



REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

**CLASSIFICATION OF CORONA COVID 19
PATIENTS FROM CHEST X-RAY IMAGES**

Wael Sattam Atiyah ATIYAH

Master's Thesis

Supervisor

Asst. Prof. Dr. Ayça KURNAZ TÜRK BEN

Istanbul, 2023

CLASSIFICATION OF CORONA COVID 19 PATIENTS FROM CHEST X-RAY IMAGES

Wael Sattam Atiyah ATIYAH

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis CLASSIFICATION OF CORONA COVID 19 PATIENTS FROM CHEST X-RAY IMAGES prepared WAEL SATTAM ATIYAH ATIYAH and submitted on 21/12/2023 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Asst. Prof. Dr. Ayça KURNAZ

Supervisor

Thesis Defence Committee Members:

Asst. Prof. Dr. Ayça KURNAZ
TÜRKBEN

Department of Computer
Engineering,

Altınbaş University

Asst. Prof. Dr. Oğuz KARAN

Department of Software
Engineering,

Altınbaş University

Asst. Prof. Dr. Serdar KARGIN

Department of Biomedical
Engineering,

Arel University

I hereby declare that this thesis meets all format and submission requirements of a Master's thesis.

Submission date of the thesis to Institute of Graduate Studies: ____/____/____

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Wael Sattam Atiyah ATIYAH

Signature



ABSTRACT

CLASSIFICATION OF CORONA COVID 19 PATIENTS FROM CHEST X-RAY IMAGES

ATIYAH, Wael Sattam Atiyah

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Ayça KURNAZ

Date: December / 2023

Pages: 57

For medical professionals to successfully focus their attention and give therapy to patients, it is essential for them to have an accurate diagnosis and categorization of COVID 19. Because COVID 19 has developed into a leading cause of mortality, the medical community must give it significant scrutiny. As a result, one of our objectives is to create a structure, with the purpose of conducting research on the collection of photographs collected from the Kaggle platform. In order to accomplish this goal, we used image merging strategies into our research. At first, we made use of a Convolutional Neural Network, abbreviated as CNN. further on, a strategy was created by merging CNN with Gated Recurrent Units (GRU). This method, which we describe to as our recommended model, was further developed. Through a preprocessing phase that entailed combining pictures, we were able to increase the overall quality of the CT scans. In addition, the color format of the photographs was changed from BGR to RGB by us. In addition, the dataset containing the micro-CT images was broken up into sets for the purposes of testing, training, and validation. According to the confusion matrix analysis, the findings collected by the network (CNN) demonstrated a remarkable accuracy rate of 98%. In addition to that, the value of 97% was displayed in the correlation matrix of our CNN-GRU model.

Keywords: Classification, CNN-GRU, CNN, Deep Learning Algorithms, Image Segmentation Techniques.

TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT	v
LIST OF FIGURES.....	ix
ABBREVIATIONS.....	x
1. INTRODUCTION	1
1.1 BACKGROUND	1
1.2 AIM.....	3
1.3 HYPOTHESIS STATEMENT	4
1.4 RESEARCH QUESTIONS	4
1.5 PROBLEM DEFINITION	5
2. LITERATURE REVIEW	7
2.1 BACKGROUND	7
2.2 COVID 19 PANDEMIC HAS HAD AN IMPACT	10
2.3 DATA COLLECTION	11
3. METHODOLOGY	13
3.1 MANAGEMENT OF IMAGES	13
3.1.1 Discoverability of Images	14
3.1.2 CNN-GRU	16
3.1.3 CNN Designs	19
3.1.4 Dataset	22
3.2 CLASSIFICATION REPORT	25
3.2.1 Sparse_Categorical_Crossentropy	26
3.3 SOFTWARE FRAMEWORK	28

4. RESULTS	29
4.1 MODEL DESIGN.....	29
4.1.1 CNN GRU Model	29
4.1.2 CNN Model.....	35
4.2 DISCUSSION.....	42
4.2.1 Comparison of Accuracy with Previous Work	42
5. CONCLUSION	43
5.1 CONTRIBUTION	43
5.2 CONCLUSION.....	43
REFERENCES	45

LIST OF TABLES

	<u>Pages</u>
Table 2.1: Listing Some Earlier Analyses Of COVID-19 Patient Images Classification	9
Table 4.1: Best Results Achieved.....	33
Table 4.2: Best Outcome of Val. Accuracy.....	40
Table 4.3: Discussion the Table Illustrates Differences.	42
Table 4.4: Comparison of Accuracy with Previous Work.....	42



LIST OF FIGURES

	<u>Pages</u>
Figure 1.1: COVID 19 Classification.	2
Figure 2.1: How to Classify COVID-19 Photos.....	8
Figure 3.1: Methodologies of Our Application.	15
Figure 3.2: Visualize the Collected Data.....	22
Figure 4.1: Layers of A Network Architecture.	30
Figure 4.2: The Summary Results.	30
Figure 4.3: Model with the Adam Optimizer.	31
Figure 4.4: History of Model.....	31
Figure 4.5: Proposed Epochs.	32
Figure 4.6: Training for Both Validation Acc. and Loss.....	33
Figure 4.7: Accuracy Classification Report with Test Data.....	34
Figure 4.8: Accuracy Classification Report with Validation Data.....	34
Figure 4.9: Layers of A Network Architecture.	36
Figure 4.10: The Summary Results.	37
Figure 4.11: Model with the Adam Optimizer.	37
Figure 4.12: Fitting Model.	38
Figure 4.13: Training Val. for Both Acc. and Loss.....	39
Figure 4.14: Proposed Epochs.	40
Figure 4.15: Accuracy Classification Report with Test Data.....	41
Figure 4.16: Accuracy Classification Report with Validation Data.....	41

ABBREVIATIONS

AI	:	Artificial Intelligence
FSK	:	Frequency Shift Keying
IoT	:	Internet of Things
PAN	:	Personal Area Network
OFDM	:	Orthogonal Frequency Division Multiplexing
TDMA	:	Time Division Multiple Access



1. INTRODUCTION

1.1 BACKGROUND

The global health crisis caused by the COVID 19 pandemic has led to an increase, in sickness and death. Diagnosing COVID 19 quickly and accurately is crucial for identifying and treating individuals at an early stage. Chest X rays and CT scans are commonly employed to diagnose COVID 19. Analyzing these images can be time consuming and challenging especially during a pandemic with a high number of cases [1].

Deep learning methods, Convolutional Neural Networks (CNNs) have shown promise in analyzing medical images, including detecting and classifying COVID 19 cases from chest X rays and CT scans. By learning features from medical images deep learning models can effectively classify different types of COVID 19 images [2].

Utilizing learning for classifying COVID 19 images offers potential advantages such as improved diagnostic accuracy, reduced interpretation time and increased availability of diagnostic tools. Consequently, it is crucial to focus on the development and evaluation of learning models specifically designed for classifying COVID 19 cases. This area of research holds potential in impacting patient care and management.

Deep learning has become a tool in the field of medical image analysis, with CNNs being widely recognized as one of the prevalent models used for classification tasks.

These models have demonstrated performance compared to machine learning algorithms and even human experts, in specific tasks related to medical image analysis.

Regarding the classification of COVID 19 numerous studies have highlighted the potential of learning techniques. For instance, an article published in the journal IEEE Access showcased a CNN based model achieving an accuracy rate of 94.3% in distinguishing between COVID 19 and non-COVID 19 pneumonia cases based on chest CT scans.[3] Furthermore, a study published in Radiology; Cardiothoracic Imaging exhibited that a deep learning model successfully classified COVID 19 and non-COVID 19 cases based on chest X rays with an area under the operating curve (AUC) of 0.96.

Category	X-rays		
COVID-19			
Normal			

Figure 1.1: COVID 19 Classification.

Deep learning models designed for COVID 19 classification hold promise in aiding the diagnosis and management of patients especially in settings where access, to healthcare resources may be limited.

Experienced radiologists may face limitations. The development of accurate and reliable models necessitates extensive and diverse datasets. It also requires validation and evaluation to ensure that they are clinically relevant and applicable, in scenarios. The field of learning for classifying COVID 19 images is constantly evolving, holding potential for enhancing patient care and management during the ongoing pandemic and beyond.

Apart from classification learning techniques can be applied to medical image analysis tasks related to COVID 19. These tasks include segmenting, detecting and quantifying abnormalities in the lungs caused by the virus. For instance,[4] a recent study published in Radiology utilized learning to automatically segment lung abnormalities and measure their extent on chest CT scans in COVID 19 patients.

Furthermore, transfer learning can be employed with learning models by tuning pre trained models from other medical imaging tasks for COVID 19 classification [5]. This approach has shown promising outcomes in studies. Can reduce the requirement for extensive datasets during model training.

Nevertheless, using learning models for COVID 19 classification also presents challenges and limitations. One significant challenge is the lack of imaging protocols along with variations, in image acquisition methods and quality. These factors can impact the performance of learning models [6].

There is another hurdle to overcome which involves the risk of bias, in the data used for training. This bias can lead to models that perform differently among populations and groups of patients. In general, the use of learning to classify COVID 19 images is an area of research with great potential for enhancing patient care and management. Through research and advancements deep learning models have the potential to become a resource for healthcare professionals in combating COVID 19 as well, as future pandemics [7].

1.2 AIM

The objectives of utilizing learning, for the classification of images related to corona disease are as follows.

- a. Enhance the precision of diagnosis; Deep learning models can effectively differentiate COVID 19 from illnesses minimizing the likelihood of incorrect diagnoses.
- b. Improve efficiency; Employing learning can expedite the diagnosis and management of COVID 19 cases enabling patient isolation and treatment.
- c. Alleviate the workload on healthcare professionals; Deep learning models serve as a tool for clinicians reducing the burden on radiologists and other healthcare practitioners in areas with limited access to specialized expertise.

- d. Enable detection; Promptly identifying and isolating COVID 19 cases are crucial, in preventing virus transmission. Deep learning models can assist in detecting COVID 19 cases thereby reducing community spread risk.
- e. Enhance patient outcomes; Accurate and timely diagnosis can lead to improved outcomes by facilitating treatment and reducing illness severity and mortality associated with COVID 19.

1.3 HYPOTHESIS STATEMENT

Deep learning algorithms possess the capacity to augment the accuracy and efficacy of COVID-19 diagnosis through the classification of corona disease photos. This measure would effectively mitigate the burden on healthcare professionals, facilitate early identification, and eventually enhance patient outcomes.

1.4 RESEARCH QUESTIONS

Here are some possible research inquiries that could be explored.

- a. How effective are deep learning models, in categorizing COVID 19 images? How do their performance measures compare to those of machine learning algorithms and human experts?
- b. In what ways can deep learning models be optimized for COVID 19 image classification? This includes considering factors such as data augmentation, model architecture and hyperparameter tuning.
- c. What biases and limitations might arise when using learning models for COVID 19 image classification. How can we address or minimize them?
- d. How can we integrate learning models into workflows to enhance the diagnosis and management of COVID 19? What benefits and challenges might come with this integration?
- e. In what manner can deep learning models be employed to classify COVID 19 images from modalities (chest X rays CT scans) and imaging protocols? What implications does this have for practice and research?

- f. To what extent do learning models for COVID 19 image classification generalize, across patient populations and geographical regions? Which factors influence this generalizability?
- g. How do deep learning models stack up against tools, like RT PCR tests in terms of accuracy, speed and cost effectiveness for diagnosing COVID 19? Additionally, what are the ethical and social implications of using learning models for classifying COVID 19 images. How can we ensure equitable and responsible deployment of these technologies?

