NAU NORTHERN ARIZONA UNIVERSITY
School of Informatics, Computing, and Cyber Systems

Project Telehealth

Ian Philippi & Tianyi Chen

4/11/24

TABLE OF CONTENTS

1. Introduction

1.1 Efficiency In the Medical Field

The current medical system depends heavily on X-ray scan technology to accurately depict certain body portions. The reason behind this is to provide skilled medical professionals with the information needed to accurately diagnose any issues, allowing doctors to create a plan to combat any ailment or damage to the human body without the need for invasive surgery. The technology behind medical scan equipment is advanced, but there is still an area in the diagnosis process that can use improvement. Currently, diagnoses take 1-3 days. Although this may not seem like an issue, there are plenty of ailments that can become lethal within that time frame. One method that can be used to improve diagnosis times is the implementation of AI in the medical field.

Surprisingly, many other teams have set out to formulate a solution to the same issue in diagnosis. However, what sets our team apart is that we have not only formulated code solutions to the issue, but we have created multiple different forms of our solution using different image analysis programs such as ViT (Visual Transform) and EfficientNet to find the best option based on updated software. Furthermore, once securing a more accurate program for diagnosis, our team moved to the 2nd stage of the project, which utilizes Kyber quantum-resistant lattice-based encryption to protect sensitive medical data. The motivation behind this extensive testing is due to the rapid pace at which programs improve. Therefore, the programs used in our research have

created updated models that may show different and more promising results than the research papers that our team used in our research phase.

1.2 Client of Team Tech-Health Project

Professor Tuy Nguyen

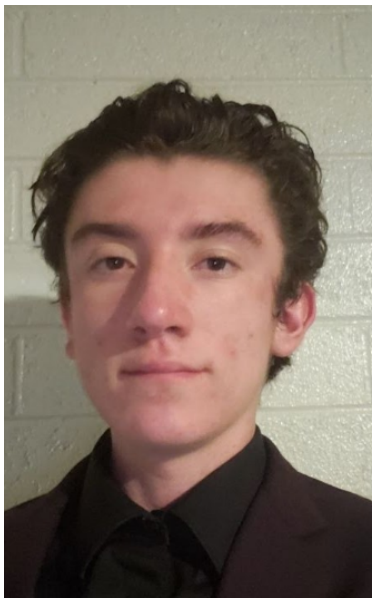
Professor Tuy Nguyen is currently an Assistant Professor at the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. Previously, he held the positions of Lecturer at the School of Global Convergence Studies and Post-Doctoral Fellow at the Department of Electrical and Computer Engineering at Inha University, from May 2021 to August 2022. Prior to that, he worked as a Senior Research Engineer at Conextt Inc., contributing from September 2019 to April 2021. He earned his Ph.D. in Information and Communication Engineering from Inha University in August 2019.

1.3 Team Members of Team Tech-Health

Leader: Tianyi (Bruce) Chen

This is Tianyi (Bruce) Chen, a computer engineering student at Northern Arizona University. His primary academic focus is on machine learning (image processing), a field that deeply fascinates him. In his leisure time, he engages in small-scale IoT projects, where he finds great satisfaction in blending hardware and software to create tangible, functional devices. His programming experience casts through a vast array of Python, R, C, Javascript, Verilog, etc.

His role involves rigorously training and testing various models with the TeleHealth dataset, meticulously analyzing the outcomes, composing the paper, incorporating encryption algorithms with the AI model into the system, and making interactive visualization software.



Secretary and Treasurer: Ian Philippi

This is Ian Philippi, a Computer Engineer at Northern Arizona University. He enjoys working hard and playing video games during his free time (although there isn't much free time during school terms). He is proficient in and enjoys formulating code in C, Python, Assembly, and SystemVerilog.

He finds project AI TeleHealth to be very important because it can greatly benefit those in need of an efficient/accurate diagnosis, which allows medical professionals more time to perform life-saving procedures.

1.4 User Manual

(our project is a research project, and this manual is for the extra content developed in our project)

Our designated system integrates two core components: AI-assisted diagnosis and homomorphic encryption for secure transmission. The process is delineated as follows:

1. **Public Key Generation Request:** The server initiates the process by generating a pair of cryptographic keys. While the public key is transmitted to the client, the secret key is securely retained by the server.

2. **Local Data Encryption:** Upon receiving the public key, the client encrypts the diagnostic image using this key, ensuring that the data remains secure during transmission.

3. **Transmission of Encrypted Data:** The encrypted data, now termed 'ciphertext', is sent back to the server. The server, holding the secret key, decrypts the ciphertext to retrieve the original image.

4. **AI-Powered Diagnosis:** The decrypted image is then analyzed using our proprietary AI model. This model, developed through rigorous research, performs the diagnostic tasks.

5. **Transmission of Diagnostic Results:** Upon completion of the diagnosis, the server encrypts the prediction results using a new public key generated by the client. This step ensures that the transmitted results remain secure until they reach the client.

6. **Result Reception and Decryption by the Client:** The client decrypts the received encrypted results using the corresponding private key. This decrypted data is then forwarded to healthcare professionals for validation and further action.

2. Problem Statement

2.1 Needs of AI/Machine Learning in Diagnosing

Modern diagnosis methods have developed significantly in recent years, but there are only a few hospitals fully utilizing machine learning. According to [1], only 24 percent of interviewed hospitals said they were experimenting with AI/machine learning techniques and 22 percent of these hospitals said they were in the initializing stage of incorporating AI. Despite few hospitals adopting healing methods with AI/machine learning, lots of top hospitals around the world (like Mayo Clinic, Cleveland Clinic, and Massachusetts General Hospital) have invested millions into AI/machine learning construction according to [2], which indicates that AI/machine learning has a broad market with diagnostics. [3] also indicates the huge potential of AI/machine learning because it has extraordinary diagnosing abilities in some aspects in the Future Healthcare Journal (2019) [4]. Additionally, [3] points out that one of the main impedances to the implementation of AI/machine learning in the medical field is the instability of most existing AI/machine learning models. However, this statement is quickly becoming invalid due to the rapid improvement of AI neural networks.

Moreover, after the explosion of the COVID-19 virus, many countries and regions are facing a lack of medical diagnosis resources. People further realize the importance of combining AI/machine learning techniques with tele-medical diagnosis. In conclusion, the market has broad prospects for high performance, accuracy, and stable machine learning methods (models) especially in COVID-19 diagnosing fields. In our project, we utilized a chest X-ray image dataset consisting of scans relating to healthy lungs as well as lungs inflicted with COVID-19,

viral pneumonia, and lung opacity, implementing existing machine learning models to classify different types of images, and compare the performance between each model to find the one that best matches our client's expectations.

2.2 Needs for Encryption in Telehealth

With the possibility of telehealth in the medical field, people notice a possible privacy problem when transferring clients' information. The current method for protecting the transferred data is end-to-end encryption which is safe enough but hard to backup and archive. However, with the development of quantum computing technologies, sensitive data in the form of medical scans used in our software can be intercepted and stolen.

Given the significant risk of security in the first phase of our project, the second phase includes using Kyber, which is a lattice-based encryption method that utilizes multi-dimensional lattices to encrypt and decrypt data given vector keys [4]. In this process, an image array is located at a specific point in a multi-dimensional lattice. In the past, RSA encryption was the most popular method, in which semi-prime numbers (products of 2 prime numbers) were used as the public keys. Given that, the private keys are the prime factors of the semiprime public key. This is highly effective on regular and supercomputers because of the time it takes for them to run the calculations necessary to break this encryption after intercepting data. However, given that quantum bits can be in multiple states at once, RSA encryption becomes ineffective in quantum computing. However, given a Kyber encryption system that is multidimensional, even utilizing bits that can be in a state of superposition still takes a significant amount of time to decrypt intercepted information.

