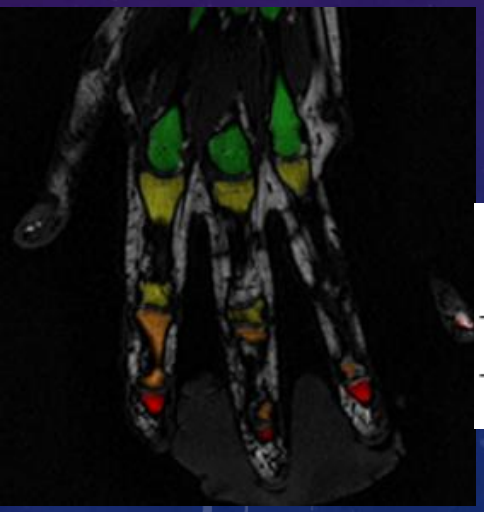
- Cascaded architecture: It is desired for the ultimate prediction to be influenced by the model's beliefs about the value of nearby labels, so they propose to feed the output probabilities of a first CNN as additional inputs to the layers of a second CNN using concatenation.
- Two-Phase Training to take care of class imbalance: Construct patches data-set such that all labels are equiprobable + Retrain only the output layer with a more representative distribution of the labels
- Metrics: Dice, Sensitivity, Specificity

UNDERSTANDING OF DATA

1. Hand MRI - 30 MRI volumes of a hand in different postures
   - Image Size: 480 x 480 x 280 downsampled & cropped to 120 x 120 x 64
   - Labels: Bone Type (phalanx) = metacarpal (1), proximal (2), middle (3), distal (4)
2. BRATS 2015 - 274 MRI images of brain with tumor (4 modalities)
   - Image Size: 240 x 240 x 155 cropped to 160 x 144 x 128 (some downsampled)
   - Labels: Tumor Type = necrosis (1), edema (2), non-enhancing (3), enhancing (4)

[1]

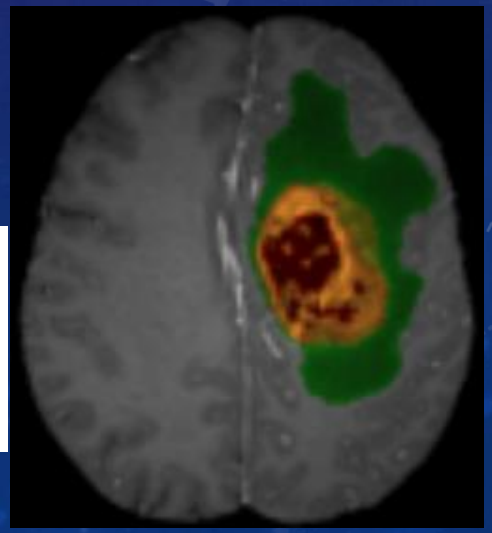
[1]



Sample images showing ground truth segmented regions (colour as per label)
Table showing class imbalance wrt frequencies of background (0) & data labels

	Average class frequency				
	0	1	2	3	4
Brain	0.96	0.002	0.024	0.005	0.006
Hand	0.99	$5.13 * 10^{-3}$	$2.29 * 10^{-3}$	$6.78 * 10^{-4}$	$4.32 * 10^{-4}$

[1]



SOLUTIONS TO ADDRESS THE PROBLEMS

- For brain and tumor segmentation, two modifications to the original U-Net were tested:
  1. Combining multiple segmentation maps created at different scales
  2. Using element-wise summation to forward feature maps across stages of the network
- The high memory demand of 3D medical images poses a difficulty. A solution is to use downsampled images whenever the memory demand becomes overwhelming. With this approach, the output of the network is a low-resolution segmentation, which can be upsampled using interpolation or superresolution. This is the training and inference scheme used in this work for images that are too large to be processed by the network.
- Loss functions and evaluation metrics have been carefully chosen to address class imbalance