1.5 PROBLEM DEFINITION

The COVID 19 outbreak has emphasized the pressing requirement, for effective tools to assist in diagnosing and managing patients who are infected.[8] Medical imaging, like chest X rays and CT scans has become a method for identifying abnormalities in the lungs caused by COVID 19. However interpreting images can be time consuming and demands expertise, which can slow down the diagnosis process especially when access to radiologists and other healthcare professionals is limited.

Deep learning models have demonstrated promise in enhancing the accuracy and efficiency of COVID 19 diagnosis by categorizing images of corona disease. This helps reduce the workload on healthcare professionals enables detection and improves outcomes. Nevertheless, there are challenges and limitations associated with using learning models for classifying COVID 19 images. These include issues such as data bias, variations in imaging protocols and quality as applicability across diverse patient populations and geographical regions.

Consequently, it is crucial to develop learning models that're accurate, reliable and applicable across different patient populations and regions for classifying COVID 19 images [9] . These models can greatly assist in enabling precise diagnosis and management of infected patients when access to specialized expertise may be limited.

In conclusion the COVID 19 pandemic has underscored the necessity for efficient tools, for patients who have contracted this virus.

Medical imaging has become a tool, in identifying abnormalities related to COVID 19 in the lungs. However, interpreting these images can be time consuming. Requires specialized expertise. Deep learning models have shown promise in improving the diagnosis and

management of COVID 19 by categorizing images of the disease thus reducing the burden on healthcare professionals. This technology also facilitates detection. Enhances patient outcomes. However, there are challenges and limitations that we must address when using learning models for COVID 19 image classification. Developing reliable and adaptable learning models specifically for classifying COVID 19 images can greatly assist in swiftly diagnosing and managing infected patients especially in situations where access, to specialized expertise may be restricted.



2. LITERATURE REVIEW

2.1 BACKGROUND

Recent research has been exploring the potential of learning models, in classifying images related to COVID 19 and the results have shown promise. For instance, Wang et al. (2020) developed a learning model that achieved an accuracy of 92.4% on the test set when classifying COVID 19, bacterial pneumonia and normal chest X ray images [10]. Similarly, Apostolopoulos and Mpesiana (2020) proposed a learning framework that achieved an accuracy of 95.6% on the test set by classifying COVID 19 cases from CT scans [11].

Apart from accuracy there have also been studies focusing on the interpretability and explainability of learning models used in COVID 19 image classification. Narin et al. (2021) for example introduced a learning model that integrates class activation maps to identify the regions within chest X ray images that contribute most significantly to the classification decision making process. This approach enables understandable decision making [12].

However, it is important to acknowledge challenges and limitations associated with employing learning models, for COVID 19 image classification. One major concern is data bias since many datasets used for training these models are biased towards populations or imaging protocols. This can potentially restrict the applicability and generalizability of these models.

Moreover, the effectiveness of learning models can be influenced by variations, in imaging protocols and the quality of images especially when different modalities or sources are employed.

It is also crucial to consider the social implications associated with utilizing learning models, for COVID 19 image classification. For instance, concerns arise regarding privacy and data protection particularly when significant amounts of data are utilized to train these models. Additionally, there is a risk of exacerbating health inequalities if these models are not developed and implemented in a responsible manner.

In general, existing literature suggests that deep learning models hold promise in enhancing the diagnosis and management of COVID 19. These models can accurately classify images related to the disease thus reducing the workload, on healthcare professionals aiding in detection and improving patient outcomes. However, it is crucial to conduct research to

address the challenges and limitations associated with these models. Additionally, it is essential to develop and deploy them in a responsible manner.

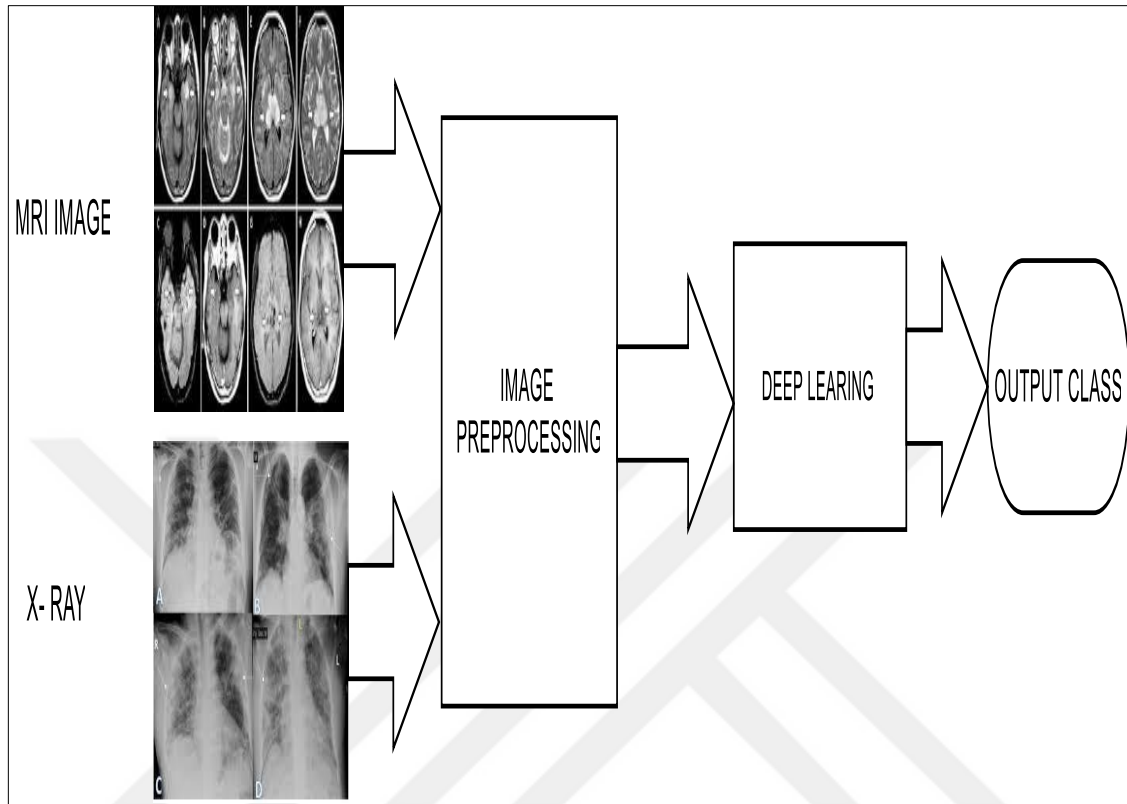


Figure 2.1: How to Classify COVID-19 Photos.

Several studies have also explored the applicability of learning models for COVID 19 image classification across patient populations and geographical regions. For instance, Wang et al. (2021) assessed their learning models adaptability for COVID 19 classification using data from three datasets originating from various geographic areas [13]. The study revealed that the model performed consistently across all datasets. Similarly, Mei et al. (2021) developed a learning model for classifying COVID 19 based on chest X ray images. Evaluated its accuracy, on data obtained from different hospitals and countries.[14] The findings demonstrated that the model achieved accuracy and showed generalizability across populations and imaging protocols.

Furthermore, ongoing research is exploring how deep learning models can contribute to COVID 19 prognosis and treatment.

For instance, in a study conducted by Yang and colleagues (2020) they created a machine learning system that can predict the severity of diseases and forecast patient outcomes based on chest CT scans. [15] The results were remarkably accurate showcasing the potential of models in assisting treatment decisions.

Nevertheless, there are difficulties associated with utilizing learning models, for COVID 19 prognosis and treatment. These challenges include the requirement for datasets of quality and the possibility of biases within the data. Furthermore, due to COVID 19s multifaceted nature, which affects organ systems differently developing precise and universally applicable deep learning models, for prognosis and treatment poses further obstacles.

To summarize existing literature suggests that deep learning models hold promise in enhancing COVID 19 diagnosis, management, prognosis, and treatment. However, it is crucial to conduct research to address the challenges and limitations linked to these models while ensuring their development and responsible implementation.

Table 2.1: Listing Some Earlier Analyses Of COVID-19 Patient Images Classification

Study Title	Year	Classification Technique	Dataset	Accuracy
"COVID-19 detection from chest X-ray images using deep learning and fuzzy rough set"[16]	2021	Deep learning and fuzzy rough set	958 X-ray images (416 COVID-19, 542 non-COVID-19)	97.30%
"COVIDX-Net: A Framework of Deep Learning Classifiers to Diagnose COVID-19 in X-Ray Images"[17]	2020	Deep learning (COVIDX-Net)	1,280 X-ray images (positive for COVID-19, normal, or pneumonia)	94.55%
"Deep Transfer Learning-based Automated Detection of COVID-19 from Chest X-ray Images using Weakly-Supervised Feature Learning"[17]	2020	Deep transfer learning	165 COVID-19 X-ray images, 165 non-COVID-19 X-ray images	98.50%

Table 2.1: Listing Some Earlier Analyses Of COVID-19 Patient Images Classification "Table Continued".

"COVID-19 Detection through Transfer Learning Using Multimodal Imagery"[19]	2020	Transfer learning	1,203 chest X-ray and CT images (positive for COVID-19 or not)	96.67%
"COVID-19 Detection Using CT Scans: Deep Learning Approach"[20]	2020	Deep learning (CNN)	3,910 CT images (positive for COVID-19 or not)	90.80%

It's worth noting that there have been studies conducted on this topic and the table mentioned above only represents a limited selection of current research. It's important to consider that evaluating the effectiveness of a classification model should not solely rely on accuracy, as the determining factor. It's possible that these studies used assessment criteria or employed distinct research methods.

Please note that COVIDx is a dataset consisting of chest X ray images specifically gathered from individuals diagnosed with COVID 19.

The COVID CT dataset includes a collection of computed tomography (CT) images obtained from individuals diagnosed with COVID 19.

The RSNA dataset encompasses a collection of chest X ray images illustrating individuals diagnosed with conditions [2].

2.2 COVID 19 PANDEMIC HAS HAD AN IMPACT

Global Economic Downturn, the pandemic has caused a recession, comparable to one of the most profound recessions witnessed since the Great Depression. Numerous countries have experienced GDP growth, business closures and an increase in unemployment rates.

Disruption in Supply Chains; Lockdowns, travel restrictions and quarantine measures have disrupted supply chains. The closure of factories and distribution centers coupled with transportation constraints has led to delays in producing and delivering goods.

Changes in Consumer Behavior; With people staying at home to prevent the virus spread, consumer behavior underwent shifts. There was a surge in demand for online shopping,

digital services and home entertainment. Conversely sectors such as travel, tourism and hospitality have suffered.

Government Measures for Support; Governments worldwide have implemented stimulus packages and financial aid programs aimed at supporting individuals, businesses and healthcare systems. These measures seek to alleviate the impact while providing a safety net, for populations.

Unemployment and Job Loss; Many businesses faced challenges. Had to let go of employees or reduce their working hours. As a result, there was an increase, in unemployment rates leading to a decline in consumer spending.

Stock Market Volatility. Financial markets experienced fluctuations as investors reacted to the changing economic conditions and uncertainties surrounding the duration and severity of the pandemic.

Debt and Fiscal Concerns; The response to the pandemic required governments to increase their spending resulting in higher levels of public debt in many countries. This raised concerns about term sustainability and the potential for higher taxes in the future.

Global Trade Impact, international trade was disrupted due to travel restrictions and border closures which affected the movement of goods and services between countries.

Healthcare and Biotechnology Industry; The pandemic prompted investments in healthcare and biotechnology sectors leading to advancements in vaccine development and medical research.

Inequality and Social Impact; The pandemic brought attention to existing economic inequalities with vulnerable populations being disproportionately affected by both the virus itself and its economic consequences.

