COVID-19 Lung CT Lesion Segmentation Challenge

Yaffa T Atkins
Data Science
The Graduate Center, CUNY
tziporahatkins@gmail.com

You Jin Chung
Data Science
The Graduate Center, CUNY
yjc433@nyu.edu

Rubiya Tasnim
Data Science
The Graduate Center, CUNY
r.tasnim16@gmail.com

Nicholas Dropp
Data Science
The Graduate Center, CUNY ndropp92@gmail.com

Abstract

Machine learning in the biomedical field has and will continue to be extremely useful in the coming years. Machine learning methods for automatic detection and quantification of COVID-19 lesions in infected patient's lungs will play a vital role in this pandemic. We implemented a neural network called nnU-Net to semantically segment lesions in lungs caused by COVID-19 for early detection and management of the virus. This paper has a complete overview of the implementation of nnU-Net network on COVID-19 lung lesion data sets. Our network had performed comparatively comparatively as the sample submitted by the host of the GrandChallenge.org.

1 Introduction

Covid-19 has been a devastating tragedy with nearly three hundred million cases worldwide [1]. Covid-19 virus affects the lung in the patient along with other organs. Since the beginning of pandemic, imaging the lung has been critical for early detection and management of the treatment for the patient [2]. The size and distribution of the lesion in the lungs caused by the virus may vary by age and severity of the patient's disease. Since the early days of the pandemic, lung imaging has been critical for both the early identification and management of individuals affected by Covid-19[2]. Machine learning and image segmentation models have been developed to detect lesions due to covid in a patient's lungs. One of these models is nnU-Net, a self-configuring network that successfully segments images with very little user-interference. In this report, we discussed implementing a nnU-Net network to automatically segment lung lesions in CT (chest computed tomography), scan images in Covid-19 patients and evaluate the performance of this neural architecture.

2 Literature Review

The wildly popular U-Net [3] structure is a successful encoder-decoder network. The encoder part of U-Net is a similar traditional classification CNN where semantic information is aggregated at the expense of reduced spatial information. The decoder of U-Net recovers the missing spatial information by recombining higher resolution feature maps obtained directly from the encoder through skip connections with semantic information from the bottom of the "U" as shown in figure (1)[3].

The structure of the U-Net enables segmentation of fine structures very well since both semantic as well as spatial information are crucial for the successful network to perform segmentation tasks.

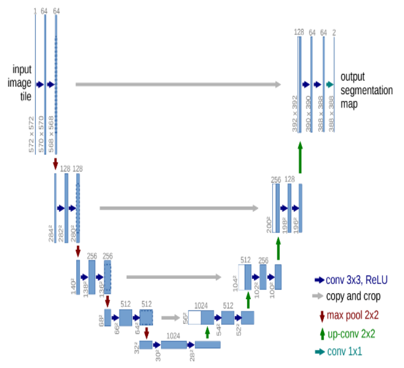


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Even though the architecture and method of U-Net are quite straightforward and commonly used as a benchmark, the remaining interdependent choices regarding the exact architecture, preprocessing, training, inference, and post-processing are quite often the cause of underperformance of U-Net when used as a benchmark [3]. The influence of these non-architectural aspects in segmentation methods is as impactful for the success of the network [3].

The nnU-Net ("no-new-Net") was proposed to condense the current domain knowledge and autonomously implement the key decisions required to transfer a basic architecture to different datasets and segmentation tasks without manual tuning. The nnU-Net framework was built on a set of three comparatively simple U-Net models with minor modifications to the original U-Net. The nnU-Net uses two plain convolutional layers between poolings in the encoder and transposed convolution operations in the decoder, just like the original U-Net. In nnU-Net architecture, ReLU activation was replaced with leaky ReLU (neg slope 1e 2), and instance normalization was used instead of batch normalization [3].