2.3 Statement of Objectives

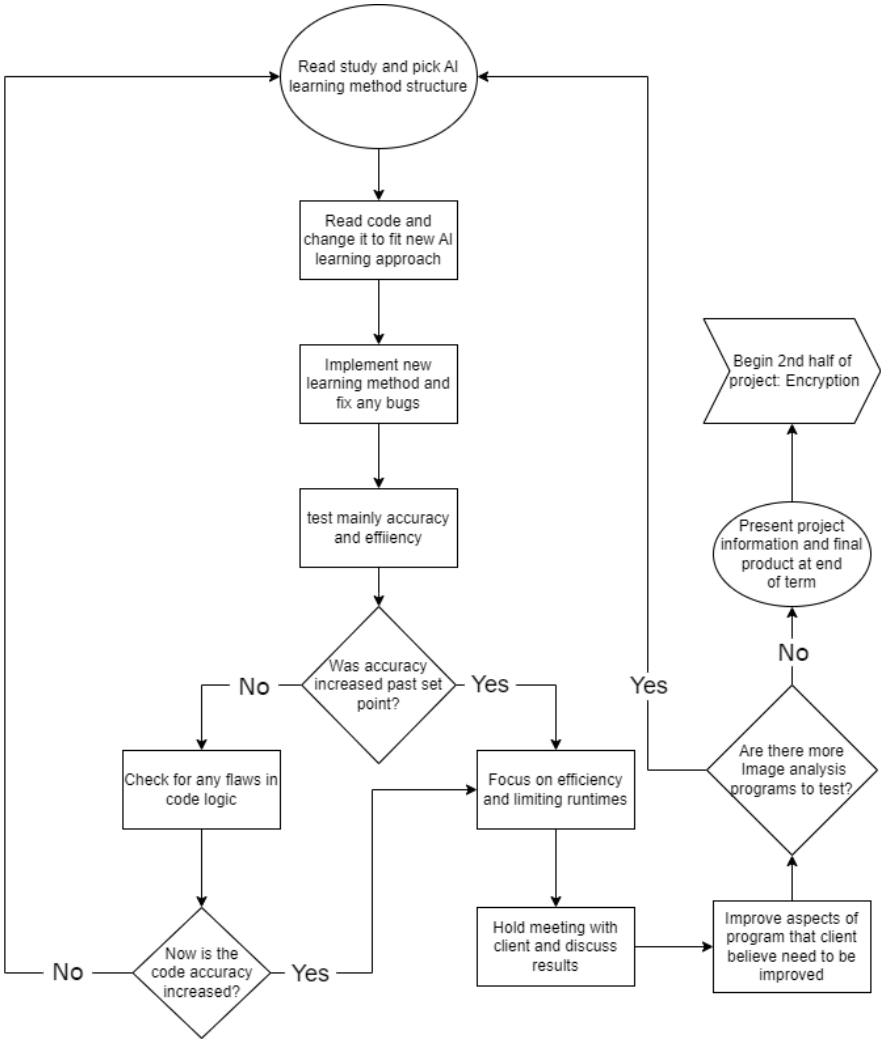


Figure 1: Project Phase 1 (Research: AI Diagnosis)

This is a flow chart of the first phase tasks of our Tech Health Comparators. Our research is conducting experiments through various model structures with a focus on their accuracy and efficiency. After assessing these two performances by several evaluation metrics, we tune the parameter configurations and start the next rounds of experiments. First, Vast experiments are

executed to explore the optimized settings for each model. Then we elaborate on the most suitable structure which is traded off from accuracy and efficiency to conduct more meticulous tuning. Finalizing our proposed model, we use the visualization technique to verify it and explore the potential bias in the experimented dataset.

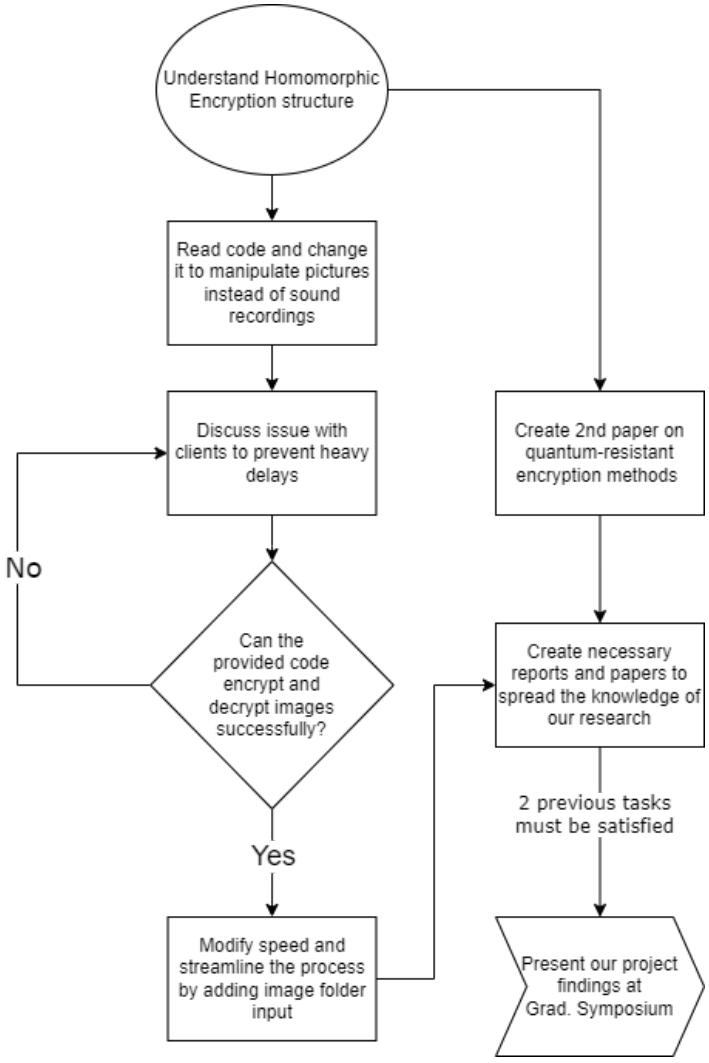


Figure 2: Project Phase 2 (Homomorphic Encryption)

This graph represents the finalized version of the second phase of our project, which involves encryption. As said previously, the method of encryption that our project is going to use is based on quantum-resistant encryption methods, In short, this means that the encryption algorithm uses a lattice, hiding the picture at a specific point in said lattice. In order to decrypt the data, a computer needs to navigate the lattice only using 2 vectors. However, the best two vectors for efficient lattice navigation are given to the receiver as keys. However, any other systems attempting to intercept and steal this data now need to navigate the same lattice using highly inefficient vectors. This concept is what makes this type of encryption effective against quantum computing. Therefore, given code that already implements this encryption, but only with audio files, with simple modification this can be applied to images. Specifically, this is done by converting the images into a one-dimensional array, encrypting the data, decrypting the data, and then reconstructing the array into a 2-dimensional array format. As this has already been accomplished, the last implementations are regarding easy mass input of medical scans using an array of images.

3. Concept

3.1 Accuracy and Efficiency of AI Learning

Chest X-rays are a crucial tool for diagnosing various illnesses. However, their current use faces limitations in efficiency, particularly during critical situations like pandemics. The delay of 1-2 days between scans and diagnosis can significantly impact patient outcomes. This challenge is particularly evident with diseases like COVID-19, which has affected over 772 million people globally, with over 7 million fatalities [1]. The COVID-19 pandemic resulted in a significant economic burden, with a 3.3 trillion dollar deficit in the US for 2020 and a peak unemployment rate of 14.7% [2, 3]. Medical institutions faced surging demand, leading to longer wait times and exacerbated disparities in access to care. Studies in [3, 4] found a concerning 117% increase in wait time disparities. To address these challenges, a critical re-evaluation of current diagnostic approaches is essential. Embracing innovative solutions and integrating technology are key to improving efficiency, minimizing delays, and optimizing healthcare outcomes.

Our study leverages machine learning to enhance the diagnostic accuracy of COVID-19 using chest X-rays. The study evaluates various architectures, including efficient neural networks (EfficientNet) [7], multiscale vision transformers (MViT) [6], efficient vision transformers (EfficientViT) [8], and vision transformers (ViT) [9], against a comprehensive open-source dataset comprising 3616 COVID-19, 6012 lung opacity, 10192 normal, and 1345 viral pneumonia images. The analysis, focusing on loss functions and evaluation metrics,

demonstrates distinct performance variations among these models. Notably, multiscale models like MViT and EfficientNet tend toward overfitting. Conversely, our vision transformer model, innovatively fine-tuned (FT) on the encoder blocks, exhibits superior accuracy: 95.79% in four-class, 99.57% in three-class, and similarly high performance in binary classifications, along with a recall of 98.58%, precision of 98.87%, F1 score of 98.73%, specificity of 99.76%, and area under the receiver operating characteristic (ROC) curve (AUC) of 0.9993. The study confirms the vision transformer model's efficacy through rigorous validation using qu*

+antitative metrics and visualization techniques and illustrates its superiority over conventional models. The innovative fine-tuning method applied to vision transformers presents a significant advancement in medical image analysis, offering a promising avenue for improving the accuracy and reliability of COVID-19 diagnosis from chest X-ray images.

