Introduction

Biomedical imaging has revolutionized the field of medicine by allowing clinicians to visualize and study the human body in unprecedented detail. From X-rays to magnetic resonance imaging (MRI), these technologies have enabled doctors to make accurate diagnoses and develop effective treatment plans for their patients.

One of the key challenges is to identify and isolate specific regions of interest within these images, a process known as biomedical image segmentation. Efficient and accurate segmentation is crucial for a wide range of applications, from detecting tumors and tracking disease progression to planning surgeries and developing new treatments.

In recent years, the emergence of deep learning has revolutionized the field of biomedical imaging by providing powerful tools for accurate and efficient image segmentation. Deep learning models, such as convolutional neural networks (CNNs), have been shown to outperform traditional segmentation techniques in many applications. These models can learn complex image features directly from the data, without the need for manual feature engineering [5]. They can also adapt to new imaging modalities and deal with the variability of biomedical images by learning from large and diverse datasets. One example of deep learning-based segmentation in biomedical imaging is the use of U-Net, a CNN architecture that has been shown to perform well on a wide range of biomedical images [8]. With the continued development of AI and deep learning techniques, there is enormous potential for these technologies to further advance biomedical imaging and improve patient outcomes.

Lung Radiography Segmentation

Lung radiography is a critical tool for diagnosing and monitoring a range of lung diseases, including pneumonia, lung cancer, and chronic obstructive pulmonary disease (COPD). In the context of COVID-19, lung radiography is also an important diagnostic tool. However, accurate analysis and interpretation of radiographic images can be challenging due to the complex anatomical structures of the lungs and the presence of other organs and tissues. Accurate segmentation of the lungs from radiographic images is crucial for identifying and characterizing the severity and extent of lung involvement in respiratory diseases, including COVID-19. This can guide treatment decisions and monitor disease progression. Moreover, segmentation can enable the extraction of quantitative features from radiographic images, such as lung volume and shape, which can be used to monitor disease progression and response to treatment. Furthermore, segmentation can facilitate the development of computer-aided diagnosis (CAD) systems that can assist radiologists in interpreting images, potentially leading to faster and more accurate diagnoses. Therefore, the development of efficient and accurate segmentation methods for lung radiography is critical for advancing research in medicine and improving patient outcomes.

Machine Learning in Biomedical Imaging

Deep learning techniques involve training a neural network on a large dataset of annotated images to learn patterns and relationships within the data. These models can then be used to segment structures of interest in new, unseen images. One popular deep learning architecture for biomedical image segmentation is U-Net. U-Net is a fully convolutional neural network that has been specifically designed for semantic segmentation of images, particularly biomedical images. U-Net uses a contracting path to reduce the spatial resolution of the input image and an expanding path to upsample the image to produce a segmentation map with the same resolution as the input image [8].

U-Net has been applied to a range of biomedical imaging tasks, including lung nodule segmentation [10], liver lesion segmentation [9], and brain tumor segmentation, among others. U-Net has demonstrated exceptional segmentation accuracy, outperforming traditional segmentation techniques on several tasks.

The success of U-Net and other deep learning-based segmentation techniques has the potential to significantly improve patient outcomes and advance research in the field of medicine. Accurate segmentation of biomedical images can aid in the diagnosis, treatment planning, and monitoring of a range of diseases. Furthermore, automated segmentation can save time and reduce errors associated with manual segmentation, allowing clinicians to focus on other aspects of patient care. Therefore, developing and refining machine learning techniques for biomedical image segmentation, such as U-Net, holds great promise for the future of medicine [1].

Related works

Many architectures have been developed over the years in computer vision, some were designed specifically for biomedical segmentation like U-Net, while others were designed in other fields of computer vision and evaluated for biomedical image segmentation, with some studies comparing their performance to that of U-Net. In the following section we will present three examples of well known articles, we will later implement all of these works and compare them against each for the task of biomedical image segmentation. The works we will mention include :

- "U-Net: Convolutional Networks for Biomedical Image Segmentation" is a paper that introduced the U-Net architecture for biomedical image segmentation. The paper showed that it is possible to achieve accurate segmentation results on medical images by using a modified fully convolutional neural network architecture that consists of a contracting path and an expansive path. The U-Net architecture replaces fully connected layers with convolutional layers and adds skip connections between corresponding layers in the contracting and expansive paths, allowing the network to capture both global and local features in the image. The U-Net has since been widely adopted in biomedical image analysis tasks and has shown excellent performance in various segmentation tasks [8].