Please note that circumstances may have changed since my update. The recovery of the economy will depend on factors such as vaccination rates, government policies, international cooperation, as well, as the emergence of new COVID 19 variants [22].

To stay updated with the information and analysis it is advisable to refer to reliable economic sources and research institutions.

2.3 DATA COLLECTION

A dataset refers to a collection of data that is typically organized in a format and is used for analysis, research or machine learning purposes. In the context of classifying pictures of

individuals, with COVID 19 a dataset would consist of a compilation of images (such as X ray or CT scans) that have been labeled with information regarding whether the patient has COVID 19 or not. These images are then utilized to train machine learning models in order to classify obtained images as either COVID 19 positive or negative.

Developing a high-quality dataset for machine learning plays a role in constructing models. It involves selection and labeling of images well as ensuring that the dataset is well balanced and representative of the population under study. When it comes to COVID 19 classification this may require gathering images from sources and ensuring a mix of positive and negative cases within the dataset.

It's important to note that obtaining access to high quality datasets can be challenging in this field due to privacy regulations surrounding medical imaging data. This is why numerous research groups and organizations have made efforts to create and share datasets specifically designed for COVID 19 classification purposes, such, as the COVID 19 Image Data Collection and the COVID 19 CT segmentation dataset.

Data collection refers to the process of gathering information or data for analysis, research or machine learning purposes. In the case of classifying pictures of COVID 19 patients' data collection involves acquiring images, like X rays or CT scans and labeling them with details indicating whether the patient has COVID 19 or not.

There are methods for collecting such data. One method is to obtain images from hospital databases where patient information is already accessible. This approach offers convenience as it allows researchers to quickly access a volume of data. However, it is crucial to ensure that the collected data is anonymized and complies with privacy regulations.

Another approach involves gathering images through a network of professionals or clinics. This method proves beneficial in acquiring representative data from sources within the population being studied. However, it may require time and resources compared to collecting data from hospital databases.

Regardless of the chosen method, accuracy and consistency in labeling the collected data are factors. Achieving this may involve seeking assistance, from professionals or trained annotators who can review the images thoroughly and provide labels.

Moreover, it is crucial to guarantee the data reliability and also to have a number of cases (individuals, with COVID 19) in the dataset to train machine learning models, with utmost effectiveness.

3. METHODOLOGY

In this chapter we will discuss the proposed approach, to using AI for identifying COVID 19 images, in data. The approach involves three stages: instance segmentation, object detection and image classification. Additionally, this chapter provides an overview of the methodology we propose. We will also showcase illustrations of our methodology at the designated location. This study thoroughly explains the algorithms used and provides a description of the parameters and optimizers employed. Furthermore, it includes an examination of our data collection techniques. Lastly, we will carefully discuss the findings from this investigation.

3.1 MANAGEMENT OF IMAGES

Image management is incredibly important, in industries like computer vision, web development, graphic design and digital marketing. This paper provides an analysis of factors and best practices when it comes to managing images.[23].

To ensure an organized collection of photos its recommended to create a system of folders with names. This will make it easier to find photos when needed. One possible way to categorize photos is by their content origin or purpose. You can have folders for types of visual assets like product photos, background images, icons, and other relevant elements. To avoid confusion and quickly identify images using naming conventions for image files is advisable.

Including metadata and tags with images is highly recommended. This can include details like the title, description, keywords and copyright information. It proves useful when managing collections of images. Some image formats, like JPEGs EXIF (Exchangeable Picture File Format) have built in metadata that includes camera settings, geographical coordinates and time information. The subject we're discussing is, about version control and backup systems. To prevent the risk of losing data due to device issues or accidental deletions it's an idea to create backups of your picture files. When working on image projects its recommended to use version control systems like Git to track changes and keep multiple versions of pictures.

Optimizing and compressing images is a process. To improve the loading speed of websites and applications it's crucial to optimize the sizes of picture files. Image compression technologies can reduce file size while maintaining image quality.

One possible approach to boost performance in web development is by incorporating pictures or providing image sizes based on the device's resolution. The use of Content Delivery Networks (CDNs) and cloud storage has become increasingly common in technology infrastructure. For scalable storage of picture assets, it's advisable to utilize storage services such as Google Drive, Dropbox and Amazon S3. Content Delivery Networks (CDNs) can be employed for picture delivery reducing server load and improving image retrieval speed for users, across different regions.[24]

3.1.1 Discoverability of Images

It's an idea to consider the option of creating a database, for images or using image management software that makes it easier to search and filter photos based on different criteria. To improve the searchability of images within an image management system its recommended to use file names and keywords.

We're discussing the topic of image formats and file formats. When choosing the file format for an image it's important to consider its content and intended purpose. The JPEG format is commonly used for images while the PNG format is specifically designed for images that require transparency. On the hand the SVG format is primarily used for creating and displaying vector drawings.

If necessary, it might be required to convert photos into formats while ensuring that image quality and aspect ratio are preserved. Copyright and permissions are crucial in settings. When using photos, it's essential to comply with copyright laws and licensing agreements. To meet your needs, make sure you have authorization or use photos, with appropriate permissions. To avoid any issues maintaining a comprehensive record of picture license information is imperative. By following these guidelines individuals can ensure that their visual assets are organized readily available and optimized for purposes such, as website design, artificial intelligence applications and creating engaging visual content.

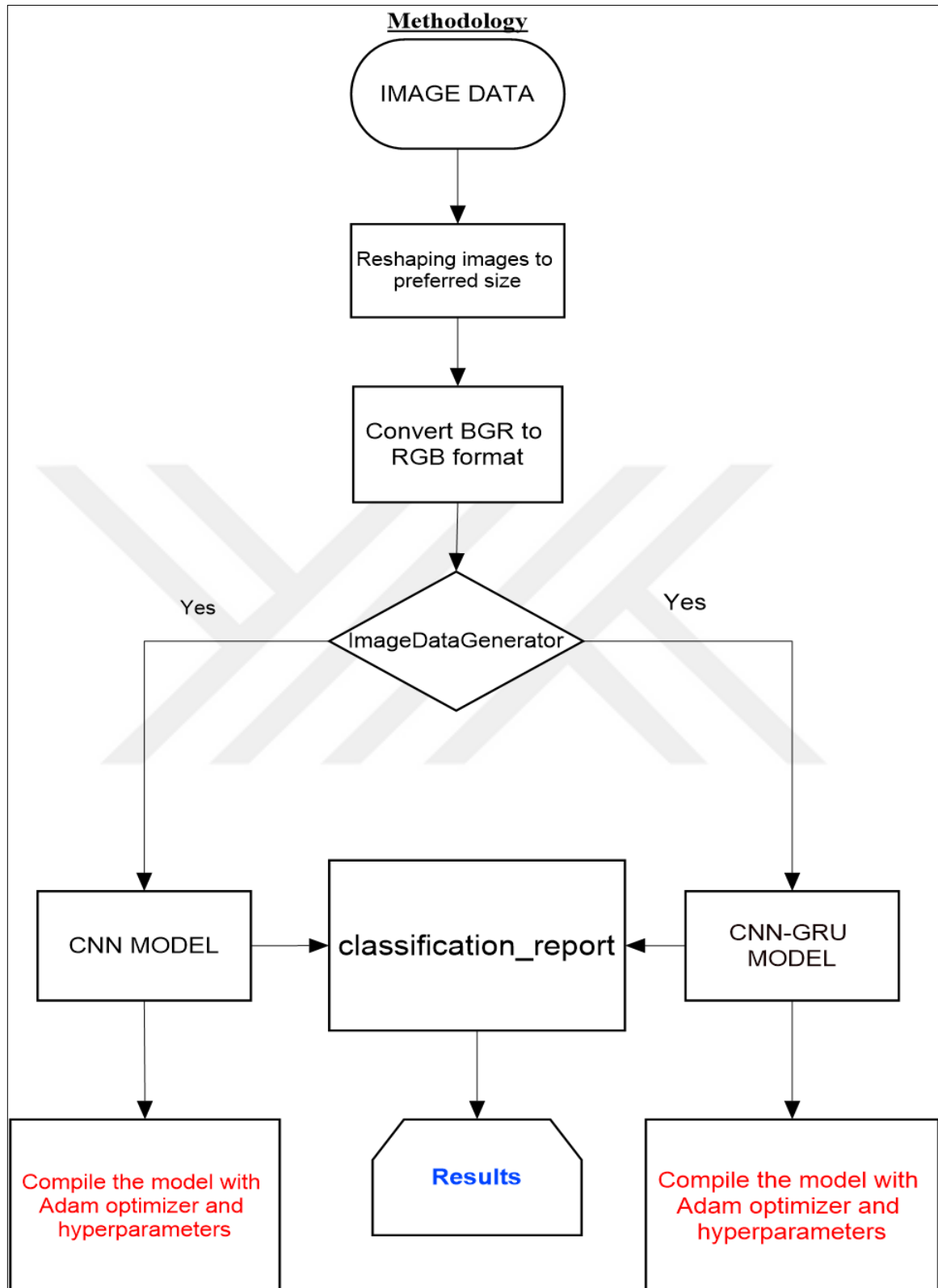


Figure 3.1: Methodologies of Our Application.

3.1.2 CNN-GRU

The CNN GRU neural network design combines the power of networks (CNNs) and gated recurrent units (GRUs). This hybrid architecture is commonly used for tasks that involve data, [25] with dependencies, such as video analysis, [26] action identification and natural language processing. Let's take a look at each component of this architecture.

CNNs, also known as Convolutional Neural Networks, are primarily used for processing and extracting features from data like images or 2D representations of sequential data (e.g. spectrograms for music). [26] They employ layers to identify patterns and hierarchical features in localized areas of the input data. The subsequent layers then. Process these extracted characteristics.

On the hand recurrent neural networks (RNNs) with gated units (GRUs) excel at modelling sequential data with long range dependencies. GRUs are an improvement over RNNs like LSTMs due to their structure and ability to mitigate the vanishing gradient problem. With gating mechanisms that regulate information flow GRUs can selectively. Forget information, over time [27].

By combining both CNNs and GRUs we can leverage the advantages offered by each architecture to tackle tasks requiring both local patterns extraction and capturing dependencies. CNN GRU networks have proven to be highly effective, for applications that involve both sequential patterns. These networks leverage the power of Convolutional Neural Networks (CNNs) to extract characteristics from the input data while Gated Recurrent Units (GRUs) capture temporal correlations between these features [28]. Here's a high-level overview of how a CNN GRU architecture structured.

- a. Input Layer: This is where the network receives its input, which could be an image or a series of data [29].
- b. Convolutional Layers: These layers analyze the input data. Identify spatial properties that are relevant to the task at hand [30].
- c. Pooling Layers: Additional layers can be included to downscale the dimensions and simplify calculations [31].
- d. Reshape; If the output from the CNN layer is in 2D format (like an image) it may need to be reshaped so that it can work effectively with the RNN component.[32]
- e. Gated Recurrent Units (GRUs); These layers process the output from the CNN layer in a sequence capturing dependencies and patterns.[33]

- f. Connected Layers; Optionally connected layers can be added for further processing and constructing the final result.[34]
- g. Output Layer: The final layer generates results based on tasks like classification or regression.[35].

By combining CNNs and GRUs in this way CNN GRU architectures offer a solution, for tasks that require both understanding and sequential analysis.

The specific architecture and hyperparameters of the CNN GRU model may vary depending on the task and dataset, at hand. To find the performing model researchers and practitioners often experiment with configurations.

Let's delve into an exploration of the key features and advantages offered by the CNN GRU architecture.