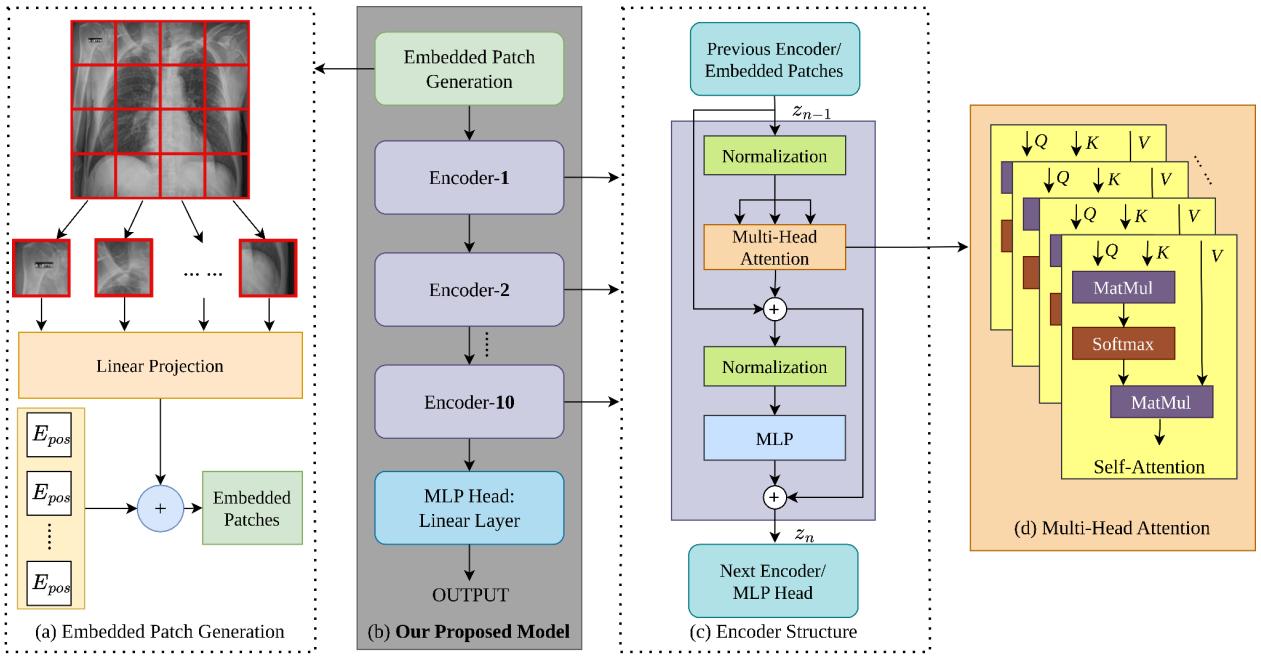


Figure 3: Overview of Vision Transformer and Our Proposed Model

3.2 Versatility and Strength of Encryption

```
image_number = 1
root_path = '/content/drive/MyDrive/Kyber_Input_Photos/'

for file in os.listdir(root_path):

    # Step 1: Load your image data
    image_path = root_path + 'Normal-' + str(image_number) + '.png'
    print(image_path)
    image = Image.open(image_path)

    # Step 2: Resize your image data
    rescale_size = (299, 299)
    rescale_image = image.resize(rescale_size, Image.BICUBIC)
    rescale_image.save('rescale_image.png')

    # Step 3: Create a histogram representing each image pixel's value
    norm_img = plt.imread('rescale_image.png')
    plt.hist(norm_img.ravel(), bins=256, range=(0.0, 1.0), fc='k', ec='k')
    plt.show()

    # Step 4: Resize the image in byte format
    rescale_image_bytes = io.BytesIO()
    rescale_image.save(rescale_image_bytes, format='JPEG')
    image_bytes = rescale_image_bytes.getvalue()
    display(rescale_image)

    # Step 5: Generate a key pair
    public_key, secret_key = Kyber1024.keygen()

    # Step 6: Define the chunk size (adjust as needed)s
    chunk_size = 32

    # Step 7: Measure encryption time
    encryption_start_time = time.time()

    # Step 8: Encrypt and decrypt audio data in chunks
    ciphertexts = []
    cipher_number = []
    decrypted_chunks = []

    for i in range(0, len(image_bytes), chunk_size):
        # Extract a chunk of image data
        chunk = image_bytes[i:i+chunk_size]

        remainder = len(chunk) % 32
        if remainder != 0:
            # Pad the image data with zeros to make it a multiple of 32 bytes
            padding_length = 32 - remainder
            chunk += b'\x00' * padding_length

        # Encrypt the chunk with the public key
        ciphertext = Kyber1024._cpapke_enc(public_key, chunk, coins=os.urandom(32))
        number_string = bytes_to_number_string(ciphertext)
        ciphertexts.append(ciphertext)
        cipher_number.append(number_string)

    # Step 9: Measure encryption time
    encryption_end_time = time.time()

    # Step 10: Concatenate ciphertexts into a single string
    ciphertext_string = b''.join(ciphertexts)
```

```
# Step 11: Convert the concatenated ciphertext to base64-encoded string
encrypted_image_string = base64.b64encode(ciphertext_string).decode('utf-8')

# Step 12: Create a new image with the desired size
enc_img = Image.new('L', (299, 299)) #Use 'L' flag for gray image
# Encode the base64-encoded string as an image
enc_img.frombytes(base64.b64decode(encrypted_image_string), 'raw', 'L', 0, 1) #Use 'L' flag for gray image
# Save the encrypted image to a file
enc_img.save('encrypted_image.png')
display(enc_img)

enc_hist = plt.imread('encrypted_image.png')
plt.hist(enc_hist.ravel(), bins=256, range=(0.0, 1.0), fc='k', ec='k')
plt.show()

# Measure decryption time
decryption_start_time = time.time()

# Step 13: Decrypt the ciphertexts
decrypted_chunks = []
for ciphertext in ciphertexts:
    # Decrypt the ciphertext with the secret key
    decrypted_chunk_bytes = Kyber1024._cpapke_dec(secret_key, ciphertext)

    # Step 6: Remove any padding to obtain the decrypted audio data
    decrypted_chunk_bytes = decrypted_chunk_bytes[:len(chunk)]

    # Convert the decrypted bytes back to a NumPy array
    decrypted_chunk = np.frombuffer(decrypted_chunk_bytes, dtype=np.float32)

    # Append the decrypted chunk to the list of decrypted chunks
    decrypted_chunks.append(decrypted_chunk)

# Step 14: Combine the decrypted chunks into the final decrypted audio data
decrypted_img_array = np.concatenate(decrypted_chunks)
decrypted_img_array = decrypted_img_array[:len(image_bytes)]
decrypted_img_bytes = io.BytesIO()
decrypted_img_bytes.write(decrypted_img_array.tobytes())
decrypted_img_bytes.seek(0)
decrypted_img_bytes = decrypted_img_bytes.read()

# Step 15: Measure decryption time
decryption_end_time = time.time()

# Step 16: Calculate the elapsed times
encryption_elapsed_time = encryption_end_time - encryption_start_time
decryption_elapsed_time = decryption_end_time - decryption_start_time

print(f"Encryption time: {encryption_elapsed_time} seconds")
print(f"Decryption time: {decryption_elapsed_time} seconds")

# Step 17: Convert decrypted bytes to an image
decrypted_img = Image.open(io.BytesIO(decrypted_img_bytes))

# tep 18: Display the decrypted image
display(decrypted_img)

# Step 19: Rescale Encrypted Image
rescaled_final_image = decrypted_img.resize(rescale_size, Image.BICUBIC)
```

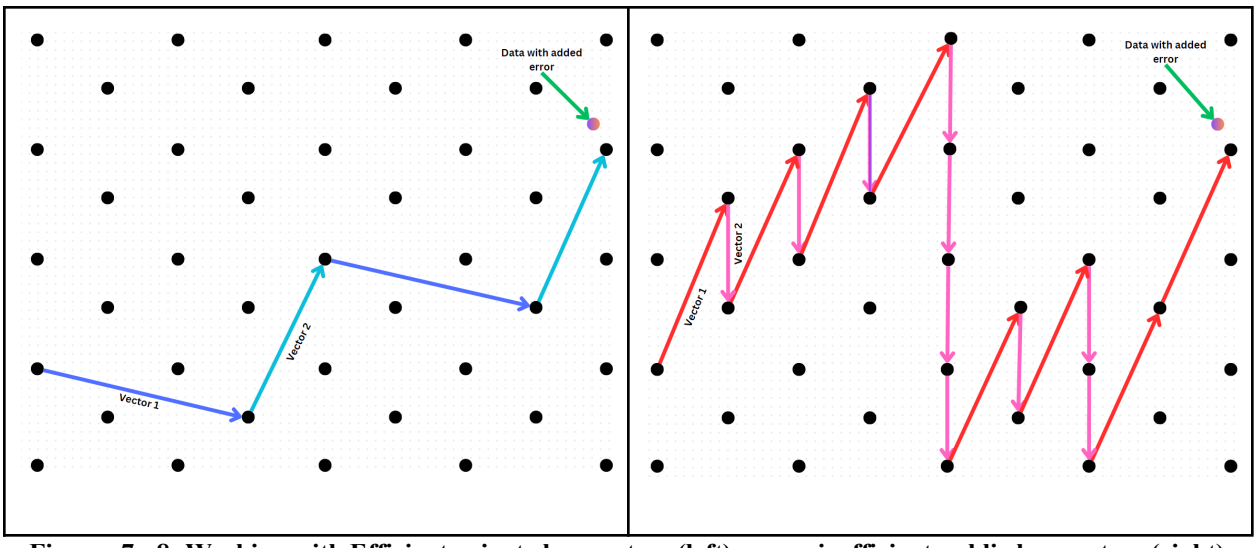
```
# Step 20: Rescale the final image
rescale_image.save('rescaled_final_image.png')
final_img = plt.imread('rescaled_final_image.png')
plt.hist(final_img.ravel(), bins=256, range=(0.0, 1.0), fc='k', ec='k')
plt.show()

# Step 21: Verify that the decrypted image data matches the original
if np.array_equal(image_bytes, decrypted_img_bytes):
    print("Decryption successful")
else:
    print("Decryption failed or image data mismatch.")

#Step 22: Increment image number to select the next picture
image_number = image_number + 1
```