- "Fully Convolutional Networks for Semantic Segmentation" is a paper that introduced the idea of using fully convolutional neural networks for semantic segmentation. The paper showed that it is possible to perform pixel-wise classification of images by replacing fully connected layers with convolutional layers [6].

- "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation" is a paper that proposes a semantic segmentation model called DeepLabV3 based on an encoder-decoder architecture with atrous separable convolution. The model is designed to effectively capture contextual information while maintaining computational efficiency. The proposed approach achieved state-of-the-art results on several benchmark datasets, demonstrating the effectiveness of the proposed method [2].

U-Net: Convolutional Networks for Biomedical Image Segmentation

"U-Net: Convolutional Networks for Biomedical Image Segmentation" is a groundbreaking research paper that revolutionized the field of biomedical image segmentation. The paper was authored by **Olaf Ronneberger**, **Philipp Fischer**, and **Thomas Brox** in 2015 and was published at the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI) [8].
The authors recognized the challenges of accurate segmentation of biomedical images and proposed a novel approach that combined traditional computer vision techniques with deep learning. The proposed approach was designed specifically for biomedical image segmentation tasks and is now known as the U-Net architecture.
The U-Net architecture consists of a contracting path that captures the contextual information of the image and a symmetric expanding path that allows for precise localization of the object of interest. The network utilizes both convolutional and deconvolutional layers for feature extraction and up-sampling, respectively [8].
One of the key advances in the article is also the incorporation of data augmentation. Data augmentation is a technique that artificially increases the size of the training dataset by applying various transformations to the original images, such as rotation, scaling, and flipping. This allows the network to learn robust features that are invariant to such transformations, improving the generalization performance of the model.
The U-Net architecture has been successfully applied to several biomedical imaging tasks, including segmentation of retinal blood vessels, liver and kidney tumors, and nuclei in microscopy images. The results obtained by U-Net were found to be superior to previous state-of-the-art methods in terms of segmentation accuracy, speed, and generalization performance [8].
The success of U-Net in biomedical image segmentation has had a significant impact on the field of medical image analysis. The U-Net architecture has become one of the most widely used and cited deep learning models for biomedical image segmentation [4] and has inspired the development of several other architectures that build on its principles. The U-Net architecture has also played a crucial role in the development of computer-aided diagnosis (CAD) systems that can assist radiologists in interpreting images, potentially leading to faster and more accurate diagnoses.

In conclusion, the U-Net architecture has shown great promise in the field of biomedical image segmentation. The incorporation of traditional computer vision techniques with deep learning has led to significant improvements in segmentation accuracy, speed, and generalization performance. The U-Net architecture has become a benchmark in the field of medical image analysis and has inspired the development of several other architectures that have advanced the field even further.

Fully Convolutional Networks for Semantic Segmentation

Published in 2014 by **Jonathan Long**, **Evan Shelhamer**, and **Trevor Darrell**, the research paper proposes a convolutional neural network architecture that can perform semantic segmentation of images in an accurate and efficient way. Previous image classification networks were unsuitable for dense prediction tasks such as semantic segmentation because they used fully connected layers to output a single class score for an entire image.
The authors proposed an architecture that used only convolutional and pooling layers to output a dense per-pixel prediction map for the input image, which they called FCN-8s.
The network architecture consisted of a contracting path and an expanding path and incorporated skip connections to preserve fine-grained spatial information from earlier layers.
They applied a pixelwise softmax layer to produce a probability map for each class at every pixel location. FCN-8s outperformed previous state-of-the-art segmentation methods on several benchmark datasets and was also effective for tasks such as image deblurring and colorization [6].

Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation

Published in 2018 by **Liang-Chieh Chen**, **Yukun Zhu**, **George Papandreou**, and **Florian Schroff**, this research paper proposes DeepLabV3, a semantic segmentation model that achieves state-of-the-art results on several benchmark datasets.
The proposed model is based on an encoder-decoder architecture with atrous separable convolution and uses an "atrous spatial pyramid pooling module" to capture multi-scale contextual information from the input image.
The proposed model uses an atrous separable convolution layer as a building block, which reduces computational cost while maintaining high accuracy. Atrous convolution, also known as dilated convolution, is a technique that increases the receptive field of a convolutional layer without adding more parameters. Separable convolution, on the other hand, decomposes a convolutional layer into depthwise and pointwise convolutions, reducing computational cost. By combining these two techniques, the proposed atrous separable convolution layer can efficiently capture contextual information from the input image.
In addition to atrous separable convolution, the proposed model also uses an atrous spatial pyramid pooling (ASPP) module, which captures multi-scale contextual information from the input image at multiple scales. The ASPP module uses atrous convolution with different dilation rates to capture contextual information at different scales. By incorporating the ASPP module into the model, the proposed approach can effectively capture both local and global contextual information from the input image, leading to improved segmentation performance.
The model outperforms previous state-of-the-art segmentation methods on several benchmark datasets, including PASCAL VOC and Cityscapes, and can be applied to real-time video segmentation tasks with high accuracy and efficiency [2].

Solution Description

We have a dataset consisting of lung radiography images along with their corresponding segmentation masks. Our objective is to train a U-Net model, an FCN model, and a DeepLapV3 model on this dataset to perform segmentation on unseen radiography data. The segmentation will help us to identify the regions of interest in the radiography images and aid in the diagnosis of lung diseases like COVID-19. Our solution involves the following steps :

- **Data Preprocessing**: The first step is to preprocess the data to reduce noise by applying a Gaussian blur and a morphological filter, to adjust contrasts with CLAHE filter (Contrast Limited Adaptive Histogram Equalization), and then normalize the data to ensure consistency.
- **Model Training**: After preprocessing the

data, we need to implement and train all three models : U-Net, FCNN, and DeepLabV3.

- **Postprocessing**: After using our models to predict segmentation masks on the test data, we will then apply a postprocessing involving morphological filters for noise reduction, and thresholding.
- **Evaluation and comparison**: We will then evaluate our models using several metrics including IoU (intersection over union), Dice coefficient, MSE (Mean Squared Error).

Here's a solution outline summarizing the previous list of steps:

Algorithm 1 Solution outline

Input: COVID-19 images data
Step 1: Data Preparation:
- pre-processing the data (filters, normalization, ...).
- Split the data into training, validation and test sets.
- Resize the data for computation efficiency and performance.
Step 2: Model Training:
- Initialize the model architecture.
- Train the model on the training data.
- Evaluate the model on the validation data.
Step 3: Segmentation:
- Apply the trained model to segment the images from test set.
- Post-process the predicted masks and apply a threshold to remove false positives and negatives.
- Use various metrics and compare their results to the related work's models.
Output: Predictions for masks of unseen data from COVID-19.

Data Description

A database of chest X-ray images for COVID-19 positive cases along with Normal and Viral Pneumonia images compiled by the researchers of "Qatar University" and consists of 33,920 chest X-ray (CXR) images [3, 7], will be used for this project and it includes:

- 3616 images of COVID-19.
- 6012 images of Lung opacity.
- ∼10k2 images of Normal.
- 1345 images of Viral pneumonia.
- Ground-truth lung segmentation masks for each of the above.
- **Train dataset**: This is the data used to train our machine learning model and it correspond to a collection of 3616 images of size 299x299 pixels (features), which will be split into training and validation and test section. For our case we proceeded to use our experiments on COVID-19 folder only.

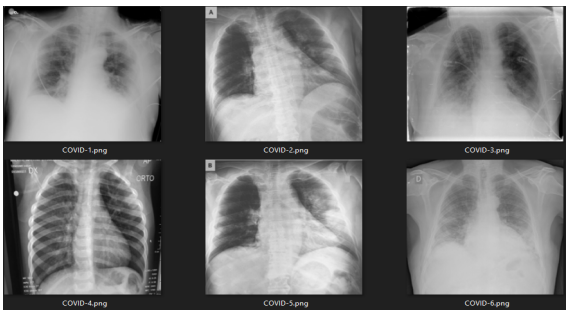

Figure 1: Train dataset sample

- **Validation dataset**: This is a subset of the training data that is used to evaluate the performance of the model during training. It helps to prevent overfitting. For the project only 20% of the training data had been used as validation.
- **Label dataset**: The label dataset values represent the masks which are also images that correspond to each image present in the training dataset.