Utilizing CNNs for Feature Extraction:

When dealing with data like photos or time series data with a 2D structure leveraging CNNs as feature extractors can be highly beneficial. The convolutional layers of CNNs can automatically identify patterns and features from sections of input data. These inherent characteristics capture information relevant to the given task.[36].

Temporal Modeling with GRUs:

useful spatial characteristics have been extracted by CNNs GRUs come into play to model dependencies. GRUs are particularly well suited for tasks where order and temporal contexture important because they can capture long range relationships in input. They possess the ability to store knowledge over time and distribute it to time steps enabling efficient comprehension of sequential patterns by the model.[37].

Combining Spatial and Temporal Information, the power of the CNN GRU architecture lies in its integration of both temporal information, through combining CNNs with GRUs. This combination proves effective when analyzing spatiotemporal patterns required for certain tasks.

For example, in the field of video analysis a machine learning model can be trained to recognize objects and actions, within each frame using Convolutional Neural Networks (CNNs). Additionally, it can also grasp the progression of events, over frames by utilizing Gated Recurrent Units (GRUs).

Addressing the Issue of Overfitting; Sometimes CNN GRU models can have a number of parameters, which can lead to overfitting particularly when the training data is limited. To mitigate this problem certain techniques can be employed during training to reduce overfitting and improve generalization. These techniques include dropout, batch normalization and regularization.

Applications, in Fields; CNN GRU architectures find utility in domains such as:

- a. Video Analysis; This includes tasks like activity detection, action recognition and video captioning.[38]
- b. Natural Language Processing (NLP); Sentiment analysis, text generation and machine translation involving word or character sequences are some examples.[39]
- c. Speech and Audio Processing; Spectrogram based music composition, emotion analysis and speech recognition.[40].

Challenges in Training; Training CNN GRU models can be computationally intensive especially when dealing with multidimensional input data. Achieving performance while keeping costs manageable requires striking a balance between the depth and breadth of the CNN and GRU components.

Benefits of Pre trained Models; Leveraging transfer learning and pre training on substantial datasets can be advantageous, for CNN GRU architectures.

The model has the potential to learn information by training the CNN component on a large dataset of images, such, as ImageNet. Then the GRU component can be fine-tuned based on the task. Dataset at hand. The effectiveness of CNN GRU models relies on factors like the nature of the data complexity of the task and availability of training data. Just like any network design making informed architectural decisions and optimizing hyperparameters are crucial for achieving optimal performance in a given task. In order to enhance their performance, for tasks researchers and practitioners often experiment with CNN GRU configurations and incorporate cutting edge methodologies.

3.1.3 CNN Designs

Convolutional neural networks (CNNs) a type of learning architecture is specifically designed to process and interpret input, like images and videos. CNNs have revolutionized the field of computer vision enabling tasks such as segmentation, object recognition and image classification.[41].

The key components of a CNN include:

- a. Convolutional Layers; These layers form the foundation of CNNs. By applying filters or kernels to the input image convolutional layers perform element multiplications and summations. This process creates feature maps that capture patterns, textures, edges and shapes in the image.
- b. Activation Function; After the convolution step feature maps undergo an activation function. The used activation function, in CNNs is Rectified Linear Units (ReLU). ReLU introduces non linearity to enable the model to learn patterns effectively and expressively.
- c. Pooling Layers; Pooling layers are employed to reduce the dimensions of feature maps while retaining information.

In the pooling operation max pooling we select the value within a local region. This helps reduce the resolution and make the computation simpler. Pooling allows the model to become more resistant, to shifts or changes in the input image.

connected layers, also known as thick layers come after multiple convolutional and pooling layers. These layers. Combine the feature maps. They play a role, in making classification or regression decisions by processing all the data from the feature maps. A loss function is a tool that helps compare the output of the layer to the real-world data labels. In image classification applications we commonly use cross entropy, for class classification and binary cross entropy for binary classification.

During the training of Convolutional Neural Networks (CNNs) we utilize Stochastic Gradient Descent (SGD) or its derivatives like Adam to minimize the loss function and update the networks weights and biases.[42].

CNNs leverage backpropagation to compute gradients of the loss function with respect to model parameters. Throughout training these gradients are used to adjust the weights and biases in a way that reduces the loss.

CNNs excel at identifying patterns utilizing local correlations within images and learning hierarchical features from raw pixel data. Their ability to learn hierarchies makes them well suited for computer vision applications. Moreover, transfer learning allows us to bootstrap training, on tasks even if there is data by leveraging pre trained CNN models trained on large image datasets. In fields where information is presented in a manner and where the arrangement of elements still holds significance, such, as in natural language processing and speech recognition (for instance when using 1D convolutions to analyze text sequences) convolutional neural networks (CNNs) have proven to be useful beyond their application in computer vision.

Here are some additional key concepts and variations within networks (CNNs), Stride; During the process of convolution, the filter or kernel moves across the input image by taking steps of a specific size called stride. When the stride value is greater than 1 it skips locations, during convolution resulting in smaller feature maps. Generally smaller strides produce output feature maps with spatial resolution. Padding; Padding is often employed to preserve the dimensions of feature maps and prevent them from shrinking during convolutions. By adding pixels around the input image padding allows the filter to slide over its edges and include boundary information. The terms "padding and "valid" padding are commonly used to describe types of padding techniques.

Dilated Convolution, this type of convolution introduces gaps, within the filters also known as convolution. It effectively captures both global information in the input by expanding the area without adding extra parameters.[43].

Depth wise Separable Convolution (DSC); DSC splits the convolution into two operations; depth wise (filter wise convolution) and pointwise (one to one convolution). This approach retains most of the power while reducing costs and parameter count.

Global Average Pooling; of connected layers certain CNN architectures employ global average pooling, which can be computationally expensive and lead to overfitting. By calculating the value of each feature map this technique reduces parameter count. Enhances the model's robustness to spatial translations. The resulting vector is then used directly for classification.

Skip connections allow gradients to pass through a CNN layer without being altered. They are commonly referred to as connections. This method is frequently used in networks, like ResNet to address the issue of vanishing gradients and facilitate learning of useful representations.

Designs and Variants; Over time CNN designs have. Improved in ways.

There is a known architecture, in the field.

- a. LeNet; This architecture played a role in recognizing digits using CNN.[44]
- b. AlexNet; It gained popularity. Won the 2012 ImageNet competition contributing to the adoption of CNNs.[45]
- c. VGG (Visual Geometry Group); Known for its design and utilization of 3x3 filters, which highlighted the importance of depth in CNNs.[46]
- d. GoogleLeNet (Inception); Introduced the concept of inception modules with filter sizes to enhance learning effectiveness.
- e. ResNet (Residual Network); Not introduced connections but also enabled training of very deep networks like ResNet 50 and ResNet 101.
- f. Tuning hyperparameters; When designing a CNN, it's common to experiment with factors like layer count, filter size, learning rate and dropout rate to achieve results for tasks and datasets.
- g. Methods, these methods combine CNN models to make predictions enhancing model versatility and robustness through techniques, like model averaging and stacking.

Apart, from classifying images Convolutional Neural Networks (CNNs) have proven to be successful in tasks. These include segmenting meaning from images recognizing objects, generating images and even applications beyond computer vision, like language processing and drug discovery. Researchers continuously explore architectures and approaches to enhance the performance, interpretability and adaptability of CNN models as the field of deep learning progresses.

3.1.4 Dataset

The data utilized in this study was obtained from the Kaggle website, which provides a comprehensive collection of authentic photos depicting people affected by the Corona virus sickness. The whole quantity of photos amounted to 2383. The photos were partitioned into three separate files. The training dataset consisted of 1697 samples, whereas the testing and validation datasets had 343 samples each.

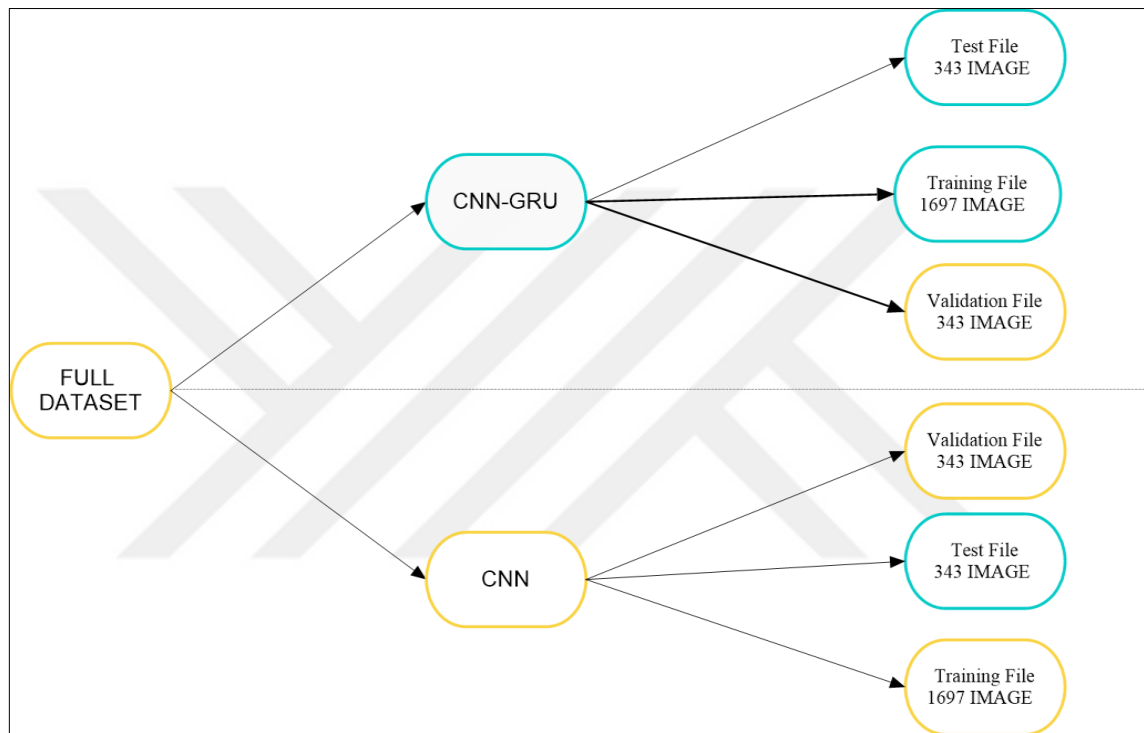


Figure 3.2: Visualize the Collected Data.

3.1.5 Image Data Generator

The Keras ImageDataGenerator can be used to improve the image data, for training, validation and testing purposes. Data augmentation is an adopted technique in learning, where various random transformations are applied to images to expand the effective size of the training dataset. This approach helps enhance the model's ability to generalize.[7]

Here's an analysis of the parameters used in the ImageDataGenerator;

- The featurewise_center parameter is set to False indicating that the data isn't centered by subtracting the mean across the dataset.

- b. The `samplewise_center` parameter is also set to `False` meaning there's no individual centering performed by subtracting each samples mean.
- c. The `featurewise_std_normalization` parameter is set to `False` indicating that normalization isn't done by dividing the data with its deviation across the dataset.
- d. Similarly setting `samplewise_std_normalization` as `False` means there's no normalization performed on each sample by dividing it with its deviation.
- e. The `zca_whitening` parameter is set to `False` well which means ZCA whitening (a technique used for dimensionality reduction) isn't applied to adjust the images.
- f. Lastly `rotation_range` determines the maximum degree limit, for rotation transformations applied to images.

In this situation we can consider rotating the images within a range of 90 to 90 degrees.

The `zoom_range` parameter determines how much random zooming can be applied to the images. A value of 0.5 means that the images can be zoomed in or out with a magnification or reduction of 50%.