Figure 4 - 6: Homomorphic Encryption Code Based on Kyber

The code that was used to apply homomorphic encryption to our project can be seen in Figures 4-6. The task given to us by our clients was to modify a previously formed solution to handle a new set of data. In this case, the original code was made to work with audio signals. Audio recordings are visually represented with peaks and valleys placed on a single axis. This visualization is quite accurate because audio files use a single-axis or 1D array format. For that reason, it is easy to do the same for an image. To accomplish this, the input image first needs to be read as a grayscale image that has a 2-axis format. From here, every row will be lined up back to back in a new 1D array. For example, for an image that is 214x214 pixels, you would have an array that follows the form of:

[row_1_column_1-214_data || row_2_column_1-214_data || … || row_214_column_1-214_data]

Once in this form, the intended needs to create the necessary encryption keys, which include a public and private key. Once generated, the recipient sends their public key to the sender. Using the public key, the data representing the medical scan is encrypted in chunks, with each encrypted chunk becoming known as a cipher text, which can be sent back to the intended recipient that has the private key associated with the encrypted data. This private key can easily decrypt the encrypted data. Once decrypted, a simple resize function can be used on the 1D array to fit within the parameters of the image bounds once again. Testing out the final project with this code has led to identical images at the start and end of the encryption/decryption cycle every time.



Figures 7 - 8: Working with Efficient private key vectors (left) versus inefficient public key vectors (right)

The previous paragraphs explain the process of our code. However, the topic of lattice-based encryption is still unclear. Understanding lattice-based encryption is made simpler by comparing it to the Ron Rivest, Adi Shamir, and Leonard Adleman (RSA) encryption method. This method was effective for a reasonable amount of time, utilizing large prime numbers more

than 300 digits long as a public key. The two private keys needed to decrypt a given message are two smaller prime numbers that are multiplied together to equal the public key's value. The computational strain and time needed to decrypt data from an RSA encryption for both regular and supercomputers increased its recognition. However, with the development of quantum computers that can run mathematical computations in multiple-bit states at once, this method became ineffective Therefore, lattice-based encryption used lattices and vectors to secure data. It can be easier to think of a lattice similar to that of a graph. In a lattice, there are highly efficient and inefficient sets of vectors (shown in figures. In this case, the public key is a lattice that can have n dimensions (we will visualize it in only 2 dimensions for now) that give an individual a poor set of vectors to navigate the lattice. For the individual with the private key, that key holds a highly efficient set of vectors, making the task of finding the data much simpler to find to find the data. However, Kyber also introduces a “learning with errors” system to this scheme. In short, this modification adds intensional errors to the equation that designates where the data is held, causing the data to slightly drift away from the given point. With this implementation, now there is the added difficulty of navigating to the point closest to the data to find it. This method is extremely effective for all computer systems and therefore, a desired encryption scheme for our project.

3.3 Combination of AI Learning and Quantum-Resistant Encryption

Artificial intelligence (AI) diagnosis-based telehealth systems have attracted lots of researcher's interest in recent years, as a potential technique to mitigate the burden on the medical system. Diagnosis accuracy, as one of the most significant aspects in this field, is putting a lot of effort into devising specific models for each kind of disease by researchers, where bias is reported as a significant factor that impedes model training [10]. Aside from the accuracy of AI auxiliary diagnosis, the privacy of patient data during transmission also raises researchers' concerns. Moreover, with the development of quantum computers, traditional encryption methods become vulnerable to threats from quantum cryptanalysis, as many experts worry [11].

In this research, we start with designing a COVID-19 AI auxiliary diagnosing system that is experimented on a chest X-ray dataset. In our designated system, we use quantum-resistant homomorphic encryption as the protection of transmission between the clients and the diagnosing server. To assist the analysis and explore bias within image cases, we make an interactive software to display all the positive and negative predictions, the model's confidence with its predictions, and how the model makes these predictions.

- **Accurate COVID-19-oriented AI diagnosing model:** Our system utilizes the accurate and efficient fine-tuned vision transformer model from our phase-1 research.
- **Quantum-resistant homomorphic encryption:** To enhance the confidentiality of patient data amidst potential interception by unauthorized entities, our system incorporates homomorphic encryption. This process initiates with the client requesting the generation of key pairs based on homomorphic encryption protocols from the server. Subsequently,

the client utilizes the acquired public key to encrypt the chest X-ray images, while the server retains the secret key, ensuring it remains undisclosed. Following this, the client transmits the encrypted images (ciphertexts) to the server, which then decrypts these ciphertexts utilizing the secret key.

Upon successful decryption, the server applies a specifically fine-tuned model to conduct diagnostic evaluations (predictions), which are complemented by visual representations of the model's predictive analytics (a detailed discussion is provided in the subsequent section). To complete the cycle, the server encrypts the diagnostic results using the public key initially provided by the client, mirroring the initial encryption process. These encrypted results are then dispatched to the client.

The final stage involves decrypting the received results, which are then forwarded to medical professionals for validation and to research institutions. This step facilitates a critical evaluation of the model's inherent biases, paving the way for necessary refinements to enhance its accuracy and reliability.

- **Interactive visualization analysis:** Discovering biases and implementing strategies to eliminate them is a crucial step in training machine learning with medical datasets. To have a more intuitive understanding of how experimented modes make predictions, we utilize the gradient-weighted class activation mapping (Grad-CAM) [12] method which calculates back-forwarded gradients from a specified class to draw a weighted heatmap. By overlapping the resized and projected heatmap with the origin image, researchers can

analyze patterns of models' predictions. To make this analysis more intuitive, comprehensive, and informative, we devise this Animated2GradCAM software, which integrates our GradCAM visualization, data loading, and the powerful interactive visualization Animint2 R package [13] together, to display all tested images with the prediction results comparing to actual their corresponding GradCAM heatmap (2-D black-white), colormap, and overlapped images by just dragging and clicking the index bar.