Figure 2: Corresponding label dataset sample

- **Test dataset**: This dataset represents 10% of the complete dataset, and it will be used as a separate set of images to evaluate the model after training. It is important to note that the test dataset will always be unseen data for the model.

The following table will show some more informations about the datasets.

Dataset	#Images	Size (MByte)
Training	3616	123
Validation (20%)	651	24.6
Label	3616	5.1
Test (10%)	362	12.3

Table 1: Dataset informations

Hardware

Training machine learning models is a very complex process, it is heavy both on memory, and computation, we didn't have the adequate material at hand, thus we chose to use cloud platforms to do our experiments, we used cloud platforms for both the computation and the storage, here are the platforms used :

- **Google Colab**: We used Google Colab, which is Google's free AI platform, it consists of a Python environment with access to a GPU for training machine learning models, the exact specifications of the GPUs accessed by Colab is not available for free usage.
- **Azure Cloud**: We used Microsoft Azure's Data Storage resource, which permitted us to upload the dataset on the cloud, we used the Data Share resource to grant distant access to the Data Storage to multiple users, we then queried the data within our Python code to retrieve it. Azure uses blobs to store data, a data blob is a type of object that can be used to store unstructured data, such as images, documents, videos, and other types of files. A blob is essentially a collection of binary data, along with some metadata that describes the blob and its properties, such as its size, content type, and last modification time.

Process Flow

We start by establishing a connection to our Azure cloud data storage, which we will explain in pseudocode.
After loading the images from the Azure database, we rearranged them. Once this step was complete, we started reading the images from Azure using their paths, which were arranged in the previous step. For each image, a corresponding mask representing the label for our models was also loaded.
The images were read in grayscale and a Gaussian blur filter was applied to reduce the noises. We then applied a morphological gradient filter twice to highlight the edges of the objects present in the images. As a final step in the preprocessing, we applied CLAHE histogram equalization to adjust the contrast of the images.

Algorithm 2 Establishing a connection to blob storage account and accessing a container

Input : connection strings, container names.
- $connect\_str \leftarrow connect\_str$;
- Create a BlobServiceClient object $blob_service_client$ using $from\_connection\_string(connect\_str)$ method ;
- $container\_name \leftarrow container\_name$;
- Create a container client object $container\_client$ using $get\_container\_client(container\_name)$ method of the $BlobServiceClient$ class ;
- Return $container\_client$;

Output : a container client object for accessing the data storage.

All three models including : U-Net, FCNN, and DeepLabV3 were trained under various changes in their hyperparameters to achieve the best results. The models will be evaluated using metrics such as IoU and Dice coefficient. We saved only the best weights of the model that gave us the lowest loss value during training.
After finding the best models for each architecture, we used them to predict the segmentation of the test dataset. However, the output was not a binary mask, but rather a probability map. To convert this into a binary mask, we had to apply a threshold to determine which pixels represented an object and which did not. Two methods were used to calculate the threshold value: a global method and a local method. We noticed that while both methods yielded good results, they were slightly different from each other. To overcome this, we decided to combine the results using a union operation rather than a dilation operation, as the latter could potentially distort the object present in the predicted mask and fill in unwanted blanks.

Models Implementation

All models were implemented using the Tensorflow Keras library, in this section we will give more details about the implementation for each model.

U-Net

Algorithm 3 U-net Implementation

Input: input_image, n_filters, dropout_rate, batch_normalization (true or false)
Step 1: Contracting path:
- The path consists of 4 blocks.
- Each block has 2 convolution layers, 1 max pooling layer and a dropout.
- All layers have ReLU activation.

Step 2: Expanding path:
- The path mirrors the contracting path and also contains 4 blocks.
- Each block starts with a transpose convolution (deconvolution) layer this layer is then concatenated with the corresponding layer in the contracting path then there's a dropout and 2 more convolution layers.
- All layers have ReLU activation. - The output layer is a convolution layer with sigmoid activation

Step 3: Compiling:
- We compile the model using BCE loss, and Adam optimizer.
- We experimented a variety of epoch sizes.
- We save the best model weights and history in a file.