This nnU-Net framework decides all the steps by adapting its architecture to the given image geometry [3]. The nnU-Net enables out the box image segmentation by adapting to arbitrary datasets on accounts of two key contribution-1) formulation of a data fingerprint (representing the key properties of a dataset) and a pipeline fingerprint (representing the key design choices of a segmentation algorithm) and 2) condensing domain knowledge into a set of heuristic rules that robustly generate a high-quality pipeline fingerprint from a corresponding data fingerprint while considering associated hardware constraints[4] . This framework, as shown in figure 2 [4] had outperformed in many international challenges on very diverse biomedical datasets without manual intervention [4].

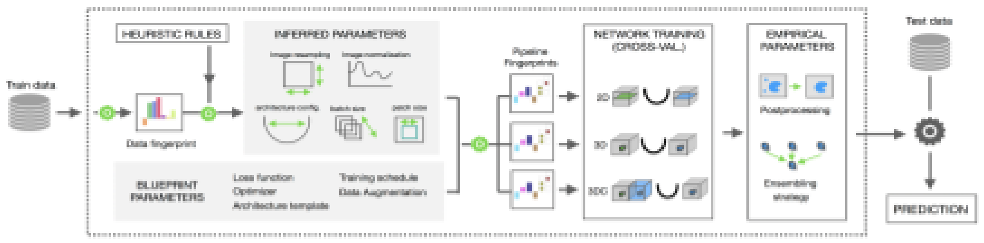


Figure 2: Automated configuration by nnU-Net: Dataset properties are summarized in a "dataset fingerprint". A set of heuristic rules operates on this fingerprint to infer the data-dependent hyperparameters of the pipeline. These are completed by blueprint parameters, the data-independent design choices to form "pipeline fingerprints"[4].

3 Methodology

We connected our python notebook with google drive and set up the base directory as the Colab Notebooks folder. We cloned the GitHub repository that we will be using and imported all the necessary packages. Our algorithm and files will be stored in the GitHub repository folder nnUNet.

a) Folder structure**:**

nnU-Net requires a specific folder structure and specific file names and formats so that nnU-Net will know how to access and interpret the data. nnUNet_raw_data_base/nnUNet_raw_data is where nnU-Net finds the raw data. Each dataset is stored as a separate task associated with its task ID and task name. We created a folder called Task101_COVID for our project. In that folder, we created a train image directory imagesTr where the training images are stored, a training labels directory labelsTr where the training labels are stored, and a test images directory imagesTs where images for the testing phase are stored. We created environment variables nnUNet_raw_data_base, nnUNet_preprocessed, and RESULTS_FOLDER to access raw data, pre-processed data, and trained model weights, respectively. [9]

```
nnUNet_raw_data_base/nnUNet_raw_data/Task101_COVID/
├── dataset.json
├── imagesTr
│   ├── fileName_003_0000.nii.gz
│   ├── fileName_004_0000.nii.gz
│   ├── ...
├── imagesTs
│   ├── fileName_001_0000.nii.gz
│   ├── fileName_002_0000.nii.gz
│   ├── ...
└── labelsTr
    ├── fileName_003.nii.gz
    ├── fileName_004.nii.gz
    ├── …
```