NEURAL NETWORK ARCHITECTURE

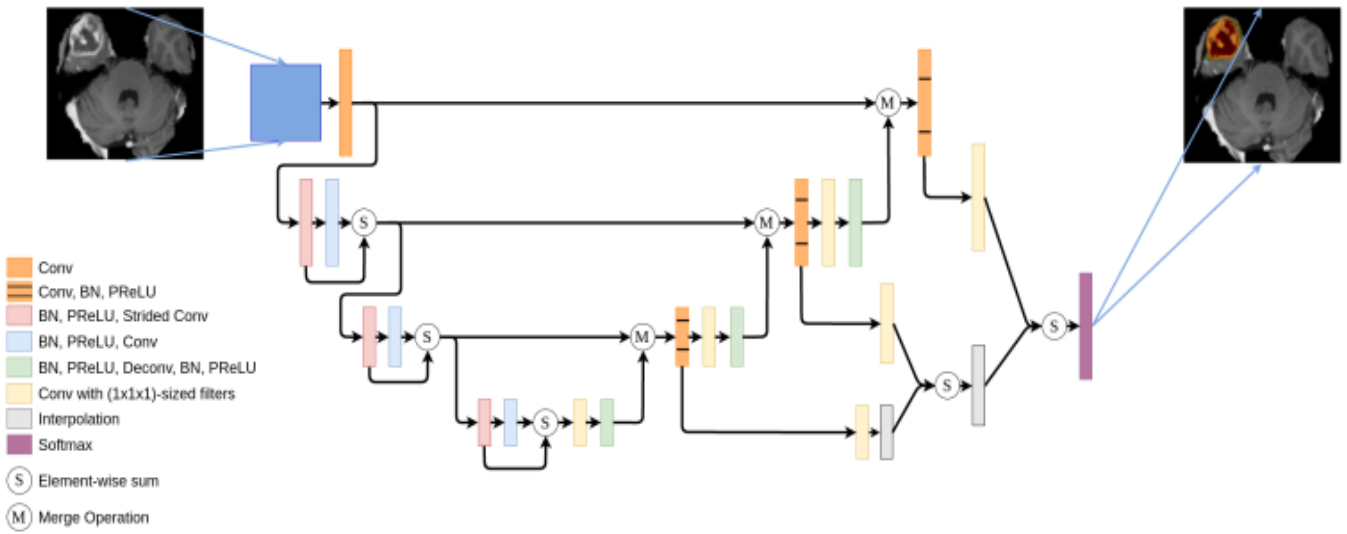


Figure 4: The basic architecture used throughout the experiments. The actual architecture is 3D. 2D is used here for the purpose of simplicity. Two different merge operations are tested for the long skip connections: element-wise addition and feature concatenation. Best viewed in electronic format.

- All convolutions used in the network have filters of size 3 × 3 × 3, with the exception of the final convolutions used to produce the segmentation maps and the ones involved in reducing the number of feature maps prior to a deconvolutional layer. Both of these use 1 × 1 × 1-sized filters. [1]

- The data for MRI, CT Scan is usually available as 3D volumes of images. Image sizes are typically 512*512*30
- If the third dimension is small, the data can be processed as a set of 2D images
- 3D convolutional layers can be used to access the volume of images to get the segmentation maps in a single forward pass if the third dimension is big enough.
- However, using 3D filters also requires increased memory.
- Throughout the network, zero-padding is used so that the size of feature maps is only changed by strided convolutions or deconvolutional operations.
- The network uses PReLU activations.
- Each activation layer is preceded by a layer of batch normalization. Strided convolution is used in place of max-pooling.

LOSS FUNCTION & PERFORMANCE METRIC

- Class imbalance is a serious issue in medical image segmentation and one that requires special care regarding the loss function used
- A loss function close to the dice similarity coefficient is used in this work for training. The dice coefficient is an overlap metric frequently used for assessing the quality of segmentation maps. Dice coefficient is given by: [1]

$$\mathrm{DSC} = \frac{2\|PT\|_2}{\|P\|_2^2 + \|T\|_2^2}$$

Where PT is the element-wise product of P and T and $\|X\|_2$ is the L2-Norm of X.

- The loss function used throughout the experiments is based on a similarity metric that is formulated very closely known as Jaccard index. It is given by: [1]

$$\mathrm{Jacc} = \frac{\|PT\|_2}{\|P\|_2^2 + \|T\|_2^2 - \|PT\|_2}$$

LOSS FUNCTION & PERFORMANCE METRIC

- Instead of maximizing Jaccard index, network is trained to minimize Jaccard distance given by (1 – Jacc).
- Limitations:
  - Samples for which the ground truth contains no foreground labels (e.g. a brain MRI scan containing no tumorous regions), the loss will be maximized, even if the output segmentation map contains very few false positives.
  - The dice coefficient and the Jaccard index are only defined for binary maps. One extension of the Jaccard index to multi-class segmentation maps is defined as: $$\text{Jacc} = \frac{\sum_i \min\{P_i, T_i\}}{\sum_i \max\{P_i, T_i\}}$$ over the set of all class labels.
  - However, in this case, some classification labels will be severely penalized as compared to other.