Output: File containing model weights and history.

DeepLab

Algorithm 4 DeepLab Implementation

Input: input_image, n_filters, dropout_rate, batch_normalization (true or false)
Step 1: Feature extraction part:
- This part consists of 3 blocks.
- The first 2 blocks consist of 2 convolution layers followed by a pooling layer.
- The 3rd block consists of 3 convolution layers, and a pooling layer.
- All layers use ReLU activation, the number of filters is ascending by a multiplier of 2 for each block.

Step 2: ASPP part:
- This part consists of 5 blocks.
- The first 4 blocks consist of 2 convolution layers, with each layer we augment the dilation rate.
- The last block concatenates all previous layers in this part of the network.
- All layers use ReLU activation, the number of filters is the same for every convolution layer.

Step 3: Decoder part:
- This part consists of 4 blocks.
- The first 3 blocks consist of one deconvolution layer for each block, with each layer we divide the number of filters by 2.
- The last block is the output layer.
- All layers use ReLU activation, except the output layer with uses sigmoid.

Step 3: Compiling:
- We compile the model using BCE loss, and Adam optimizer.
- We experimented a variety of epoch sizes.
- We save the best model weights and history in a file.

Output: File containing model weights and history.

FCNN

Algorithm 5 FCNN Implementation

Input: input_image, n_filters, dropout_rate, batch_normalization (true or false)
Step 1: Model Architecture:
- The path consists of 5 blocks.
- The first 2 blocks consist of 2 convolution layers followed by a pooling layer.
- The next 2 blocks consist of 2 convolution layers followed by a deconvolution layer.
- The last block consists of 3 convolution layers, the last one being the output layer.
- All layers use ReLU activation except for the last layer that uses sigmoid.

Step 3: Compiling:
- We compile the model using BCE loss, and Adam optimizer.
- We experimented a variety of epoch sizes.
- We save the best model weights and history in a file.

Output: File containing model weights and history.

Experimentation

Experimentations were conducted with all three models U-Net, FCNN, and DeepLabV3 during 50 epochs with a batchsize of 128, in the following section we will evaluate each one then compare them against each other.

U-Net

We monitored the evolution of loss and accuracy during training :

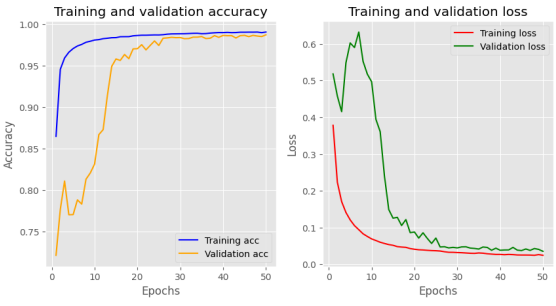

Figure 3: Loss and Accuracy curves for U-Net

We can observe that the validation and training loss curves both converged, indicating that we have no overfitting.
Both the validation and training accuracy curves also converged.
Here's a sample prediction done by U-Net on the Testing set, and compared with the ground truth mask :

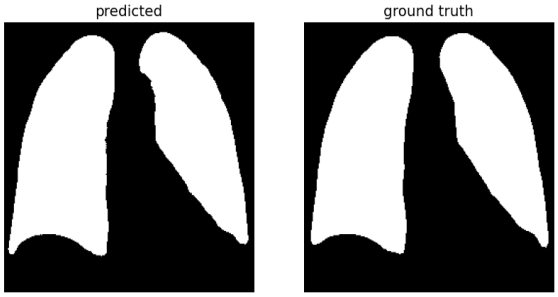

Figure 4: Sample prediction with U-Net

DeepLab

Here is the evolution of loss and accuracy during training :

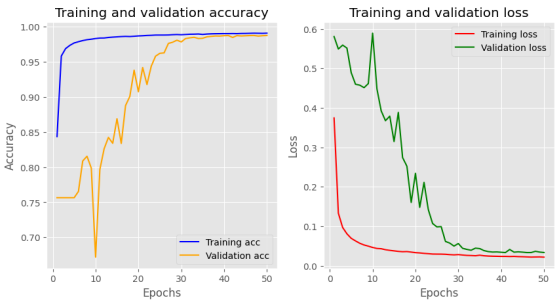

Figure 5: Loss and Accuracy curves for DeepLab

Despite some divergence in the beginning of training, both curves ultimately converged nicely.
Here is a sample prediction :