Figure 3: Proper folder structure when working with nnU-Net

b) Unzip Rename and Split the Training and Testing Files**:**

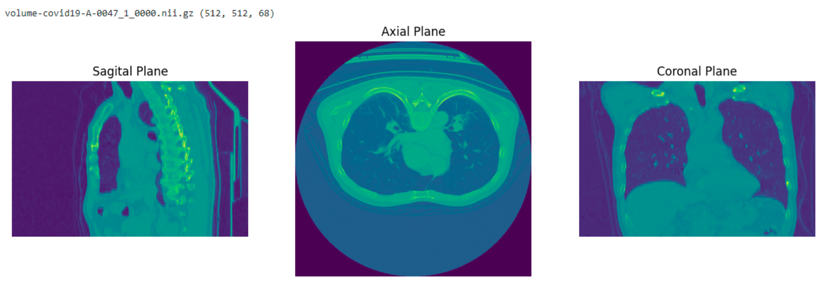
Manually put the train and test data in a zip folder within the Task101_COVID folder. The algorithm unzips the files and renames the training image and label files so that they match, removing the _ct or _seg from the file name, and giving corresponding training images and labels an identifier so nnU-Net recognizes them as pairs. Training image and label files are then placed in the proper directory. We manually put all the testing images in the proper directory. Modality refers to different types of images, such as CT scans versus different types of MRIs, and so on. Imaging modalities are identified by nnU-Net by the last four digits at the end of the filename. Since our data only has one modality, we do not need to check the imaging modality. Instead, we simply renamed all training images by adding _0000 at the end before .nii.gz because nnU-Net expects images with that name format that specifies the image's modality. Next, the algorithm writes the dataset.json, which is used for verification and training. Dataset.json stores essential properties of the images such as modality, which in this case, as mentioned, all images share the same modality: CT scan. Dataset.json also stores the labels that will be segmented. Here there are two labels, background, and the actual lesion.. Dataset.json also stores the amount of test and training images, and the names of all the test and train files without their modality included in the file name. The proper directory format with the proper files stored in the proper folders can be seen in Figure 3. [10] Now that the data is all formatted and structured correctly, we can call built-in nnU-Net commands.

c) Data verification**:**

We call the nnU-Net verification function
experiment_planning/nnUNet_plan_preproccess.py
This function populates nnU-Net/preprocessed/Task101_COVID with subfolders of preprocessed training data 2D U-Net and all applicable 3D U-Net training. This command also creates 'plans' files that end in .pkl for the 2D and 3D configurations, and they contain generated segmentation pipeline configuration that the trainer will read in. We also called the verify_dataset_integrity command, which ensures the data is in the proper format and is compatible with nnU-net [11].

d) Data visualization**:**

We then printed a visual example of the training data, labels of the training data, and testing data. All three planes are displayed, the sagittal plane, which divides the body into left and right, the axial plane which divides the body into top and bottom halves, and the coronal plane which divides the body into front and back. As discussed, there are only two labels, background and lesions, so the labels images only have two colors, the purple background and the yellow lesions. Training images, their labels, and testing images are displayed in figures 4, 5, and 6 respectively.



```
volume-covid19-A-0047_1_0000.nii.gz (512, 512, 68)
```

Figure 4: Training images, these images are an example of what the CT scans look like

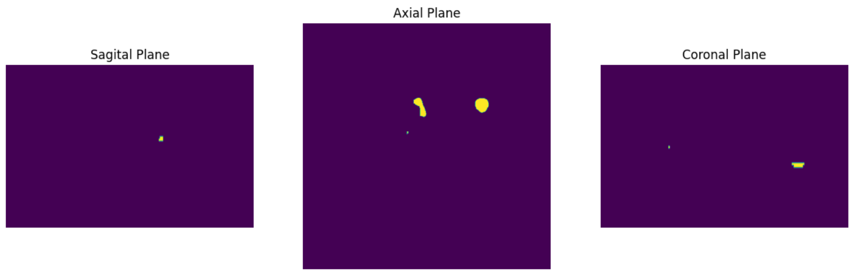


Figure 5: Training labels: labels of the covid lesions for the lung CT scans for the training data.

volume-covid19-A-0584_0000.nii.gz (512, 512, 49)

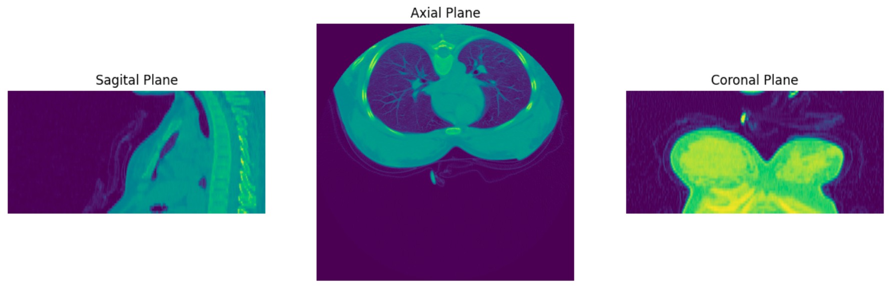


Figure 6: Testing data, there are no labels; those will be predicted.