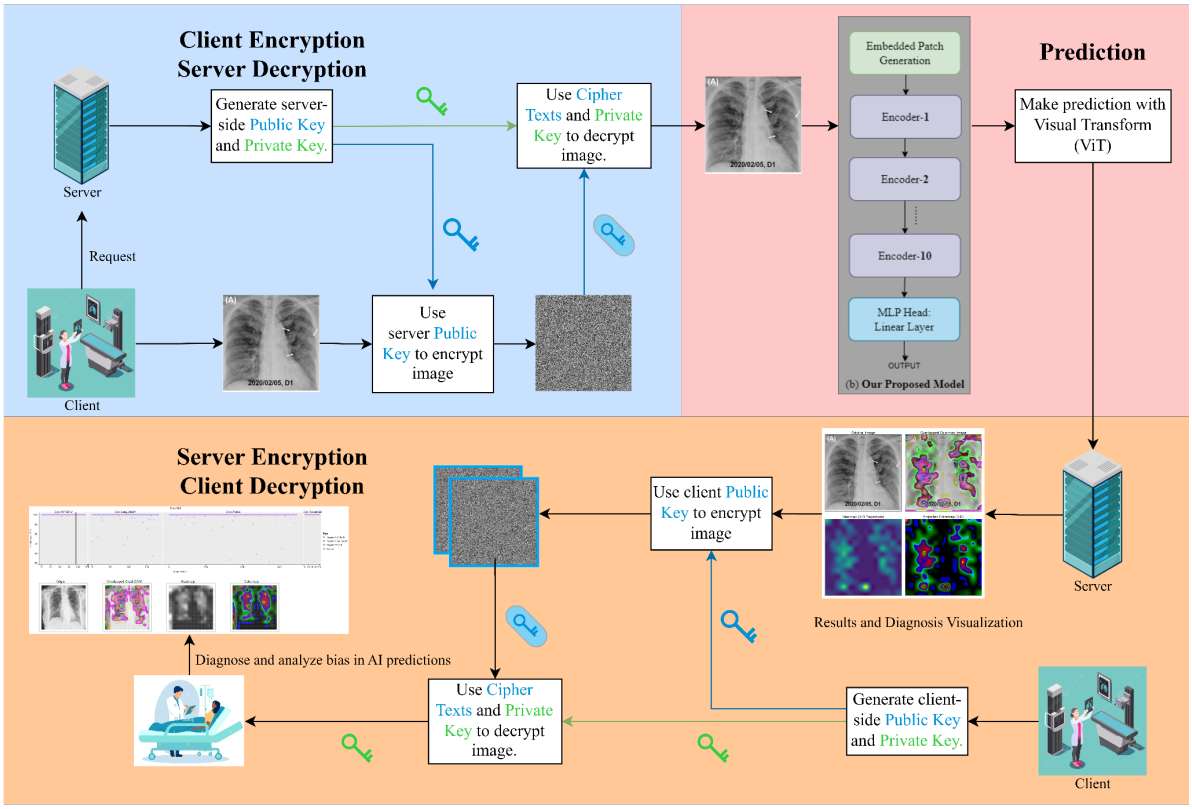


Figure 9: Our Proposed AI Telehealth System

4. Project Management

4.1 Gantt Chart

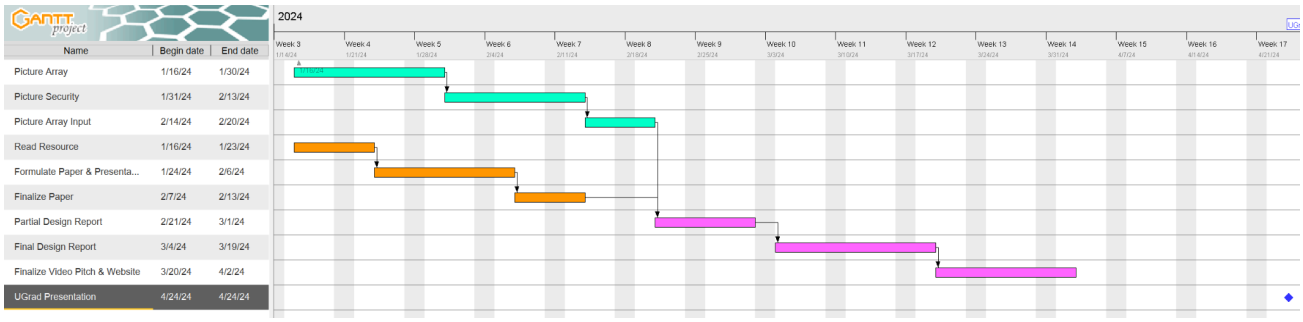


Figure 10: Gantt Chart

This is the Gantt Chart for the 2nd phase of our capstone project, alongside our PERT diagram, this chart does an excellent job of showing the split between our team for the first portion of the 2nd term. This divide and conquer strategy is useful to complete the 2nd project research paper and encryption code simultaneously. In fact, trusting this approach has placed our project near completion months before the end of the current term. Currently, any further modifications to the code will be to improve accessibility, which is greatly received but not necessary to complete the project.

4.2 WBS

Table 1: WBS

ID	Activity	Description	Deliverables	Duration (Days)	People	Resources	Dependencies
1	**Encryption Code**						
1.1	Picture Array	Convert code into array	Finished Code	2 weeks	Ian	Laptop / Google Collab	
1.2	Picture Security	Encrypt and Decrypt	Finished Code	2 weeks	Ian	Laptop / Google Collab	1.1
1.3	Picture Array Input	Allow for Mass Image Input	Finished Code	1 week	Ian	Laptop / Google Collab	1.2
2	**Research**						
2.1	Read Resource	Read research source		1 week	Tianyi	Research Sources	
2.2	Formulate Paper & Presentation	Create research paper	Rough Draft	1-2 weeks	Tianyi	Research Sources	2.1
2.3	Finalize Paper	Review and finalize Paper	Final Draft	2-7 days	Entire team	Paper Review Software	2.2
3	**Report & Presentation Material**						
3.1	Partial Design Report	Create Updated Report	Report	1 week	Ian & Tianyi	Google Docs	
3.2	Final Design Report	Finalize Partial Design Report	Finalized Report	2 weeks	Ian & Tianyi	Google Docs	3.1
3.3	Finalize Video Pitch & Website	Finalize Website and Video Pitch	Video & Finalized Website	1-2 weeks	Ian & Tianyi	Video Software, Google Sites, & Video Tech	3.2 & Website Draft
3.4	UGrad Presentation	Present Solution	Presentation, Website, & Final Project	1 day	Entire team		

This WBS chart is important for observing our project in a tabular format while also creating a medium that can assist our team with formulating and modifying the Gantt and PERT charts. Unlike the other charts, these charts allow for better visualization of how work is divided between team members. For example, it can be easily observed that Ian is working on data encryption (section 1) while Tianyi (Bruce) is working on the 2nd paper produced by this team (section 2), summarizing encryption. However, it is important to note that the publication of the 2nd paper is not guaranteed and still going through the process of being accepted. After both

Tianyi (Bruce) and Ian finish with their portions of the project, the team will reassemble to work on the capstone class deliverables (section 3).

4.3 PERT Chart

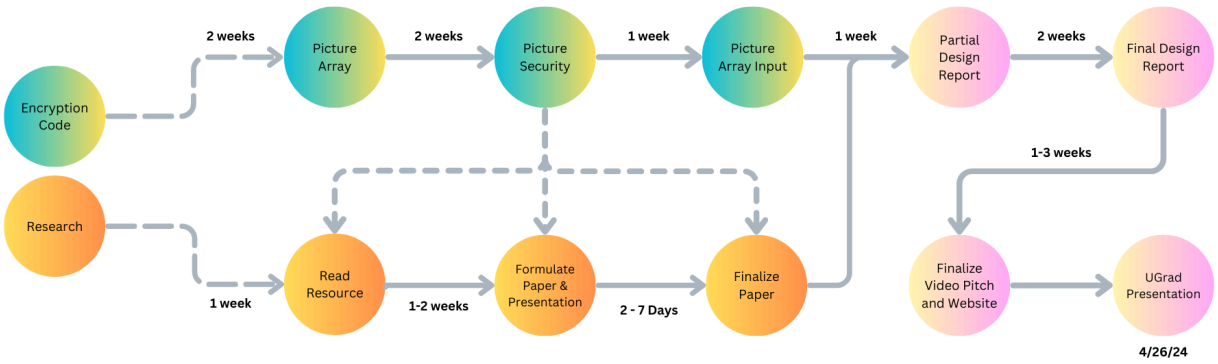


Figure 11: PERT Chart

Figure 7 displays a PERT Chart, which is a diagram that graphically shows the flow of the 2nd phase of our project. For this phase of our project, our team of 2 is split into 2 sections. Specifically, Tianyi (Bruce) is working on our team's 2nd paper, which can be seen as orange nodes. Therefore, Ian was tasked to work on the green node tasks, which relate to implementing encryption code. Once both of these sections are complete, which is where our team is currently, our team will work as one unit to complete the necessary documentation for our capstone course and symposium presentation (seen as pink nodes).

4.4 Bill of Materials

Table 2: Bills Of Materials

Vendor Name	Link to Item	Description	Item/Catalog #	Size/Color	Qty	Discount Code	Total Cost
Dell/HP/MSI/etc.	Depend on User Requirements	Laptop to create, modify, and run code	N/A	N/A	1	N/A	$700-1500
N/A	N/A	Labor Cost in Hours	N/A	N/A	12	N/A	~$720
Google	https://colab.research.google.com/signup	**Optional:** Monthly Google Colab Pro Subscription for Python Code	N/A	N/A	7	N/A	$80
						Total:	$1,420-2,300

Given that this project is specifically designed to be completed in an online environment, there are not many materials needed for this project. Additionally, given that this software is meant to be outsourced to medical institutions, the dataset that our code learns from is supplied by the user. Therefore, depending on the size of the team that would be either improving or recreating this project, they will need 1 laptop minimum and a Google Colab membership for multiple months if their team needs to outsource a powerful GPU for image processing. Additionally, labor costs at an average computer engineer's pay grade will lead to an additional ~$720 labor payment to be supplied. Therefore, as a startup, this project is estimated to cost $1,420-2,300 given a single individual is working on the project. Note that this value may fluctuate given a larger team.

5. Testing

5.1 Research: Fine-tune Models

1. Dataset and Preprocess

The COVID-19 chest X-ray dataset used in this study was sourced from the publicly available COVID-19 Radiography Database [13,14,15]. This dataset comprises four distinct classes: 3,616 COVID-19 positive cases, 10,192 normal cases, 6,012 lung opacity (non-COVID lung infection) cases, and 1,345 viral pneumonia images, with an initial resolution of 299 by 299, as illustrated in Fig. 10. In our experiments, the dataset is fully shuffled and split into 80% for training, 10% for validation, and 10% for testing. To address potential data imbalances and enhance training efficiency, we preprocess the images in several ways. First, we normalize the original pixel values from [0, 255] to [0.0, 1.0]. This normalization step facilitates the training process by ensuring all pixel values are on a similar scale. Second, we resize the images to 224 by 224 pixels to match the input requirements of the pre-trained models we employ for transfer learning. Inspired by the work in [14], which investigated various preprocessing techniques for chest X-ray images with CNNs, we conduct additional experiments with the following preprocessing methods: barely normalized images, (partially) segmented images based on provided masks in the dataset, and gamma-corrected images with different levels of constant settings.