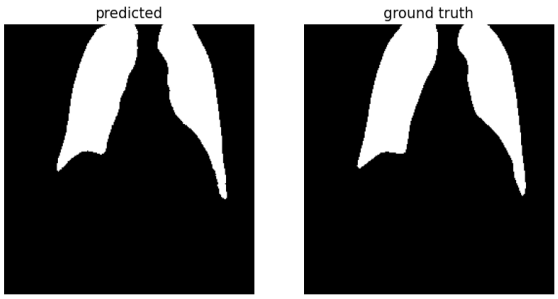

Figure 6: Sample Prediction with DeepLabV3

FCNN

Here is the evolution of loss and accuracy during training :

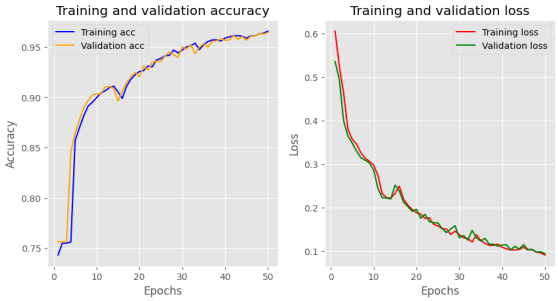

Figure 7: Loss and Accuracy curves for FCNN

We can observe that loss and accuracy curves all converged nicely.
Here is a sample prediction :

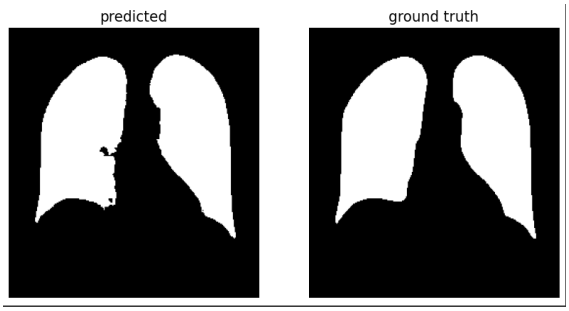

Figure 8: Sample Prediction with FCNN

Quantitative Evaluation

In the following, we will evaluate all our models using 3 metrics : Mean IoU (Intersection over Union), Mean Dice, and MSE (mean squared error).
Mean Intersection over Union (IoU) and Mean Dice are both evaluation metrics commonly used in image segmentation tasks. Mean IoU calculates the ratio of intersection over union for all classes and averages them. A high value indicates that the predicted segmentation is in good agreement with the ground truth. Mean Dice is also a ratio of overlap, but unlike IoU, it uses the sum of twice the intersection over the sum of the number of pixels in both the prediction and ground truth. This metric also ranges from 0 to 1, with a higher value indicating better segmentation performance. In both metrics, the higher the value, the better the segmentation.
The math formulas for Mean IoU and Mean Dice are as follows:

$$IoU = \frac{TP}{TP + FP + FN} \tag{1}$$

$$MeanIoU = \frac{1}{N}\sum_{i=1}^{N} IoU_i \tag{2}$$

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{3}$$

$$MeanDice = \frac{1}{N}\sum_{i=1}^{N} Dice_i \tag{4}$$

MSE (Mean Squared Error) is another commonly used metric in image segmentation tasks. Unlike Mean IoU and Mean Dice, MSE calculates the average of the squared differences between the predicted and ground truth segmentation for each pixel. A lower value of MSE indicates better segmentation performance. The math formula for MSE is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5}$$

where y is the ground truth segmentation and $\hat{y}$ is the predicted segmentation.

Model	mDice	mIOU	MSE
U-Net	0,97	0,94	0,02
FCNN	0,92	0,86	0,04
DeepLab	0,96	0,95	0,01

Table 2: Quantitative evaluation

What we can observe is that all three models perform well, and perform similarly, so we can't conclude much about which model is best for our dataset.