e) Training the model:

We then trained our model. nnU-Net trains all U-Net configurations in 5-fold cross-validation to determine post-processing and ensembling on the training data. The format of the command is nnUNet_train CONFIGURATION TRAINER_CLASS_NAME TASK_NAME_OR_ID FOLD. Configuration identifies the requested configuration, the trained class name is the name of the model trainer, task name or id specifies the dataset being trained, and fold specifies the fold of the 5-fold-cross-validation being trained. The configuration we used is a 3D full resolution U-Net. The trained models are written to the results_folder/nnUNet folder. This nnU-Net training method automatically saves a progress image that is updated each epoch (figure 9) to display the results of the training and an image depicting the network architecture (figure 7) [11].

f) Inference:

Now the trained models can be applied to test images for inference. Call this built-in nnU-Net function !nnUNet_predict -i INPUT_FOLDER -o OUTPUT_FOLDER -t TASK_NAME_OR_ID -m CONFIGURATION and store the results in a predicted results folder. By default, inference will be performed using all 5-folds. Lastly, we printed those predicted results [11].

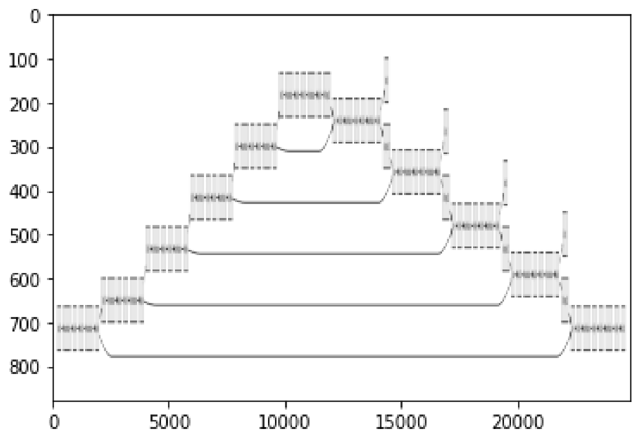


Figure 7: This figure shows the architecture of the trained network

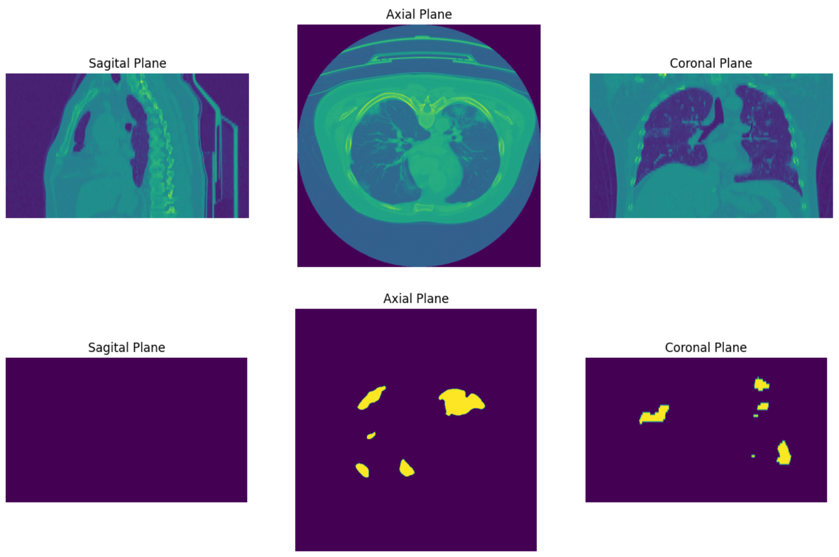


Figure 8: Here is an example of test images and the labels predicted by our model.