The "`width_shift_range`" parameter controls how much random horizontal shifting can be done to the images represented as a fraction of their width. In this case we can horizontally shift the images by, up to 20% of their width.

For displacement the "`height_shift_range`" parameter determines how much shifting can occur vertically expressed as a fraction of the height. In this scenario we are allowed to displace the images by up to 20% of their heights.

Lastly by setting "`horizontal_flip`" to `True` it means that there is a possibility, for flipping of the images.

Once you have set up the data augmentation parameters you can use the) method of the Image Data Generator, on the training, validation and test datasets (`x_train`, `x_val` and `x_test`) to apply data augmentation. It's important to note that data augmentation should only be applied to the training data to avoid overfitting. It is not recommended to use data augmentation on the validation and test sets.

By using the Image Data Generator you can generate augmented versions of your training data in time while training your model. This approach helps improve the model's ability to generalize well with unseen data. During each epoch the model is exposed to modified versions of the training images, which adds variety to the dataset.

Here are some additional factors and techniques related to data augmentation using Keras Image Data Generator module.

Rescaling: This technique is commonly used for preprocessing data. The `rescale` parameter, in the Image Data Generator function allows you to normalize values within a range like `[0, 1]` or `[ -1 1]`. Rescaling is usually performed as a step before applying data augmentation techniques.

To ensure learning, by the model it is necessary to adjust the values to a more suitable range. In Keras the Image Data Generator allows for enhancing image attributes like brightness, contrast and saturation. This can be achieved by tweaking parameters such as `brightness_range`, `contrast_range` and `saturation_range`. By implementing this augmentation technique, we can improve the model's ability to handle variations in lighting conditions.

Data normalization and standardization are steps in data preprocessing. Although the `featurewise_std_normalization` and `samplewise_std_normalization` parameters are currently disabled in ImageDataGenerator it is recommended to include normalization or standardization as steps, in your pipeline. Doing so can be particularly beneficial when working with trained models that require input data to be normalized based on specific criteria.

The Image Data Generator has a feature that allows it to directly read images from directories using the `flow_from_directory` method of using `fit()` with NumPy arrays. This can be useful when dealing with datasets that exceed the computer's memory capacity.

If the built in augmentation options are not sufficient, for a task custom augmentation function can be created. These functions can then be used by specifying the `preprocessing_function` parameter in the Image Data Generator allowing them to be applied to the images during training.

To apply augmentation strategies to subsets of training data multiple instances of the Image Data Generator can be generated, each customized with its own set of augmentation parameters.

Validation data augmentation involves applying data augmentation techniques to the validation set, which is typically used to evaluate a trained model's performance. While data augmentation is usually applied to training data it is worth considering applying augmentation techniques to the validation set as well. This helps make the validation data

more representative, by incorporating real world variations. However, it is important to exercise caution and avoid introducing variation[48].

It is crucial to consider how data augmentation may impact the task at hand.

In the field of imaging there are improvements that can potentially introduce artificial errors, which may affect the ability of the model to accurately interpret real world data.

Data augmentation is a technique that can improve the performance and adaptability of neural network (CNN) models. However, it's crucial to strike a balance, between enhancing the data and preserving the authenticity and coherence of the information. By experimenting with augmentation strategies and evaluating how well the model performs on validation data we can determine the augmentation pipeline, for our specific task.

3.2 CLASSIFICATION REPORT

Scikit learn is a Python toolkit, for machine learning. It provides a method called `classification_report` which generates a report of classification metrics based on predicted labels and true labels.[49].

To use this method, you need to pass two arrays. `labels` and `y_pred` (predicted labels). With matching lengths. The `classification_report` function calculates metrics for each class in the classification problem including accuracy (the ratio of predictions to all predicted positives) precision (the ratio of correct positive predictions to all other positive predictions) F1 score (a combination of recall and accuracy) support (the number of samples, in each class) and overall accuracy (the percentage of correctly classified samples).

The report also provides macro micro average values, which offer different ways to aggregate the metrics across all classes.

The `classification_report` is a tool, for evaluating the performance of a classification model. It provides an overview of how the model is performing for each class and across the entire dataset. This report can help identify classes where the model struggles to classify allowing us to focus on improving its performance in those areas.[50].

Here are some additional details about the classification report:

- a. It is built into scikit learns metrics module.
- b. It is commonly used in supervised machine learning tasks, where the objectives to predict class labels for input instances.

- c. It can handle classification problems with than two classes that need to be predicted.
- d. It can also handle classification scenarios with two classes to anticipate.
- e. The summary of classification metrics generated by the classification report function can be Saved in a file for reference.
- f. The metrics in the report are calculated independently for each class. Then averaged to provide a measure of the model's success.
- g. For a assessment of a models performance the classification report can be combined with other evaluation metrics, like confusion matrix.
- h. It is essential to choose evaluation methods that're suitable, for the problem at hand. For example, if the distribution of classes is uneven or if there are costs associated with positives or false negatives accuracy may not be the appropriate metric to use. In situations considering accuracy along, with recall and F1 score can provide informative insights.

3.2.1 Sparse_Categorical_Crossentropy

In situations where we have classes and the target variable is represented as integers (such, as 0, 1 2) than using one hot encoded vector we utilize the `sparse_categorical_crossentropy` loss function. This function calculates the difference between the class labels and the expected probabilities [51].

Here's how it works:

- a. The model first generates a probability distribution for each input indicating the likelihood of belonging to each class.
- b. The `sparse_categorical_crossentropy` loss function then computes the sum of expected probabilities for each samples class using their logarithms.
- c. This loss is then averaged across all samples in a batch to provide a value that can be used by the optimizer to adjust the neural networks weights.

One of the advantages of using `sparse_categorical_crossentropy` is that it doesn't require one encoding, for target variables. This saves memory and processing resources especially when

dealing with a number of classes. Instead, it directly accepts integer values corresponding to class labels.

It's important to note that `sparse_categorical_crossentropy` assumes that integer labels are densely and consistently encoded starting from 0. If your integer labels are sparsely or inconsistently encoded another loss function may be more appropriate.

Here is some additional information, about the loss function called `sparse_categorical_crossentropy`;

- a. Activation for output; When using `sparse_categorical_crossentropy` on the output layer of a network it is common to pair it with a Softmax activation function. This allows the loss function to calculate cross entropy by converting the models output into a probability distribution across classes.[52].
- b. Integer labels; The `sparse_categorical_crossentropy` algorithm assumes that the target variable is represented as a tensor of integers, where each integer represents a class label. In contrast `categorical_crossentropy` assumes that the target variable is encoded in a tensor using one encoding.
- c. Stability: There can be issues with stability when dealing with extremely small or large probabilities in `sparse_categorical_crossentropy` and other cross entropy loss functions [60]. To address this it's practice to take the logarithm and then add a small epsilon value to the estimated probability.
- d. Gradient calculation: Automatic differentiation methods such as backpropagation can be used to calculate gradients of the loss function with respect, to the network weights.

For problems involving class classification where the target variable is represented as integers, `sparse_categorical_crossentropy` serves as a useful loss function.

Many popular deep learning frameworks, like TensorFlow and Keras provide support, for this feature, which is widely utilized in deep learning applications.

3.3 SOFTWARE FRAMEWORK

It seems like you're referring to "Google Colab," which's a used cloud-based platform provided by Google for running Jupyter notebooks. Google Colab, also known as "Colaboratory " allows users to create and execute Python code within a web-based interface eliminating the need, for installations.

Here are some notable features and benefits of Google Collab:

- a. Free access to GPUs (Graphics Processing Units). Tpus (Tensor Processing Units); These hardware accelerators significantly speed up the training process of learning models making it convenient for researchers and students without access to high performance GPUs.
- b. Integration with Jupyter Notebook; Users can easily create, modify and execute code cells using a user interface. They can also include text using Markdown cells making it a versatile tool for programming and data analysis.
- c. Cloud based technology with collaboration features; Since its cloud-based users can access their notebooks from any device, with an internet connection.

This feature allows you to collaborate with your peers by sharing notebook links or working on the notebook simultaneously.

Google Colab comes with popular Python libraries like TensorFlow, PyTorch, NumPy and Pandas already installed. This means you can start coding without worrying about installing dependencies.

With Colabs integration, with Google Drive you can store your notebooks and data persistently. This feature lets you save and retrieve your work across sessions.

Colab supports widgets making it easy to create user interface components and visualize data dynamically.

You can effortlessly upload datasets from your machine. Retrieve files, from Google Drive or Google Cloud Storage. Colab notebooks support Markdown for text formatting and LaTeX for writing equations. Google Colab provides a variety of code snippets and examples covering subjects and machine learning tasks. This makes it convenient for users to start their projects quickly [53]. To get started with Google Colab simply visit the website at <https://colab.research.google.com>.

When users arrive, they need to verify their identity by logging in with their Google accounts. After that users can create notebooks or import existing ones start coding activities and utilize the platforms features for tasks, like data analysis, machine learning and more.

4. RESULTS

4.1 MODEL DESIGN

In this study we utilized two learning models, the CNN model and the CNN GRU model.

4.1.1 CNN GRU Model

The code snippet provided by the user describes a model using the Keras library, in Python. This particular architecture combines a Convolutional Neural Network (CNN) with a Recurrent Neural Network (RNN) to handle data processing. [54] The main purpose of this model is to process time distributed data, sequences of images used as input.

To process data in 2D image form it is common to use TimeDistributed Conv2D and MaxPooling2D layers. In the layer TimeDistributed is used along with Conv2D to apply 100 filters, each with dimensions of (3, 3) to every frame, within the input. The activation function employed here is Rectified Linear Unit (ReLU). Following that a TimeDistributed MaxPooling2D layer is applied independently on each frame using a pool size of (2, 2). We will now discuss the usage of Second TimeDistributed Conv2D and MaxPooling2D Layers. A new layer called TimeDistributed Conv2D is added, which has 32 filters sized (3, 3) and uses the Rectified Linear Unit (ReLU) activation function. Afterwards a TimeDistributed MaxPooling2D layer is applied with a pool size of (2, 2) again. The TimeDistributed Flatten Layer is a component used in learning models that works on a sequence of input tensors and converts each tensor, into a one format.

We use the TimeDistributed Flatten layer to convert the output produced by the layers into a one vector for each frame. The main focus here is on restructuring and utilizing Reshape and RNN layers within networks. The Reshape layer plays a role, in converting the three outputs from the TimeDistributed Flatten layer into a two-dimensional format specifically with dimensions of (batch_size, time_steps * feature_dim).

To regulate the data, we implement a dropout rate of 0.2 as a regularization technique. To process information from sequences we employ a Bidirectional GRU (Gated Recurrent Unit) layer consisting of 100 units. Bidirectional recurrent neural networks (RNNs) are built to analyze data by taking into account information from both the backward directions. This allows for capturing relationships, in both directions. The dense output layer is an element of a network architecture that exhibits strong connectivity among its neurons. To generate

the result we employ a layer, with two units and utilize the sigmoid activation function. The current model has been specifically designed to handle tasks related to classification.

```

1  model = Sequential()
2  model.add(TimeDistributed (Conv2D (180, (3,3), activation='relu'),
   input_shape=(1,158,150,3)))
3  model.add(TimeDistributed (MaxPooling 2D (pool_size=(2,2))))
4  model.add(TimeDistributed (Conv2D(32, (3,3), activation='relu'))})
5  model.add(TimeDistributed MaxPooling2D(pool_size=(2,2)))})
6  model.add(TimeDistributed Flatten()))
7  model.add(Reshape((1, -1)))
8  model.add(Dropout (0.2))
9  model.add(Bidirectional (GRU(180)))
10 model.add(Dense (2,activation="sigmoid"))
11 model.summary()

```

Figure 4.1: Layers of A Network Architecture.