Qualitative Evaluation

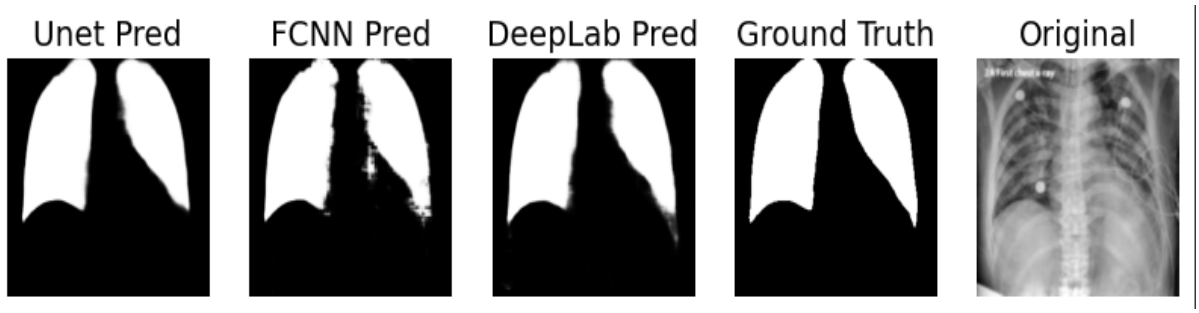

Figure 9: Test Prediction 1

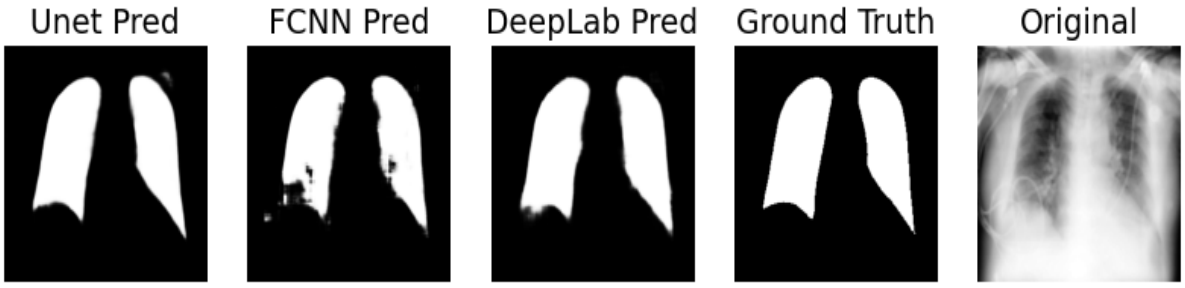

Figure 10: Test Prediction 2

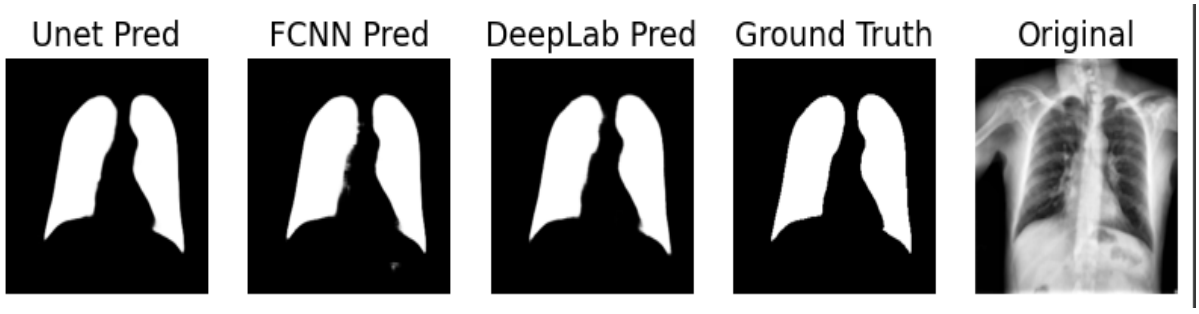

Figure 11: Test Prediction 3

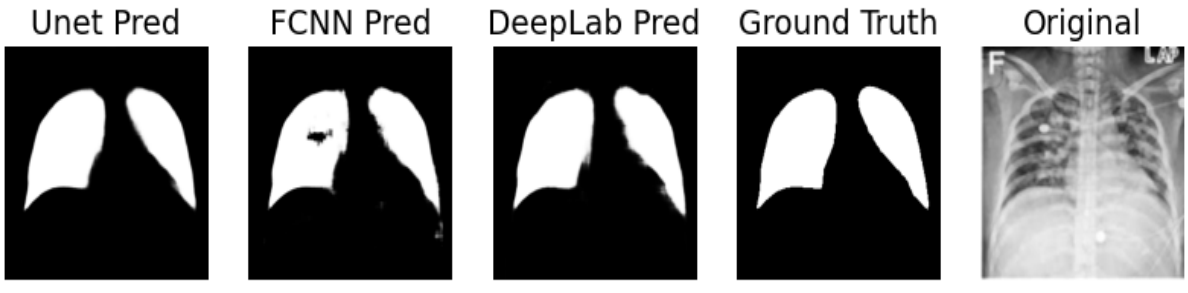

Figure 12: Test Prediction 4

Conclusion

In conclusion, our project focused on implementing three different algorithms for biomedical image segmentation, specifically using lung radiography images. We used the U-Net, FCNN, and DeepLabV3 models, each of which has been previously used and tested in the field of biomedical image analysis. Through our implementation and testing, we found that all three models were able to accurately segment the lung areas from the radiography images.

One of the key advantages of using these deep learning models for image segmentation is their ability to automatically learn features from the data without the need for hand-crafted features. This reduces the workload for researchers and allows for more accurate and efficient analysis. Additionally, the use of cloud storage solutions, such as Azure Cloud, allowed for easy data management and scalability.

However, we also faced some challenges during the implementation process. One of the main challenges was the need for large amounts of computational power and resources, which can be a limiting factor for some researchers or institutions. Additionally, preprocessing of the images, such as normalization and resizing, was necessary for optimal model performance.

Overall, our project demonstrates the potential for using deep learning models for biomedical image analysis and highlights the importance of data storage solutions for efficient and scalable research. With continued advancements in technology and increasing availability of cloud computing resources, we believe that these models have the potential to greatly impact the field of biomedical image analysis and contribute to improved medical diagnosis and treatment.

Potential improvements