4 **Experimentation**

4.1 Description of the dataset**:**

Data utilized in this model was from two public resources on chest CT images, namely the "CT Images in COVID-19" 34,35 (Dataset 1) and "COVID-19-AR"36 (Dataset 2) available on The Cancer Imaging Archive (TCIA) [6]. CT (chest computed tomography) images were acquired without intravenous contrast enhancement from patients. While dataset 1 was acquired from China, Dataset 2 originated from the US Population. In total, there were 295 images, including 272 images from Dataset 1 and 23 images from Dataset 2. Of these images, 199 and 50 from Dataset 1 were used for training and validation, respectively. The test set contained 46 images in total (23 images each from Datasets 1 and 2). All lung lesions related to COVID-19 in these images were automatically segmented by a previously trained model [7, 8]. A board certified radiologists manually adjudicated and corrected these segmentations. The annotation tool used shows multiple reformatted views of the CT scans that enables manipulations and corrections of the initial automated segmentation results in three dimensions [8]. Both the image and label files are in Nifti file format. Nifti stands for Neuroimaging Informatics Technology Initiative. Image files can have any scalar pixel type and label files have segmentation maps with consecutive integers starting at 0 being background and each following number referencing a different segmented class.

4.2 Performance measurement:

The loss function used here is called the Dice Loss function. It is based on the Dice coefficient which is essentially a measure of overlap between two samples. The Dice loss value can range from 0 to 1, with 1 indicating complete overlap and

0 indicating no overlap. Dice is calculated as (2|A∩B|) / (|A|+|B|) which is the common elements between sets A and B multiplied by two, divided by the number of total elements in A and B. The common elements between A and B can be approximated as the sum of the matrix produced by the element-wise multiplication between the predicted and target mask. Since the target mask is binary, any pixel not in the target mask is zero in the multiplication matrix and not included in the final sum. Also low confidence predictions lead to smaller numbers in the multiplication matrix, leading to a smaller numerator and dice score. The numerator is multiplied by two because the denominator double counts the elements that exist in both sets. [12]

4.3 Validation process:

We trained our model with a combination of dice and entropy loss. The validation process was done using 5-fold cross-validation on the training set. Adam optimizer with learning rate 0.0009476 was used. We ran the network for about 160 epochs, but at epoch 57, we reached best performance. As shown in figure 9, we can see that further away from 57 epochs, the network might run into overfitting or underfitting issues. We terminated our model at 57 epochs. The model took 10 hours to be trained.

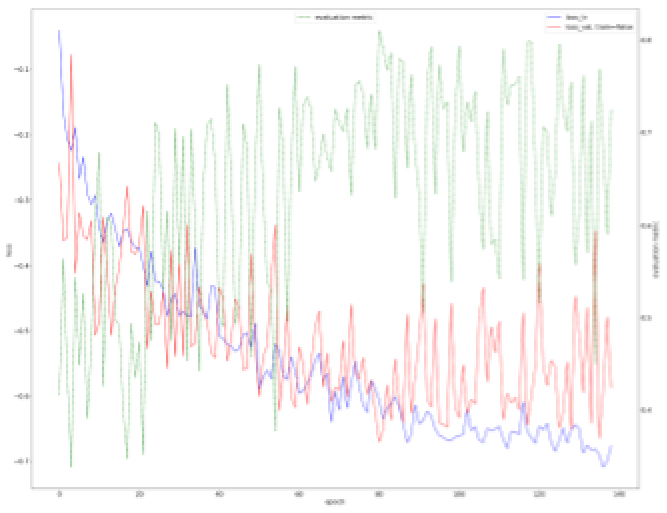


Figure 9: shows the performance of the network. With increasing epochs, the training loss and validation loss decreases and evaluation metric which is calculated as the dice loss decreases meaning accuracy improves. Beyond epoch 57, the evaluation metric reaches ,and starts decreasing and becomes unstable.