The summary results might differ based on the shape and batch size of the input. The number of parameters displayed here indicates the parameters, within the model.

```

Model: "sequential"

```

Layer (type)	Output Shape	Param #
time_distributed (TimeDistributed)	(None, 1, 148, 148, 100)	2800
time_distributed_1 (TimeDistributed)	(None, 1, 74, 74, 100)	0
time_distributed_2 (TimeDistributed)	(None, 1, 72, 72, 32)	28832
time_distributed_3 (TimeDistributed)	(None, 1, 36, 36, 32)	0
time_distributed_4 (TimeDistributed)	(None, 1, 41472)	0
reshape (Reshape)	(None, 1, 41472)	0
dropout (Dropout)	(None, 1, 41472)	0
bidirectional (Bidirectional)	(None, 200)	24944400
dense (Dense)	(None, 2)	402

```

=====
Total params: 24,976,434
Trainable params: 24,976,434
Non-trainable params: 0

```

Figure 4.2: The Summary Results.

The provided code snippet involves compiling the model, with the Adam optimizer and utilizing a learning rate of 0.001. For this study we will be using the 'sparse_categorical_crossentropy' loss function. Evaluating the model's performance based on the 'accuracy' metric during training.

```
opt = Adam(lr=0.001)
model.compile(optimizer = opt, loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

Figure 4.3: Model with the Adam Optimizer.

In this analysis we are going to break down the code.

The Adam optimizer is a used optimization algorithm, in machine learning and deep learning. It is known for its efficiency and ability to converge smoothly.

The Adam optimizer is an algorithm that adapts the learning rate effectively making it particularly useful for training networks. The model adjusts parameters using learning rates based on previous gradients. A value of 0.001 is assigned to the learning rate (lr) which determines how much the models' weights are adjusted during the optimization process. The loss function is a function used in machine learning algorithms to measure the difference between predicted and actual values. It helps evaluate how well the model performs. In this context we use 'sparse_categorical_crossentropy' as our loss function. This specific loss function is commonly used in class classification tasks when target labels are represented as integers in a sparse way. If target labels are represented using one encoding its recommended to use the 'categorical_crossentropy' loss function metrics, Throughout the training process we will assess the model's performance using accuracy, as our metric. Accuracy measures how many samples are correctly classified by the model. After the model is compiled, it is ready, for training, by using the specified optimizer, loss function and evaluation metric. You can use the) function on the model instance to train it with the given data.

```
history = model.fit(x_train,y_train,epochs = 25,validation_data = (x_val, y_val),shuffle=True,batch_size=64)
```

Figure 4.4: History of Model.

Replace the variables `x_train` `y_train`, `x_val` and `y_val` with the corresponding datasets, for training and validation. The `batch_size` parameter determines how many samples are used to update the gradient at each step while `epochs` indicate the number of training iterations performed on the dataset.

It's crucial to preprocess the data and accurately determine the number of classes, for the classification task.

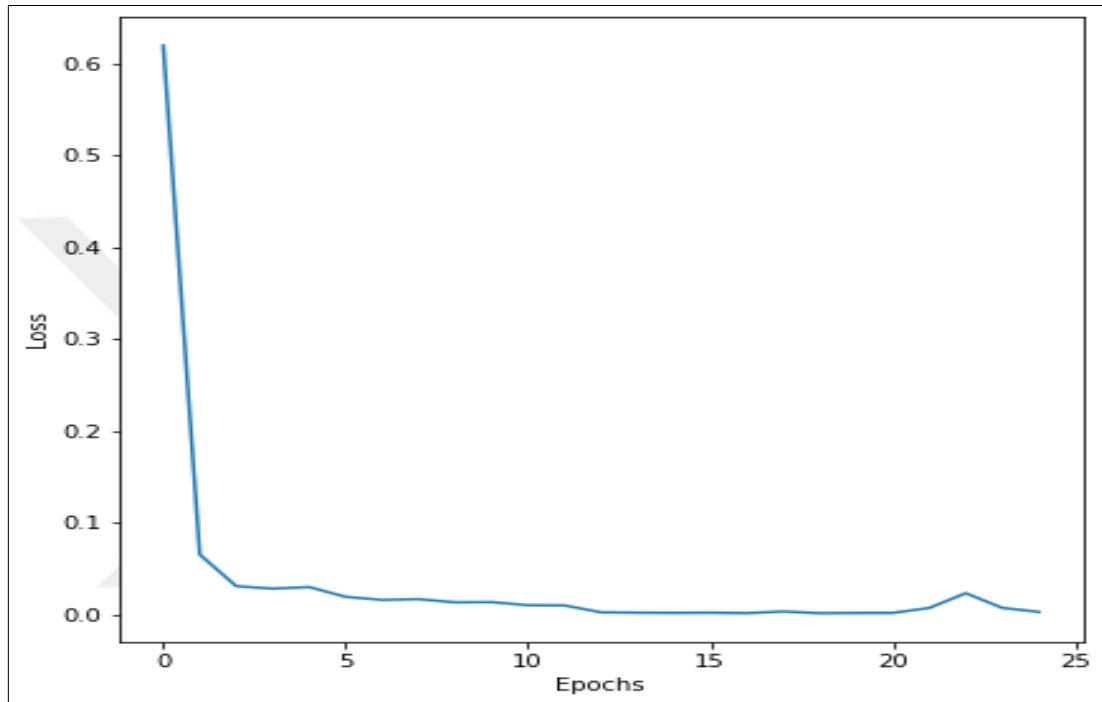


Figure 4.5: Proposed Epochs.

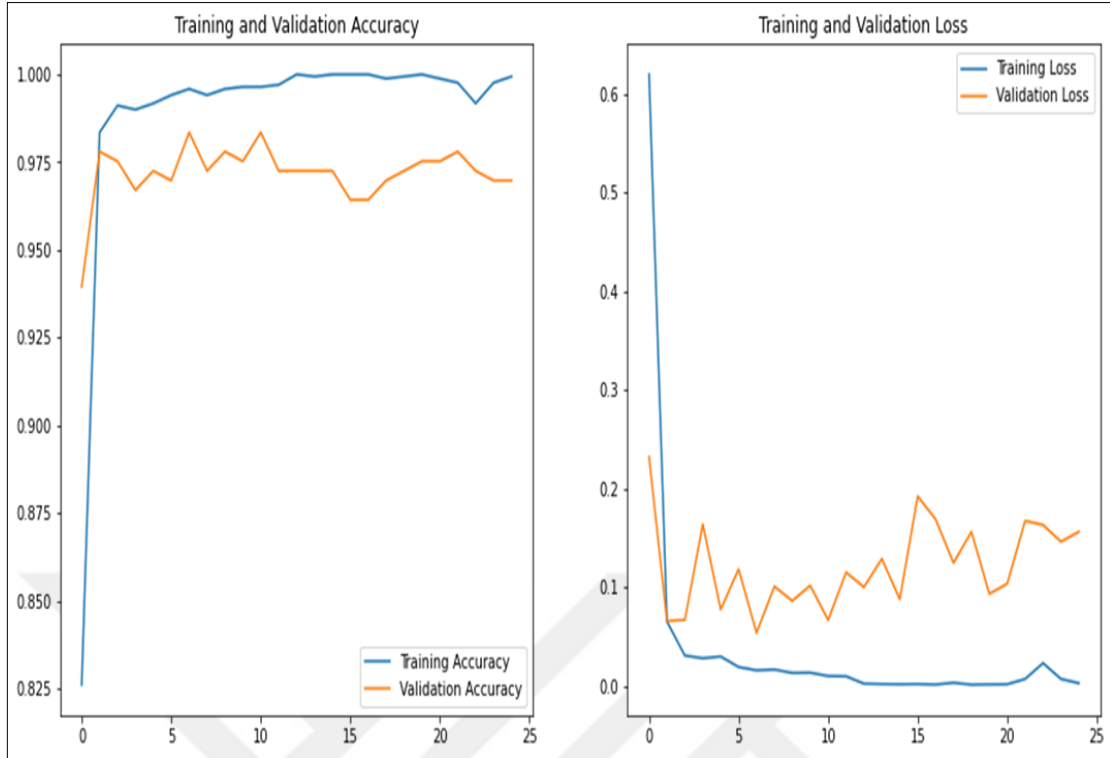


Figure 4.6: Training for Both Validation Acc. and Loss.

The study revealed that throughout 25 epochs the CNN-GRU model accurately forecasted the progression of loss, for activations.

Table 4.1: Best Results Achieved.

Epoch	Time	Validation Loss	Validation Accuracy	Accuracy	Loss
7/25 27/27	172s 6s/step	0.0537	0.9835	0.9959	0.0157

The report, on classification provides metrics for each class included in the target data. Precision, recall and F1 score are some of the metrics that can be used to evaluate the effectiveness of the classification model. In the example we demonstrate how these metrics are utilized in the categorization report. Initially we observed the outcomes, in the CNN GRU algorithm as shown Figure 4.7.

```

predictions = model.predict(x_test)
pred=np.argmax(predictions,axis=1)
pred = pred.reshape(1,-1)[0]
print(classification_report(y_test,pred, target_names = ['normal(Class 0)','covid-19 (Class 1)']))

```

12/12 [=====] - 11s 795ms/step

	precision	recall	f1-score	support
normal(Class 0)	0.98	0.92	0.95	216
covid-19 (Class 1)	0.89	0.97	0.93	147
accuracy			0.94	363
macro avg	0.94	0.95	0.94	363
weighted avg	0.95	0.94	0.94	363

Figure 4.7: Accuracy Classification Report with Test Data.

```

▶ predictions = model.predict(x_val)
pred=np.argmax(predictions,axis=1)
pred = pred.reshape(1,-1)[0]
print(classification_report(y_val,pred, target_names = ['Normal(Class 0)','covid-19 (Class 1)']))

```

12/12 [=====] - 7s 583ms/step

	precision	recall	f1-score	support
Normal(Class 0)	0.95	1.00	0.98	217
covid-19 (Class 1)	1.00	0.93	0.96	147
accuracy			0.97	364
macro avg	0.98	0.96	0.97	364
weighted avg	0.97	0.97	0.97	364

Figure 4.8: Accuracy Classification Report with Validation Data.

4.1.2 CNN Model

The provided code demonstrates the creation of a model, in Keras, which is specifically designed for image classification tasks. Here is a brief overview of the model's architecture. In a network the input layer is responsible for receiving and passing the input data to subsequent layers.

To begin with our model includes an input layer with dimensions (150, 150 3). This means that it expects input images sized at 150x150 pixels and containing three color channels representing green and blue (RGB).

In computer vision applications it is common to employ max pooling layers. These layers play a role in extracting features from input data like images. Convolutional layers are particularly effective, in this regard.

To extract features from the input images a series of four Conv2D and MaxPool2D layers is used. Each Conv2D layer has 32 filters, with dimensions of (3, 3). The activation function applied is ReLU (Rectified Linear Unit). After each Conv2D layer a MaxPool2D layer is used with a pool size of (2, 2) and strides of 2 to reduce the dimensions of the feature maps. To prevent overfitting a Dropout layer with a dropout rate of 0.2 is implemented after the Conv2D layer. The flattening layer in networks transforms input into a one-dimensional output. In this case the Flatten layer converts the two feature maps from the layer into a one-dimensional vector to make it suitable for fully connected layers. Dense layers, also referred to as layers are essential components, in deep learning models.

These layers are made up of neurons.

After the Flatten layer the neural network architecture includes two layers. The first Dense layer has 128 units. Uses the Rectified Linear Unit (ReLU) activation function.

The second Dense layer has 2 units representing the number of classes, in the classification task. It uses the softmax activation function, which's ideal, for class classification.