EXPERIMENTS / TASKS

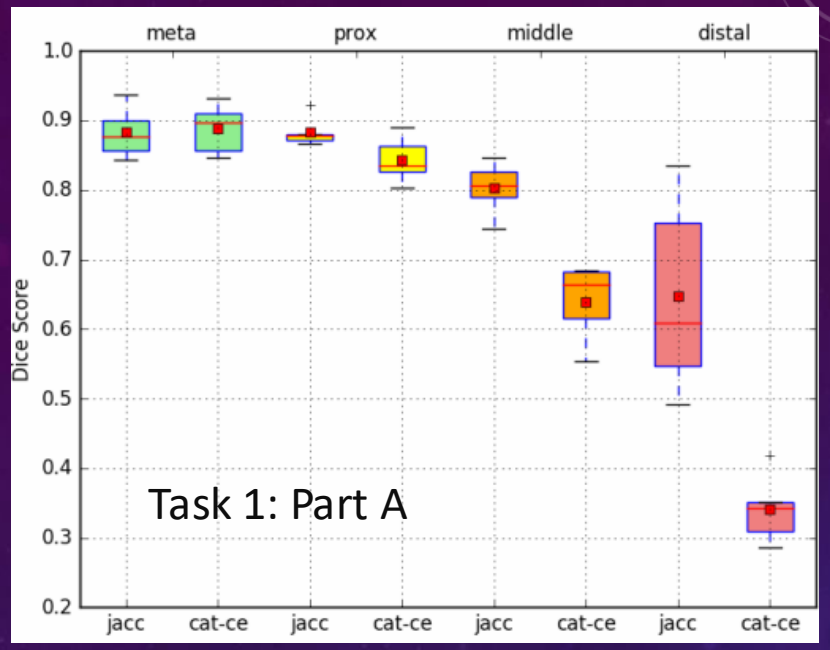
Task 1: Segmentation of bones in hand MRI

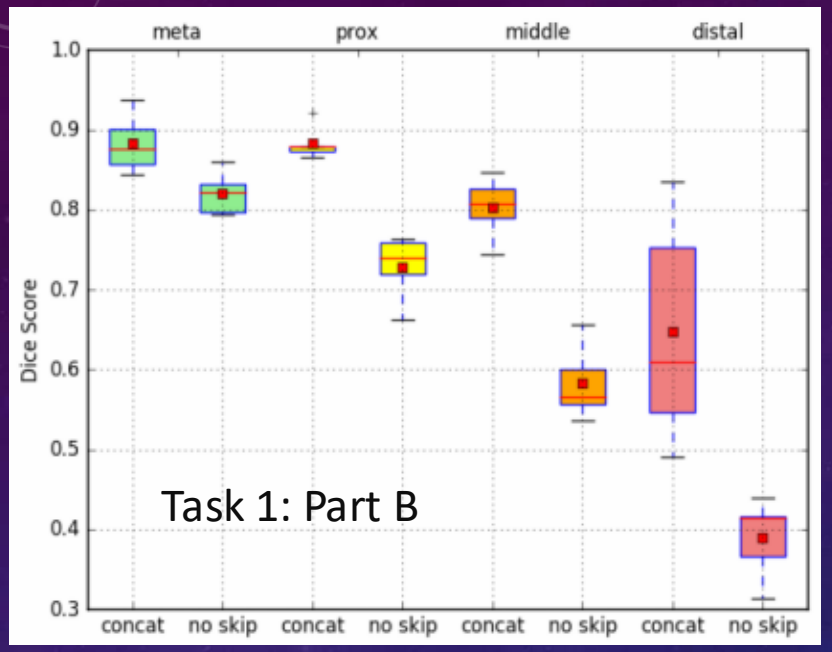
- Data Split: 20 Train + 5 Validation + 5 Test
- Data Augmentation:
  - a) Identity transformation
  - b) Flipping along random axis
  - c) Rotation by random angle
- Training:
  - a) Optimisation Technique = Adam
  - b) No of Epochs = 600
  - c) Learning Rate = 5 x 10-5
- Evaluation: Record dice scores for different labels and compare for:
  - a) Jaccard loss vs categorical cross-entropy
  - b) With vs without long skip connections @ Testing
  - c) Multiple segmentations maps vs single
  - d) Element-wise summation vs concatenation @ 6f CV