5 Results

As seen in Figure 9, our model reached its optimal performance at around epoch 57 where loss is at its lowest point. After that, validation loss starts getting higher than the training loss which means the model is beginning to overfit the errors and will no longer be accurate. Our trained model is able to predict labels similar to the ground truth labels. As seen in figure 10, image a is the CT scan and images b and c appear quite similar with c depicting the actual segmentation labels of the CT scan and b depicting the predicted labels that our model has generated. Table 1 displays the final results in number. Our train loss at optimal training is -0.5747 while validation loss is -0.4634. Each epoch took 616.8 seconds to train, and the learning rate we used is 0.0009476. Overall, our model reached an average global foreground dice score of 0.4958.

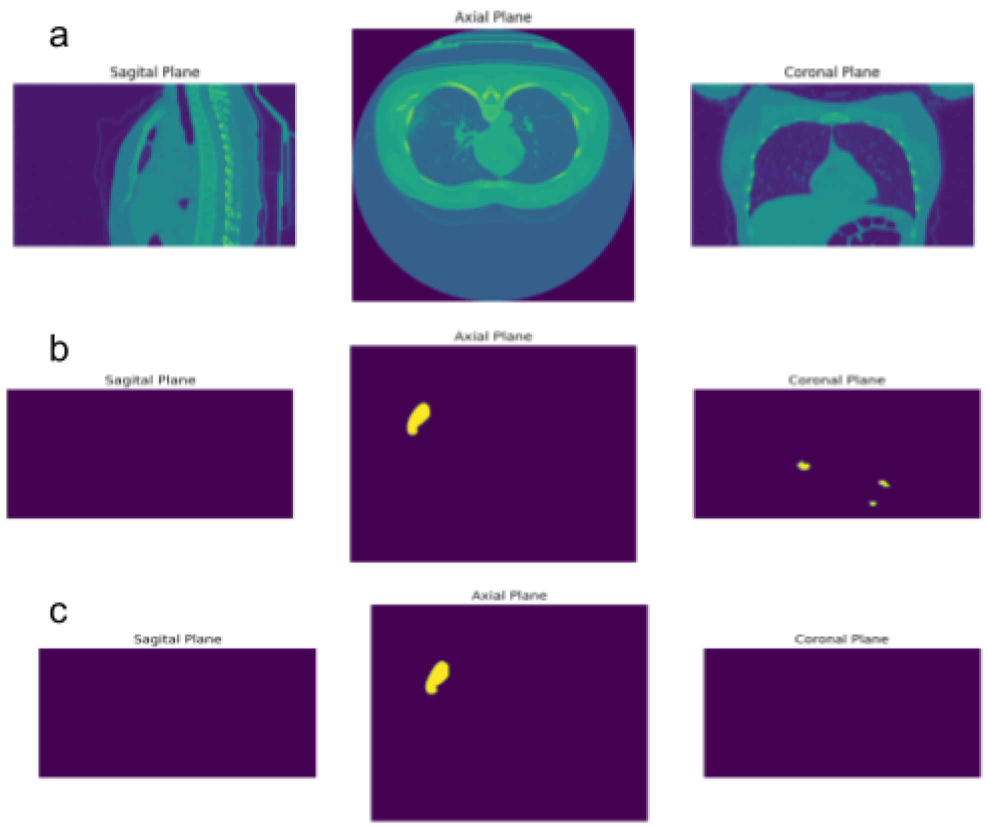


Figure a) test image file
Figure b) out model prediction
Figure c) sample prediction from the challenge host

Figure 10: Shows how predicted labels produced by our trained model look very similar to actual labels.

Best Epoch	57
Train loss	-0.5747
Validation lost	-0.4634
Average global foreground dice	0.4958
Train duration per epoch	616.8 seconds
Learning Rate	0.0009476

Table 1: shows the result of our experiment

6 **Discussion and conclusion**

Our neural network did an excellent job at predicting the lesions on the lung without manual intervention. However, training the network was computationally challenging since the task required 12+ hours to train. The network used the dice loss function and entropy loss to calculate the losses during training. The dataset was comparatively small (only 199 CT images for training and 50 images for validation), due to having less access to public data. Using nnU-Net was significantly easier than other existing networks since this framework automatically adapts itself to the geometry of the given datasets and already has all the steps calculated. This framework did not require any hyper tuning of the parameters as U-Net. The network also did not require us to identify lesions manually. We believe that our network would perform even better if we provide our nnU-Net with more data.