```

model = Sequential()
model.add(Conv2D(32,(3,3),padding="same", activation="relu", input_shape=(150,150,3)))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Conv2D(32,(3,3), padding="same", activation="relu"))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128,activation="relu"))

model.add(Dense(2, activation="softmax"))

model.summary()

```

Figure 4.9: Layers of A Network Architecture.

The summary gives details, about the number of parameters in each layer and the total count of parameters, in the model. To compile and train the model you can use an optimizer, a loss function, and metrics.

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	896
max_pooling2d (MaxPooling2D)	(None, 75, 75, 32)	0
conv2d_1 (Conv2D)	(None, 75, 75, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 37, 37, 32)	0
conv2d_2 (Conv2D)	(None, 37, 37, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 18, 18, 64)	0
conv2d_3 (Conv2D)	(None, 18, 18, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 9, 9, 64)	0
dropout (Dropout)	(None, 9, 9, 64)	0
flatten (Flatten)	(None, 5184)	0
dense (Dense)	(None, 128)	663680
dense_1 (Dense)	(None, 2)	258
Total params: 729,506		
Trainable params: 729,506		
Non-trainable params: 0		

Figure 4.10: The Summary Results.

```

opt = Adam(lr=0.001)
model.compile(optimizer = opt, loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])

```

Figure 4.11: Model with the Adam Optimizer.

The provided code compiles the model by using the Adam optimizer, which has a learning rate of 0.001. The chosen loss function, for this study is 'sparse_categorical_crossentropy'. Throughout the training process the model's performance will be evaluated based on its accuracy.

To provide clarity about the components.

The Adam optimizer is a widely used optimization algorithm in machine learning and deep learning. It builds upon stochastic descent (SGD) to offer learning rate and momentum.

The Adam optimizer is commonly employed when training networks due to its ability to automatically adjust the learning rate for each parameter during training.

In the code snippet "opt = Adam(lr=0.001)" we initialize the Adam optimizer with a learning rate of 0.001. You can modify this value according to your task and data.

The loss function serves as a measure in machine learning algorithms quantifying how predicted values differ from actual values.

For this study we have chosen 'sparse_categorical_crossentropy' as our loss function. It is suitable, for class classification tasks where target labels are represented as integers instead of one hot encoded value.

When you represent the target labels using one encoding it is advisable to make use of the 'categorical_crossentropy' loss function.

During the training process we will evaluate the performance of the model using a metric called "accuracy." This metric measures how well the model predicts samples correctly compared to the number of samples. After compiling the model, the next step is to train it on the dataset using the `fit` function.

```
history = model.fit(x_train,y_train,epochs = 100,validation_data = (x_val, y_val),shuffle=True,batch_size=64)
```

Figure 4.12: Fitting Model.

The training process, for the compiled model begins with the code snippet by using the `fit` function. In this analysis we will break down the arguments used in the `fit` function.

The `x_train` variable represents the input features of the training data.

The `y_train` variable represents the target values for the training labels.

The "epochs" variable indicates how times the model will iterate through the training dataset. In this case it has been set to 100.

Validation data is provided as a tuple containing both input data (`x_val`) and corresponding target values (`y_val`). This helps evaluate how well the model performs on a validation dataset during training.

If a specific condition is met, shuffling of training data occurs at the beginning of each epoch. Shuffling can be useful to minimize any biases, in the training process.

Batch size refers to how many samples are processed before updating the models' weights during descent.

Throughout the training process the model will go through cycles over the training dataset for a total of 100 epochs. This repetitive procedure entails adjusting the models' weights to minimize the loss function and improve the model's accuracy. By incorporating validation data, we can assess how well the model performs on data during training thereby aiding in identifying any problems, like overfitting or limited generalization capabilities.

Once the training is finished the `fit()` function will provide a history object that contains information, about the training process, including metrics like loss and accuracy, at each epoch. This object can be used to create training and validation curves to analyze how well the model performed.

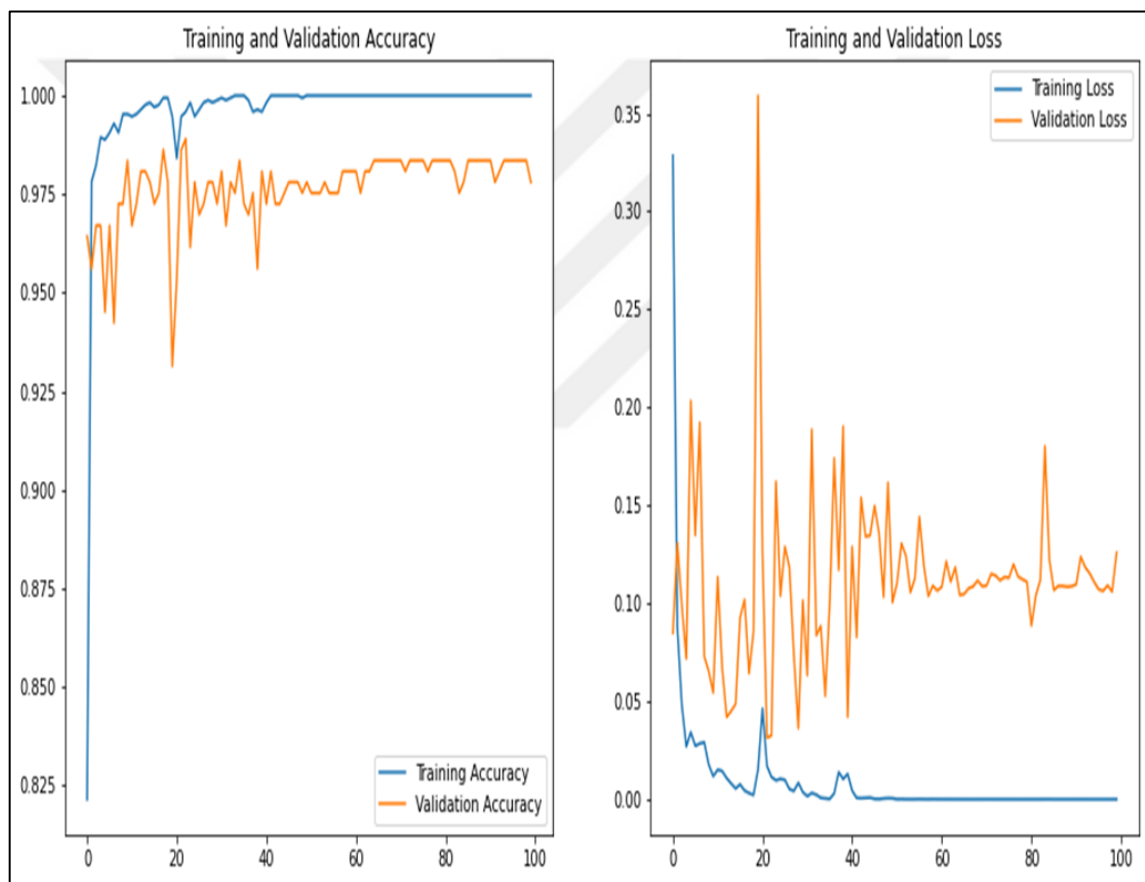


Figure 4.13: Training Val. for Both Acc. and Loss.

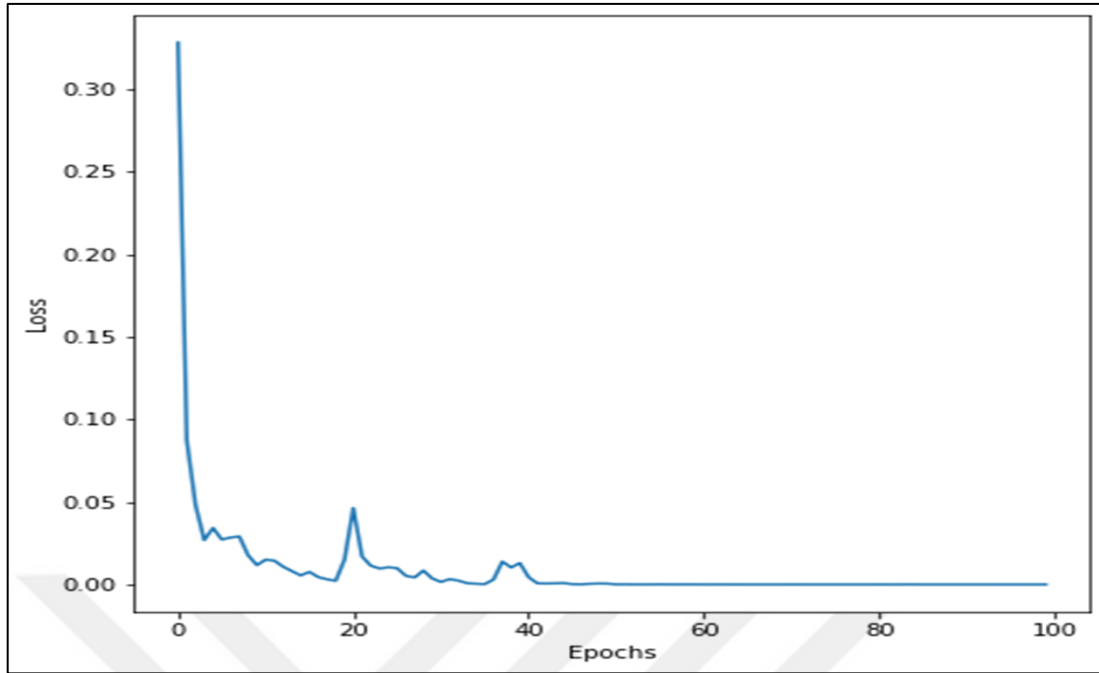


Figure 4.14: Proposed Epochs.

The study demonstrated that throughout 100 epochs the CNN model effectively anticipated the progression of loss, for activations.

Table 4.2: Best Outcome of Val. Accuracy.

Best Result	Time	Validation Loss	Validation Accuracy	Accuracy	Loss
Epoch 10/100 27/27	83s 3s/step	0.0543	0.9835	0.9953	0.0117

The classification report provides measurements, for each class included in the target data. In this context we can consider precision, recall and F1 score as metrics. These indicators help evaluate how effective the classification model is. The following example demonstrates how these elements are used in the categorization report. Initially we observed the results, in the CNN algorithm as follows.

```
[ ] predictions = model.predict(x_test)
    pred=np.argmax(predictions,axis=1)
    pred = pred.reshape(1,-1)[0]
    print(classification_report(y_test,pred, target_names = ['normal(Class 0)','covid-19 (Class 1)']))
```

12/12 [=====] - 5s 369ms/step

	precision	recall	f1-score	support
Normal(Class 0)	0.99	0.96	0.97	216
Stroke (Class 1)	0.95	0.98	0.96	147
accuracy			0.97	363
macro avg	0.97	0.97	0.97	363
weighted avg	0.97	0.97	0.97	363

Figure 4.15: Accuracy Classification Report with Test Data.