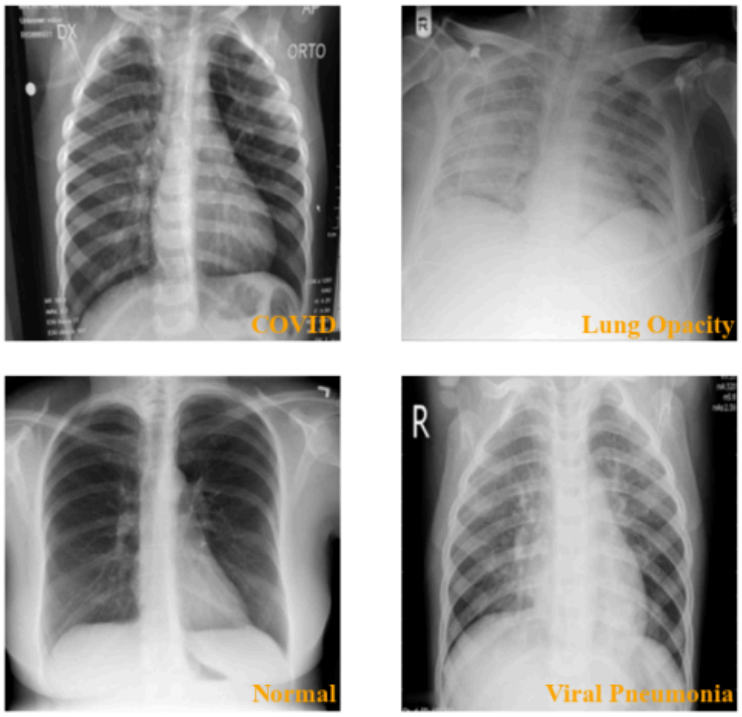


Figure 12: Dataset Samples

Table 3: Optimized training configuration for each experimented model after fine-tuning

Model	Preprocess	Learning Rate	Batch Size	Weight Decay
EfficientNet-B0	gamma correction (-0.8)	1.00e-03	16	1.00e-05
EfficientNet-B5	bare normalization	1.00e-03	16	1.00e-05
EfficientViT-B3	bare normalization	2.00e-06	32	1.00e-02
MViT2	bare normalization	2.00e-06	16	1.00e-02
ViT-Base-patch8	bare normalization	2.00e-06	16	1.00e-01
ViT-Base-patch16	bare normalization	1.00e-05	64	1.00e-02
ViT-Base-patch32	bare normalization	1.00e-05	64	1.00e-01
Proposed model	bare normalization	1.00e-05	64	5.00e-03

2. Model Configuration

We employ transfer learning with pre-trained models on the COVID-19 chest X-ray dataset. Through fine-tuning on preprocessing methods (bare normalization, segmentation, and gamma correction), learning rate (range [$1e{-}3, 1e{-}6$]), batch size (within 16, 32, 64, 128), and weight decay (range [$1e-1, 1e-6$]), we obtain optimized settings (as reported in Table 3) and corresponding models. Additionally, the cross-entropy function [16] served as the loss function, and AdamW/Adam [17, 18] were utilized as optimizers for experimented models. Then, observing the superior performance of ViT-B16 compared with others in the early stage of experiments, we further fine-tuned it by setting different numbers of encoder blocks inside. The configurations and results on different numbers of block settings are shown in Table 4.

Table 4: Training configuration and Test results after fine-tuning on ViT-16

(batch size: 64, learning rate: 1e-5, preprocessed with bare normalization)

Number of Encoder Blocks	Weight Decay	Accuracy (%)	Recall (%)	Precision(%)	F1 Score(%)
5	1.00e-02	94.37	98.31	98.58	98.44
7	1.00e-02	95.50	99.15	98.87	99.01
8	1.00e-02	95.55	98.87	98.59	98.73
9	1.00e-03	95.41	98.87	99.15	99.01
10 (proposed)	5.00e-03	95.79	98.58	98.86	98.72
11	1.00e-02	95.64	99.15	98.87	99.01
12 (original)	1.00e-02	95.22	98.31	98.86	98.58

3. Result Analysis

To assess the training effectiveness of the experimented models, we monitor the loss function on both the training and validation datasets. The loss curves for each model (as depicted in Fig. 11) are obtained through the fine-tuning process detailed in Section 3.2. Notably, weight decay plays a crucial role in achieving optimal performance.

Convergence Rate: Under the optimized settings reported in Table 1, EfficientNet-B0, ViT-Base-patch8, ViTBase-patch16, and ViT-Base-patch32 achieve minimum loss within 10 iterations, demonstrating their superior convergence speed. MViTv2 and EfficientNet-B5 achieve minimum loss between 10 to 15 iterations, whereas EfficientViT-B3 requires nearly 30 iterations to reach its minimum.

Overfitting: We employ the cross-entropy loss function and implement early stopping with patience of 30 epochs to mitigate overfitting. As observed in the training and validation loss curves depicted in Fig. 11, the extent of overfitting varies across models. Notably, EfficientViT-B3 exhibits a stable fitting behavior, while EfficientNet models demonstrate a tendency towards relatively unstable overfitting on our experimental dataset.

Our experimented models demonstrate relatively promising results after fine-tuning, with validation loss values typically fluctuating in the range of 0 to 0.2. Trading off with the convergence rate, overfitting, and loss value, we select the ViT-Base-patch16 structure for further tuning, where our proposed model is generated. Table 4 shows the corresponding configurations and results. Compared to other models, our proposed model achieves lower training loss and

higher accuracy, although a slight overfitting gap is still present. A detailed analysis of the prediction performance is provided in the next subsection.

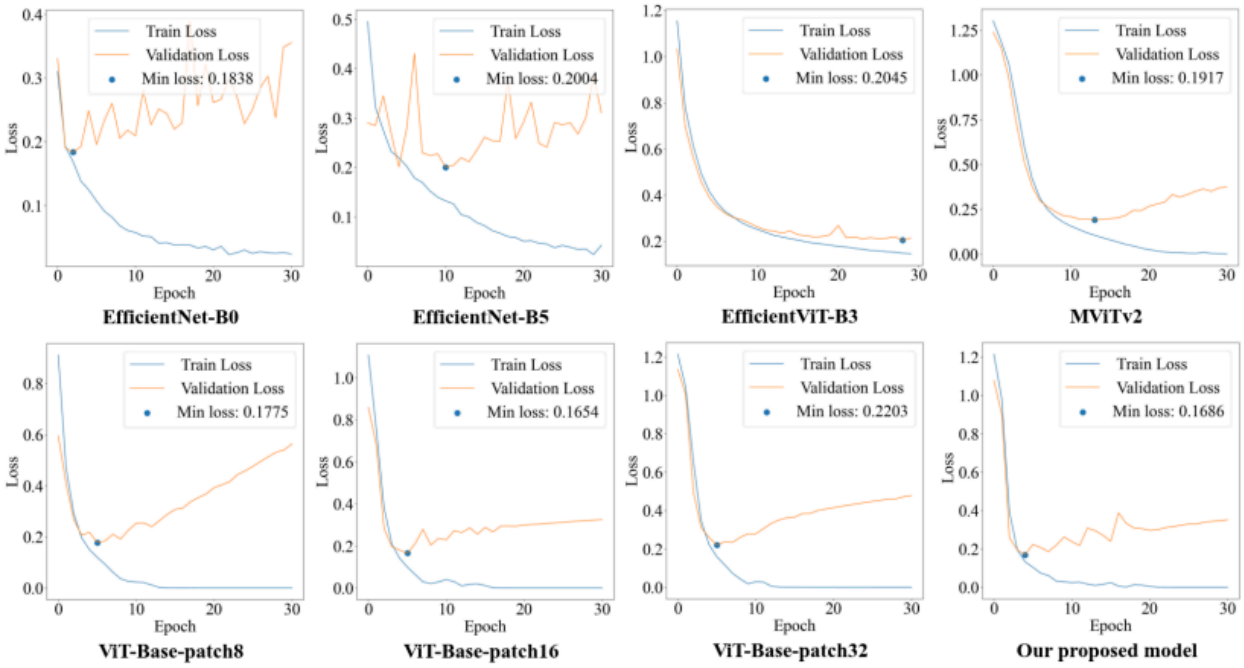


Figure 13: Train and validation losses during training process

After the training process, we proceed with testing on the test set, evaluating the results based on the specified criteria, and finally comparing them with existing models. Firstly, we observe that the training loss accurately reflects the achieved accuracy when the experimental models reach a high accuracy score (around 93% to 96%) as shown in Table 4. However, this is also balanced by the training time per epoch; models with higher accuracy often require more resources to train as the complexity of the model increases. For instance, the ViT-Base-patch8 has excellent performance with accuracy (95.41%), recall (99.15%), precision (98.59%), and F1 score (98.87%) among finetined models. However, with a trainable parameter of 85.81 M, the

training time taken in each iteration is several times longer than in other models (6.2 times longer in comparison to our model). Although it has a similar number of trainable parameters ((224⁄8)2 × 82 × 3) as ViT-Base-patch16's ((224⁄16)2 × 162 × 3), ViT-Base-patch8 has more tokens (patches) ((224⁄8)2) which exhibits more computational cost than flattened features (8 2). EfficientNetB5, with the highest recall value of 99.43% among experimented models, shows its high sensitivity to detect COVID-19 cases, although it has a relatively bad classification ability in four-class classifying (accuracy of 93.34%). MViT2 and EfficientViTV3, although maintaining moderate performance in the experimented four-class classification COVID-19 chest X-ray dataset, demonstrate outstanding accuracy in multi-class classification with other high-complex datasets [19, 20].