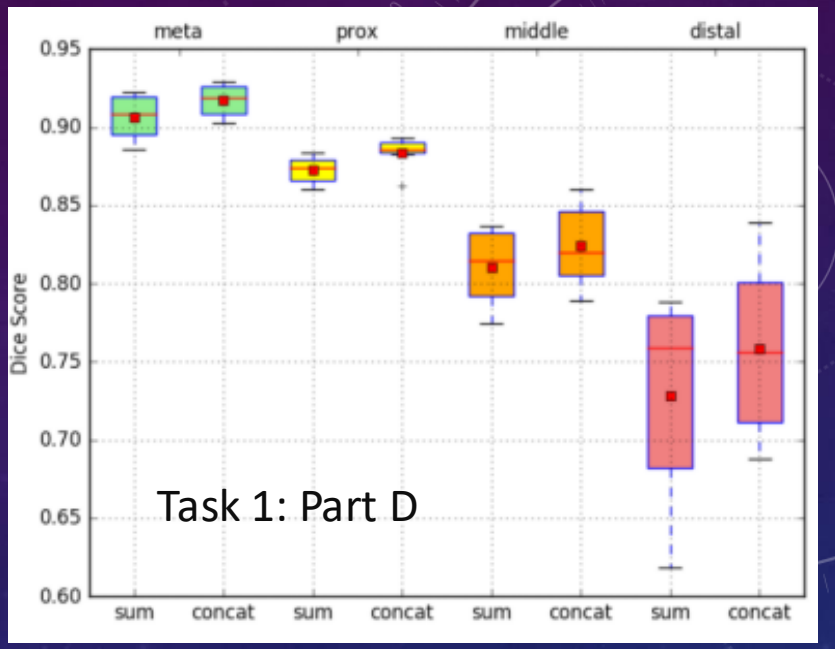
EXPERIMENTS / TASKS

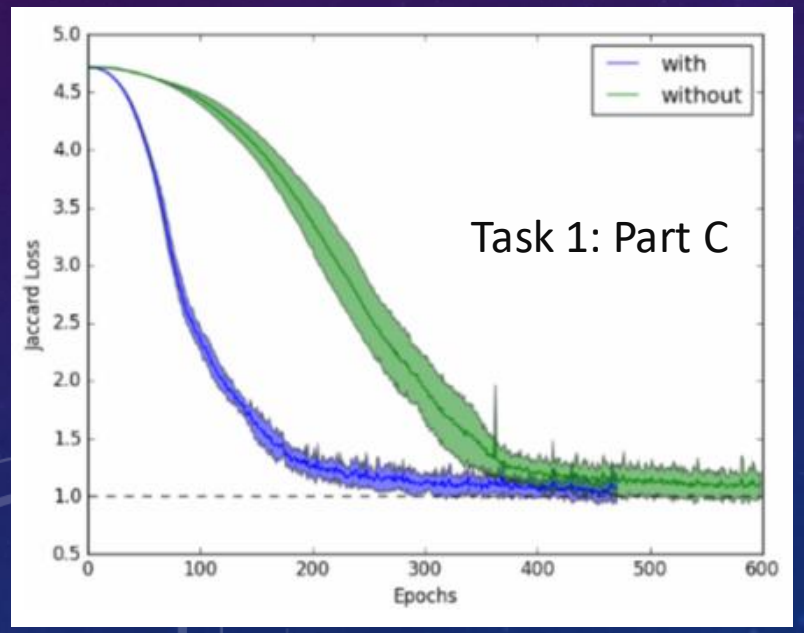
Task 2: Segmenting tumor regions in brain MRI

- Data Split: 220 Train + 27 Validation + 27 Test
- Data Augmentation:
  a) Identity transformation
  b) Flipping along random axis
  c) Rotation by random angle
- Training:
  a) Optimisation Technique = Adam
  b) No of Epochs = 50
  c) Learning Rate = 5 x 10-5
- Evaluation: Record dice scores for
  a) each modality (w/o CV)
  b) different combinations (w/o CV)
  c) identified best network (averaged over 5-fold CV)









Task 2

RESULTS for all the tasks [1]

Modalities	necrosis	edema	enhancing	non-enhancing
Flair	0.19	**0.74**	0.28	0.36
T1	0.27	0.56	0.26	0.4
T1C	**0.47**	0.61	**0.38**	**0.74**
T2	0.34	0.69	0.29	0.45
Flair, T1C	0.47	0.81	0.47	0.78
Flair, T1C, T2	**0.55**	0.82	0.49	**0.8**
all	0.49	**0.84**	**0.5**	**0.8**