While the three algorithms implemented in this project showed promising results in segmenting lung radiography images, there are still potential improvements that could be made to further enhance their performance. One area of improvement could be in the data augmentation techniques used, such as applying additional transformations to the images or using more advanced techniques like CycleGAN. Another potential improvement is in the architecture of the models themselves. For example, exploring deeper or wider networks, or experimenting with different activation functions or loss functions could potentially yield better results. Another area of potential improvement is in the data storage, we could dive more into Azure Cloud to make our data storage solution more secure and more scalable, or we could explore other cloud storage options. Additionally, incorporating additional types of data, such as patient metadata or additional imaging modalities, could potentially improve the accuracy and generalizability of the models. Finally, optimizing hyperparameters such as learning rate, batch size, and optimizer could also potentially improve the performance of the models. Overall, there are numerous avenues for future research and improvement in the field of biomedical image segmentation, and continued exploration of these avenues could lead to even more accurate and effective segmentation models.

Bibliography

[1] Andrew L Beam and Isaac S Kohane. Big data and machine learning in health care. *Jama*, 319(13):1317–1318, 2018.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[3] ME Chowdhury, T Rahman, A Khandakar, R Mazhar, MA Kadir, ZB Mahbub, KR Islam, MS Khan, A Iqbal, N Al-Emadi, et al. Can ai help in screening viral and covid-19 pneumonia? arxiv 2020. *arXiv preprint arXiv:2003.13145*, 2020.

[4] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[7] Tawsifur Rahman, Amith Khandakar, Yazan Qiblawey, Anas Tahir, Serkan Kiranyaz, Saad Bin Abul Kashem, Mohammad Tariqul Islam, Somaya Al Maadeed, Susu M Zughaier, Muhammad Salman Khan, et al. Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images. *Computers in biology and medicine*, 132:104319, 2021.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas 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.

[9] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*, pages 240–248. Springer, 2017.

[10] Hongtao Xie, Dongbao Yang, Nannan Sun, Zhineng Chen, and Yongdong Zhang. Automated pulmonary nodule detection in ct images using deep convolutional neural networks. *Pattern Recognition*, 85:109–119, 2019.