Table 5: Comparison with existing studies (Test Result, Binary: COVID-19 vs. non-COVID, Three-Class: COVID-19 vs. Normal + Lung Opacity vs. Pneumonia).

Prediction Model	Classification Type	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	Specificity (%)
Basic CNN [35]	Binary	93.99	89.16	95.63	87.41	88.28
Cycle GAN [36]	Binary	93.75	84.00	90.00	-	97.00
CovXmlc (VGG16 + SVM) [37]	Binary	95.00	95.00	96.00	95.00	96.00
Hybrid Deep Learning Mdoel [38]	Binary	92.00	92.00	93.00	92.00	98.68
Multi-Model [39]	Binary	97.83	95.45	100.00	97.67	100.00
DenseNet201 [28]	Three-Class	95.11	94.55	94.56	94.53	95.59
DenseNet121 [40]	Three-Class	93.5	92.59	93.44	93.00	-
Resnet50-BiLSTM [41]	Three-Class	98.51	98.51	98.52	98.51	-
ViT-B32 [12]	Three-Class	96.00	96.00	-	-	96.00
This work	Binary	99.57	98.58	98.87	98.73	99.76
	Three-Class	99.57				
	Four-Class	95.79				

For our model, which is based on ViT-Base-patch16 with specific adjustments, we remove unnecessary layers and retain crucial layers for feature extraction, reducing the model size. This, coupled with reduced trainable parameters (from 85.81 M to 71.62 M) and training time (from 313.94 s to 264.79 s), results in our model achieving the highest accuracy of 95.79% by mitigating overfitting. On the other hand, when delving deeper into the confusion matrix of the proposed model, as depicted in Fig. 5, we can discern the model's effectiveness. The most crucial class, COVID-19, achieves an accuracy of 98.58% based on *TP* and *FN*. However, the lung opacity and normal class exhibit suboptimal performance, which can be attributed to their relatively large image count and potential challenges posed by image quality in the classification task. Thus, we hypothesize that the classification between lung opacity and normal images is the most challenging bias existing in this COVID-19 chest X-ray dataset. Additionally, by examining the ROC curves and AUCs (as shown in Fig. 12), we conclude that our proposed model is confident regarding the decision it makes, and an almost ideal ROC curve is plotted on the prediction of COVID-19 and pneumonia cases, with a COVID-19 AUC value of 0.9993, lung opacity AUC value of 0.988, normal AUC value of 0.9896, and pneumonia AUC value of 1. The slightly lower AUC value (0.988 and 0.9896) of lung opacity and normal classes can further prove their biases in our model training. To comprehensively assess the advancements offered by our proposed model, we compare its accuracy and other metrics with existing research studies, as shown in Table 5. A major focus in the existing literature has been the binary classification of COVID-19 and non-COVID-19 cases. While such approaches achieve high accuracy (up to 97.83%), they lack the granularity needed for a more comprehensive diagnosis. Our multi-class

classification approach addresses this limitation by focusing on identifying specific classes within the spectrum of COVID-19 and related conditions. Remarkably, our model achieved an outstanding accuracy of 99.57% when classifying the COVID-19 class. Furthermore, the high recall (98.58%) and leading specificity (99.76%) demonstrate the model's exceptional sensitivity in detecting COVID-19 features while accurately distinguishing other conditions. These results highlight the model's potential to improve the precision of medical diagnosis using chest X-ray images.

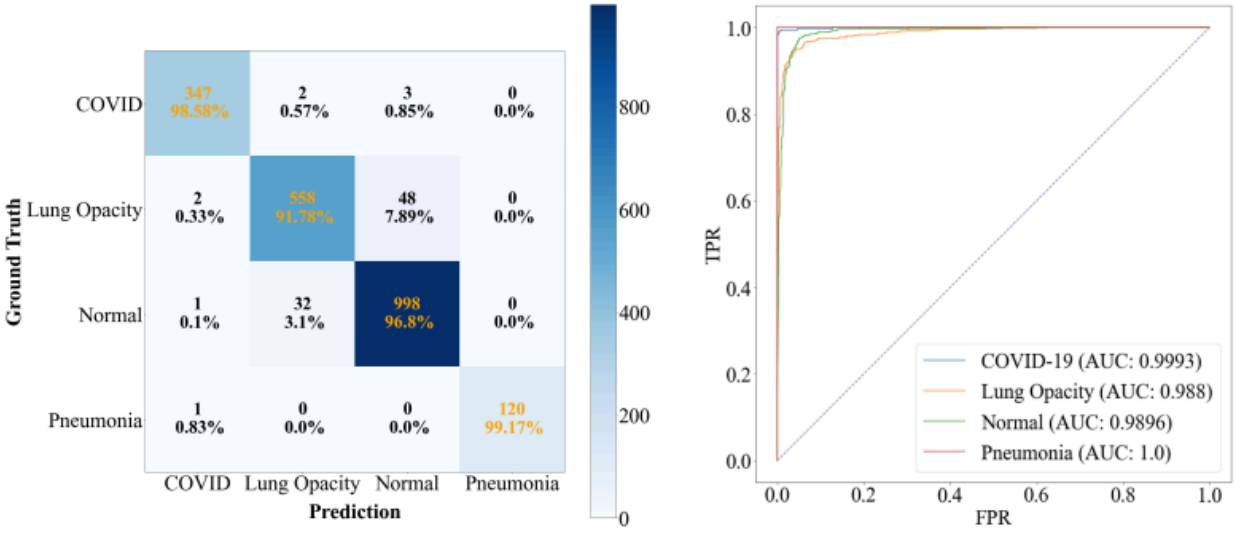


Figure 14: Confusion matrix (left) and ROC curves with AUC (right) for our proposed model.

Research Conclusion:

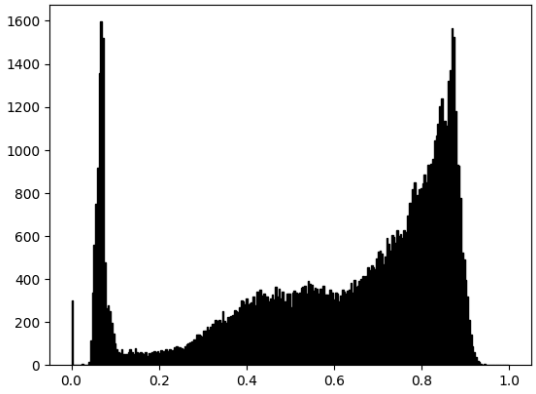
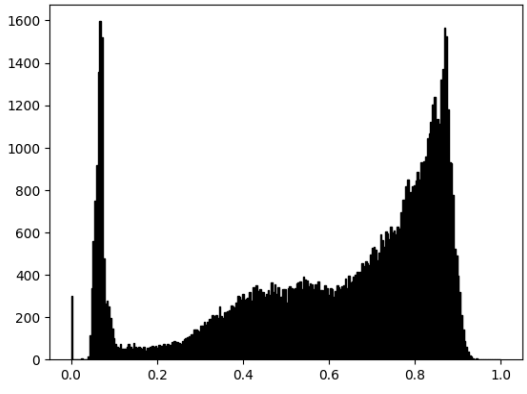
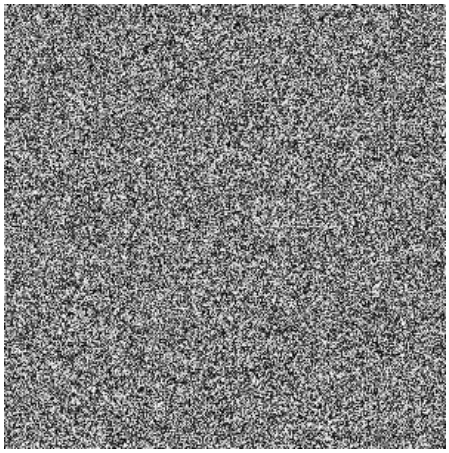
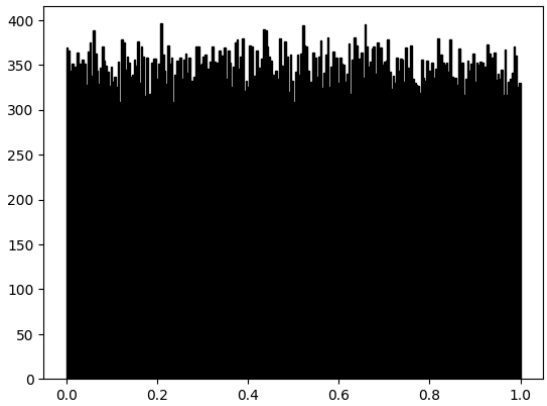
This study investigates the effectiveness of computer vision models, particularly vision transformers, for diagnosing COVID-19 from chest X-rays. We meticulously analyze and compare the performance of four diverse architectures: vision transformer, EfficientNet, MViT,