CONCLUSIONS & CHALLENGES

- Combining multiple segmentation maps created at different scales sped-up convergence without hurting final performance while using element-wise summation to forward feature maps across stages produced slightly worse results as compared to the original.
- To address the issue of class imbalance, a loss function based on the Jaccard similarity index was used while the issue of high memory demand of 3D images was resolved by down-sampling.
- Removing long skip connections unambiguously worsens performance while categorical cross-entropy is far less capable of detecting infrequent classes as compared to jaccard loss.
- Despite the relative scarcity of labelled medical images, 3D CNN architectures have been capable of achieving good quality results.
- Availability and inclusion of healthy brain images data can be beneficial as it may help the network learn to explicitly identify and discriminate healthy vs tumorous regions.
- To address class-imbalance, weighting of classes, comparison of weighted-CE & other loss functions with Jaccard, cropping regions with empty space etc. should be explored.
- The network could be applied to different types of medical images depicting different structures and using different kinds of imaging technology (ultrasound, EM, CT, etc.) to check versatility.
- Using deeper architectures with more contracting and expanding blocks can be investigated.

PROJECT STATUS

Data Acquisition :

- Data NOT Available on given parent data website.
- Found the data on Kaggle after searching through data archives.
- Data in not so readily-usable format (Lots of text files, License files, File structure is not straight forward).
- Lots of file with same name (but different data sizes) (approx. 4000 such files)
  - This made downloading data from kaggle and unzipping it on local machine difficult. Over and above, dataset is also huge. The archive size is ~5.6 GB.
  - This also meant that porting data from kaggle to Google Colab is difficult.
- Decision taken : Continue with Kaggle dataset and preprocess it in kaggle. Store the preprocessed data in Numpy arrays and download them to upload on Google Colab

PROJECT STATUS

Training :

- Major limitations on RAM/ Compute:
  - Local Machine has limited RAM and Compute, but infinite storage and infinite running time, but cannot access original dataset
  - Kaggle has dataset, limited RAM, good enough storage
  - Google Colab has very good RAM, good enough storage, excellent compute power, but can't access original data source
- Decision : Pre-process dataset on Kaggle, upload NumPy arrays on Google Colab to process/train. (If data size is small, we can also use Kaggle)
- Code limitations:
  - The code by the paper was developed 5 years ago in Theano and python 2.7 along with other libraries. The code also assumes a certain set file structure for training.
  - Since, the files could not be editted in either Colab or Kaggle, we decided to design our own code that can resemble the code by paper authors to the maximum levels.

PROJECT STATUS

Network Coding and Experiments: (Improvements thought of in brackets)

- Started by deciding pre-processing technique: Subtraction of mean from all pixels and division by standard deviation. Instead of predicting 4 categories right now itself, we have made a binary classification network by clubbing categories 1,2,3 together.
- Saving pre-processed data in NumPy arrays. (This eats away RAM, a data generator function can be designed)
- Small Image size. The image is resized to (32,32,32) and later experimented with (96,96,96). (Original reduced size couldn't be done as doing so caused RAM to exhaust and Kernal got restarted. This issue might get solved after data generators are designed)
- Network Architecture : A small UNet architecture of 20 layers in total is designed. Total of ~69000 learnable parameters. (There are a lot of networks with description in code by original authors. Our architecture is simplest amongst them. More complex architectures can be implemented.)
- Loss Function: As there are only two categories, we have used binary cross entropy loss function. We will move to loss function used in Paper as we shift from 2 categories to 4 categories.

IMPROVEMENTS / SUGGESTIONS

- Comparing Loss Functions: We plan to investigate the effect of other different loss functions relevant for segmentation such as Focal loss.
- Use of newer/different datasets: We plan to explore applying the network to datasets from latest BRATS challenges, other medical image datasets and check the segmentation performance.
- Deeper networks: We plan to experiment with deeper NN architectures and tabulate results for varying NN hyperparameters.
- As the paper is from 2017, we hope to find more ideas from the latest works which have cited it and work upon them.

REFERENCES

- Papers :

[1] Kayalibay, Baris, Grady Jensen and Patrick van der Smagt. “CNN-based Segmentation of Medical Imaging Data.” *ArXiv* abs/1701.03056 (2017): n. pag.

[2] M Havaei, Axel Davyet al., "Brain tumor segmentation with Deep Neural Networks", Medical Image Analysis, Volume 35, 2017, Pages 18-31

[3] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv*. https://doi.org/10.48550/arXiv.1505.04597

- Github Repo :
  - https://github.com/BRML/CNNbasedMedicalSegmentation
- Data :
  - https://www.kaggle.com/datasets/xxc025/brats2015
- Links provided as help :
  - https://neptune.ai/blog/image-segmentation
  - https://viso.ai/deep-learning/image-segmentation-using-deep-learning/