Reference

[1] *COVID Live - Coronavirus Statistics - Worldometer*. (n.d.). Retrieved December 18, 2021, from https://www.worldometers.info/coronavirus/

[2] Roth, H., Xu, Z., Diez, C. T., Jacob, R. S., Zember, J., Molto, J., Li, W., Xu, S., Turkbey, B., Turkbey, E., Yang, D., Harouni, A., Rieke, N., Hu, S., Isensee, F., Tang, C., Yu, Q., Sölter, J., Zheng, T., … Linguraru, M. (2021). Rapid Artificial Intelligence Solutions in a Pandemic - The COVID-19-20 Lung CT Lesion Segmentation Challenge. *Research Square*. https://doi.org/10.21203/RS.3.RS-571332/V

[3] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in MICCAI. Springer, 2015, pp. 234–241.

[4][8] Isensee, F., Jaeger, P.F., Kohl, S.A.A. et al. "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation." Nat Methods (2020). https://doi.org/10.1038/s41592-020-01008-z

[5] Isensee, F., Jaeger, P. F., Kohl, S. A. A., Petersen, J., & Maier-Hein, K. H. (n.d.). *Automated Design of Deep Learning Methods for Biomedical Image Segmentation*.

[6] Chest Imaging with Clinical and Genomic Correlates Representing a Rural COVID-19 Positive Population (COVID-19-AR) - The Cancer Imaging Archive (TCIA) Public Access - Cancer Imaging Archive Wiki. https://doi.org/10.7937/tcia.2020.py71-5978.

[7] Yang, D. et al. Federated Semi-Supervised Learning for COVID Region Segmentation in Chest CT using MultiNational Data from China, Italy, Japan. arXiv [eess.IV] (2020).

[8] Roth, H., Xu, Z., Diez, C. T., Jacob, R. S., Zember, J., Molto, J., Li, W., Xu, S., Turkbey, B., Turkbey, E., Yang, D., Harouni, A., Rieke, N., Hu, S., Isensee, F., Tang, C., Yu, Q., Sölter, J., Zheng, T., … Linguraru, M. (2021). Rapid Artificial Intelligence Solutions in a Pandemic - The COVID-19-20 Lung CT Lesion Segmentation Challenge. *Research Square*. https://doi.org/10.21203/RS.3.RS-571332/V1

[9] *nnUNet/setting_up_paths.md at master · prateekgupta891/nnUNet · GitHub*. (n.d.). Retrieved December 18, 2021, from https://github.com/prateekgupta891/nnUNet/blob/master/documentation/setting_up_paths.md

[10] *nnUNet/dataset_conversion.md at master · prateekgupta891/nnUNet · GitHub*. (n.d.). Retrieved December 18, 2021, from https://github.com/prateekgupta891/nnUNet/blob/master/documentation/dataset_conversion.md

[11]*GitHub - MIC-DKFZ/nnUNet*. (n.d.). Retrieved December 18, 2021, from https://github.com/MIC-DKFZ/nnUNet

[12] *An overview of semantic image segmentation.* (n.d.). Retrieved December 18, 2021, from https://www.jeremyjordan.me/semantic-segmentation/#loss

Appendix:

Nick: Researched image viewing, contributed toward wroting the abstract, and introduction.

Rubiya: Contributed to Literature & algorithm Research, introduction, dataset, experimentation sections, results, discussions, abstract

Yaffa: Wrote the methodology section of the report, contributed to the introduction and experiment sections, researched image segmentation algorithms.

Youjin: Dataset preparation for training, nnU-Net pipeline implementation, Nifti image data visualization, and readme.md file