and EfficientViT. We assess overfitting behavior in both CNN and VT models by visualizing the training and validation loss curves. Our analysis reveals several techniques to mitigate overfitting, with weight decay emerging as a particularly impactful method. Additionally, we explore the influence of varying the number of encoder blocks within the transformer models. Through careful fine-tuning and modifications, we present our proposed fine-tuned ViT model (FT-ViT) as a leading performer, achieving exceptional accuracy. FTViT demonstrates not only outstanding accuracy in four-class classification (95.79%) but also a remarkable 99.57% accuracy in a clinically relevant three-class grouping, along with consistently high performance in binary classification scenarios. Stringent validation using quantitative metrics, ROC curves, and Grad-CAM visualizations solidifies FTViT's effectiveness. Our validation process also identified potential biases within the dataset, particularly between lung opacity and normal images. We believe this challenge can be addressed through improved image segmentation and balancing techniques. This comparative analysis across various architectures highlights the superiority of our fine-tuned VT approach for COVID-19 diagnosis. We envision that FT-ViT, with its exceptional accuracy, could contribute to more efficient and potentially faster COVID-19 diagnosis, particularly as a screening tool in clinical settings. Furthermore, our findings provide valuable insights for future research in medical image analysis tasks. Beyond specific instructions, this work opens doors for exploring transfer learning from other medical imaging datasets to broaden the applicability of vision transformer-based models in the healthcare domain.

5.2 Encryption Testing

To accurately test the accuracy of our project, multiple steps were implemented into our code to ensure data integrity was upheld to our client's expectations. Given that this project is handling image data, displaying the image and histograms showing the pixel values at each step in the process can be used to benchmark our results. Therefore, that is exactly what our team did to test our encryption algorithm. In Table 6 (shown on the following page), you will see many images. These images represent the medical scan before, during, and after the encryption/decryption process as well as histograms that show all the possible data values of each pixel in the image (0-1) and the number of pixels that have that specific pixel value. Analyzing the tables below shows that before and after the encryption and decryption process, the images and their associated histograms remain identical, showing no data loss and successful encryption. Therefore, having this process combined with our Visual Transformer deep-learning neural network, successful diagnoses as well as secure data transfer or ensured in the medical field.

Table 6: Image Development and Pixel Value Comparison Throughout Encryption/Decryption Process

Image Before Encryption/Decryption 	Image After Encryption/Decryption 
Graph Showing Pixel Values (x) and Quantity (y) before encryption 	Graph Showing Pixel Values (x) and Quantity (y) after decryption 
Encrypted Image 	Graph Showing Pixel Values (x) and Quantity (y) after encryption 

6. Conclusion

We have successfully completed all the assigned tasks in our capstone project, which included developing a model capable of predicting medical-related data and implementing encryption techniques to protect the privacy of transmitted medical data.

Specifically, we have composed a paper that summarised our research on AI diagnosing models on the COVID-19 chest X-ray dataset. After meticulous revision, our paper was announced to be accepted for publication by Healthcare Analytics | Journal | ScienceDirect.com by Elsevier. In this research, we conducted experiments through a vast range of computer vision models, which include state-of-the-art transformer structures. We finally got our proposed fine-tuned Vision Transformer model which has an accuracy of 95.79% on four-class classification, and 99.57% on three-class classification and binary classification. To verify the efficiency of our model and explore potential biases within the dataset, we utilized several evaluation metrics like the F-1 score (98.73%), precision (98.87%), specificity (99.76%), and recall (98.85%), drew the confusion matrix and receiver operating characteristic (ROC) curve, and used the Grad-CAM visualization method.

Moreover, after deploying the homomorphic encryption to encrypt and decrypt the chest X-ray images with the Kyber package, we extensively devised our AI Telehealth system, which demonstrated the whole working procedure online. Users can access this website to experience the products of our project.

Last but not least, discovering the lack of an intuitive demonstration of a dataset in the field of human-sense visualization in image classification, we created the Animint2GradCAM software by combining Python and Animint2 R package, which displayed our tested dataset (696 cases, 1.5 GB) with index, confidence, prediction results, and GradCAM graphs, to help researchers better understand how classification models make decisions.

COVID-19 diagnosis is just the starting point of AI Telehealth project. Currently, we are planning to publish our entire system (AI diagnosing + homomorphic encryption + interactive visualization feedback). We believe and hope this pattern can be applied to other fields of medical image diagnosis and make fewer people suffer pain from illness around the world.

Bibliography

[1] Stewart, Conor. "AI/machine learning in hospitals in the U.S. in 2021," Statista. https://www.statista.com/statistics/1249788/ai-machine-learning-in-hospitals-in-the-us/#statisticContainer (accessed Dec. 02, 2023).

[2] Kumba Sennaar, "How America's 5 Top Hospitals are Using Machine Learning Today | Emerj," Feb 19, 2019 https://emerj.com/ai-sector-overviews/top-5-hospitals-using-machine-learning/

[3] Davenport T, Kalakota R. The potential for artificial intelligence in healthcare. Future Healthc J. 2019 Jun;6(2):94-98. doi: 10.7861/futurehosp.6-2-94. PMID: 31363513; PMCID: PMC6616181.

[4] "Future Healthcare Journal," Future Healthcare Journal, vol. 9, no. 1, Mar. 2022, Accessed: Jun. 01, 2022. [Online]. Available: https://www.rcpjournals.org/content/futurehosp

[5] P. Schwabe, "Kyber," pq-crystals.org. https://pq-crystals.org/kyber/

[6] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, C. Feichtenhofer, MViTv2: Improved multiscale vision transformers for classification and detection (2022). arXiv:2112.01526.

[7] Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, N. A. Alajlan, Classification of remote sensing images using EfficientNetB3 CNN model with attention, IEEE Access 9 (2021) 14078–14094. doi:10.1109/ACCESS.2021.3051085.

[8] H. Cai, J. Li, M. Hu, C. Gan, S. Han, EfficientViT: Multi-scale linear attention for high-resolution dense prediction (2023). arXiv:2205.14756.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: transformers for image recognition at scale (2021). arXiv:2010.11929.

[10] Seyyed-Kalantari, L., Zhang, H., McDermott, M.B.A. et al. Underdiagnosis bias of artificial intelligence algorithms applied to chest radiographs in under-served patient populations. Nat Med 27, 2176–2182 (2021). https://doi.org/10.1038/s41591-021-01595-0

[11] Raheman, F. The Future of Cybersecurity in the Age of Quantum Computers. Future Internet 2022, 14, 335. https://doi.org/10.3390/fi14110335

[12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74. keywords: {Visualization;Cats;Dogs;Computer architecture;Knowledge discovery}

[13] Sievert, Carson & Vanderplas, Susan & Cai, Jun & Ferris, Kevin & Khan, Faizan & Hocking, Toby. (2018). Extending ggplot2 for Linked and Animated Web Graphics. Journal of Computational and Graphical Statistics. 28. 1-26. 10.1080/10618600.2018.1513367.

[14] A. K. Tawsifur Rahman, Muhammad Chowdhury, COVID-19 Radiography Database, 2022. URL: https://www.kaggle.com/datasets/ tawsifurrahman/covid19-radiography-database, this database consists 3616 COVID-19 positive cases along with 10,192 Normal, 6012 Lung Opacity (Non-COVID lung infection), and 1345 Viral Pneumonia images and corresponding lung masks.

[14] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. Abul Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, M. E. Chowdhury, Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images, Computers in Biology and Medicine 132 (2021) 104319.

[15] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, M. B. I. Reaz, M. T. Islam, Can AI help in screening viral and COVID-19 pneumonia?, IEEE Access 8 (2020) 132665–132676.

[16] A. Mao, M. Mohri, Y. Zhong, Cross-entropy loss functions: Theoretical analysis and applications, in: Proceedings of the 40th International Conference on Machine Learning, ICML'23, 2023.

[17] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2014).

[18] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2017. URL: https://api.semanticscholar.org/CorpusID:53592270.

[19] H. Cai, J. Li, M. Hu, C. Gan, S. Han, EfficientViT: Multi-scale linear attention for high-resolution dense prediction, 2022. URL: https://api.semanticscholar.org/CorpusID:262824134.

[20] Y. Fang, S. Yang, S. Wang, Y. Ge, Y. Shan, X. Wang, Unleashing vanilla vision transformer with masked image modeling for object detection, 2023 IEEE/CVF International Conference on Computer Vision (ICCV) (2022) 6221–6230.