```
[ ] predictions = model.predict(x_val)
    pred=np.argmax(predictions,axis=1)
    pred = pred.reshape(1,-1)[0]
    print(classification_report(y_val,pred, target_names = ['Normal(Class 0)','covid-19 (Class 1)']))
```

12/12 [=====] - 5s 414ms/step

	precision	recall	f1-score	support
Normal(Class 0)	0.96	1.00	0.98	217
Stroke (Class 1)	1.00	0.95	0.97	147
accuracy			0.98	364
macro avg	0.98	0.97	0.98	364
weighted avg	0.98	0.98	0.98	364

Figure 4.16: Accuracy Classification Report with Validation Data.

4.2 DISCUSSION

In the discussion the table illustrates differences, in accuracy among the CNN model the CNN GRU model and the dataset employed for testing and validation. It also highlights disparities in training duration. The provided table showcases accuracy levels, for testing, training, and validation stages.

Table 4.3: Discussion the Table Illustrates Differences.

Model	Accuracy With Training Data	Accuracy With Test Data	Accuracy With Validation Data
CNN-GRU	1.00	0.94	0.97
CNN	1.00	0.97	0.98

4.2.1 Comparison of Accuracy with Previous Work

The table clearly shows that both the CNN model and the CNN GRU model outperform studies, in terms of accuracy. Both models have achieved accuracy rates indicating their superiority.

Table 4.4: Comparison of Accuracy with Previous Work.

Works	Classifier	Accuracy
Wang, N., Liu, H.,[10]	CNN	92.4%
Alom,M. Z., [20]	CNN	90.80%
This study	CNN	0.98 %
	CNN-GRU	0.97 %

5. CONCLUSION

5.1 CONTRIBUTION

This study makes a valuable contribution to the current scientific advancements by suggesting alternative models for prediction. The first approach exclusively utilises the Convolutional Neural Network (CNN) algorithm, whereas the second model integrates both CNN layers and Gated Recurrent Unit (GRU) algorithm layers. Moreover, the implementation of the BGR to RGB colour conversion technique demonstrated its efficacy in improving accuracy. Moreover, after verifying the accuracy of the measurement, the obtained results showed positive outcomes, thus improving the practicality of extrapolating the proposed models.

5.2 CONCLUSION

The researchers involved in this study presented two prediction models that could potentially be used to identify patients, with COVID 19. we obtained the dataset for their research from the Kaggle website, which consisted of 2,383 images comparing individuals with and without the disease. These images were then divided into three categories; training, testing and validation.

The results of the study indicated that the first model, which utilized networks (CNN) achieved an impressive accuracy rate of up to 98%. However, when they incorporated GRU into the model its accuracy increased to 97%. The researchers also considered research findings as a step in determining the accuracy of their models.

When evaluating these models, it is recommended to employ performance metrics and rating reports as they offer information on their effectiveness. These metrics include measurements like accuracy, recall and F1 score to provide insights, into how these models can identify various categories of COVID 19 cases.

The high level of accuracy achieved by the models highlights the effectiveness of the methods used in the research. CNNs are commonly chosen for image classification tasks because they excel at extracting features, from images. This makes them a suitable option for tasks involving image classification. On the hand GRU is often used for processing data, such as natural language. However, the authors demonstrate that it does not benefit text classification but also performs well in picture classification tasks.

It is worth noting that relying on a dataset gathered from Kaggle limits the scope of this study. To generalize the model, it would be beneficial to have a dataset, from a hospital or similar source.

To summarize this study developed two prediction models to identify photos of COVID 19 patients. Both CNN and GRU achieved accuracy rates of 98% and 97% respectively when compared to models. Performance metrics and rating reports provided insight into their performance. However additional investigation is needed to evaluate how well these models can be applied to datasets and populations.



REFERENCES

- [1] Pollard, C. A., Morran, M. P., & Nestor-Kalinoski, A. L. (2020). The COVID-19 pandemic: a global health crisis. *Physiological genomics*.
- [2] Hussain, E., Hasan, M., Rahman, M. A., Lee, I., Tamanna, T., & Parvez, M. Z. (2021). CoroDet: A deep learning based classification for COVID-19 detection using chest X-ray images. *Chaos, Solitons & Fractals*, 142, 110495.
- [3] Ozsahin, I., Sekeroglu, B., Musa, M. S., Mustapha, M. T., & Ozsahin, D. U. (2020). Review on diagnosis of COVID-19 from chest CT images using artificial intelligence. *Computational and Mathematical Methods in Medicine*, 2020.
- [4] Gozes, O., Frid-Adar, M., Greenspan, H., Browning, P. D., Zhang, H., Ji, W., ... & Siegel, E. (2020). Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection & patient monitoring using deep learning ct image analysis. *arXiv preprint arXiv:2003.05037*.
- [5] Niu, S., Liu, M., Liu, Y., Wang, J., & Song, H. (2021). Distant domain transfer learning for medical imaging. *IEEE Journal of Biomedical and Health Informatics*, 25(10), 3784-3793.
- [6] Luz, E., Silva, P., Silva, R., Silva, L., Guimarães, J., Miozzo, G., ... & Menotti, D. (2021). Towards an effective and efficient deep learning model for COVID-19 patterns detection in X-ray images. *Research on Biomedical Engineering*, 1-14.
- [7] Bhattacharya, S., Maddikunta, P. K. R., Pham, Q. V., Gadekallu, T. R., Chowdhary, C. L., Alazab, M., & Piran, M. J. (2021). Deep learning and medical image processing for coronavirus (COVID-19) pandemic: A survey. *Sustainable cities and society*, 65, 102589.
- [8] Misra, R., Acharya, S., & Sushmitha, N. (2022). Nanobiosensor-based diagnostic tools in viral infections: Special emphasis on Covid-19. *Reviews in medical virology*, 32(2), e2267.

- [9] Kumar, R., Arora, R., Bansal, V., Sahayasheela, V. J., Buckchash, H., Imran, J., ... & Raman, B. (2022). Classification of COVID-19 from chest x-ray images using deep features and correlation coefficient. *Multimedia Tools and Applications*, 81(19), 27631-27655.
- [10] Wang, N., Liu, H., & Xu, C. (2020, July). Deep learning for the detection of COVID-19 using transfer learning and model integration. In *2020 IEEE 10th international conference on electronics information and emergency communication (ICEIEC)* (pp. 281-284). IEEE.
- [11] Khan, A. I., Shah, J. L., & Bhat, M. M. (2020). CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images. *Computer methods and programs in biomedicine*, 196, 105581.
- [12] Narin, A., Kaya, C., & Pamuk, Z. (2021). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, 24, 1207-1220.
- [13] Agarwal, S., & Chowdary, C. R. (2021). Combating hate speech using an adaptive ensemble learning model with a case study on COVID-19. *Expert Systems with Applications*, 185, 115632.
- [14] Ibrahim, A. U., Ozsoz, M., Serte, S., Al-Turjman, F., & Yakoi, P. S. (2021). Pneumonia classification using deep learning from chest X-ray images during COVID-19. *Cognitive Computation*, 1-13.
- [15] Yan, L., Zhang, H. T., Xiao, Y., Wang, M., Sun, C., Liang, J., ... & Tang, X. (2020). Prediction of survival for severe Covid-19 patients with three clinical features: development of a machine learning-based prognostic model with clinical data in Wuhan. *MedRxiv*.
- [16] Rajeshwari, M. R., & Kavitha, K. S. (2022). Enhanced tolerance-based intuitionistic fuzzy rough set theory feature selection and ResNet-18 feature extraction model for arrhythmia classification. *Multiagent and Grid Systems*, 18(3-4), 241-261.

- [17] Hemdan, E. E. D., Shouman, M. A., & Karar, M. E. (2020). Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. *arXiv preprint arXiv:2003.11055*.
- [18] Ahuja, S., Panigrahi, B. K., Dey, N., Rajinikanth, V., & Gandhi, T. K. (2021). Deep transfer learning-based automated detection of COVID-19 from lung CT scan slices. *Applied Intelligence*, 51, 571-585.
- [19] Horry, M. J., Chakraborty, S., Paul, M., Ulhaq, A., Pradhan, B., Saha, M., & Shukla, N. (2020). COVID-19 detection through transfer learning using multimodal imaging data. *Ieee Access*, 8, 149808-149824.
- [20] Alom, M. Z., Rahman, M. M., Nasrin, M. S., Taha, T. M., & Asari, V. K. (2020). COVID_MTNNet: COVID-19 detection with multi-task deep learning approaches. *arXiv preprint arXiv:2004.03747*.
- [21] Jaiswal, A. K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., & Rodrigues, J. J. (2019). Identifying pneumonia in chest X-rays: A deep learning approach. *Measurement*, 145, 511-518.
- [22] Agarwal, R., & Gopinath, M. G. (2021). *A proposal to end the COVID-19 pandemic*. International Monetary Fund.
- [23] Ryan, D. (2016). *Understanding digital marketing: marketing strategies for engaging the digital generation*. Kogan Page Publishers.
- [24] Pathan, M., Buyya, R., & Vakali, A. (2008). Content delivery networks: State of the art, insights, and imperatives. *Content Delivery Networks*, 3-32.
- [25] Omara, E., Mosa, M., & Ismail, N. (2022). Applying Recurrent Networks For Arabic Sentiment Analysis. *Menoufia Journal of Electronic Engineering Research*, 31(1), 21-28.
- [26] Omara, E., Mousa, M., & Ismail, N. (2022). Character gated recurrent neural networks for Arabic sentiment analysis. *Scientific Reports*, 12(1), 9779.

- [26] Yang, R., Feng, L., Wang, H., Yao, J., & Luo, S. (2020). Parallel recurrent convolutional neural networks-based music genre classification method for mobile devices. *IEEE Access*, 8, 19629-19637.
- [27] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.
- [28] Reza, S., Ferreira, M. C., Machado, J. J., & Tavares, J. M. R. (2022). Traffic state prediction using one-dimensional convolution neural networks and long short-term memory. *Applied Sciences*, 12(10), 5149.
- [29] Graves, A., Fernández, S., & Schmidhuber, J. (2007, September). Multi-dimensional recurrent neural networks. In *International conference on artificial neural networks* (pp. 549-558). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [30] Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1), 221-231.
- [31] Pesah, A., Cerezo, M., Wang, S., Volkoff, T., Sornborger, A. T., & Coles, P. J. (2021). Absence of barren plateaus in quantum convolutional neural networks. *Physical Review X*, 11(4), 041011.
- [32] Zheng, Z., Chen, Z., Hu, F., Zhu, J., Tang, Q., & Liang, Y. (2020). An automatic diagnosis of arrhythmias using a combination of CNN and LSTM technology. *Electronics*, 9(1), 121.
- [33] Chawla, A., Lee, B., Fallon, S., & Jacob, P. (2019). Host based intrusion detection system with combined CNN/RNN model. In *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18* (pp. 149-158). Springer International Publishing.
- [34] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

- [35] Xie, Y., Xing, F., Kong, X., Su, H., & Yang, L. (2015). Beyond classification: structured regression for robust cell detection using convolutional neural network. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18* (pp. 358-365). Springer International Publishing.
- [36] Kwak, G. H., Park, C. W., Lee, K. D., Na, S. I., Ahn, H. Y., & Park, N. W. (2021). Potential of hybrid CNN-RF model for early crop mapping with limited input data. *Remote Sensing*, 13(9), 1629.
- [37] van Amsterdam, B., Clarkson, M. J., & Stoyanov, D. (2021). Gesture recognition in robotic surgery: a review. *IEEE Transactions on Biomedical Engineering*, 68(6).
- [38] Muhammad, K., Ullah, A., Imran, A. S., Sajjad, M., Kiran, M. S., Sannino, G., & de Albuquerque, V. H. C. (2021). Human action recognition using attention based LSTM network with dilated CNN features. *Future Generation Computer Systems*, 125, 820-830.
- [39] Mathew, L., & Bindu, V. R. (2020, March). A review of natural language processing techniques for sentiment analysis using pre-trained models. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 340-345). IEEE.
- [40] Korvel, G., Treigys, P., Tamulevicius, G., Bernataviciene, J., & Kostek, B. (2018). Analysis of 2d feature spaces for deep learning-based speech recognition. *Journal of the Audio Engineering Society*, 66(12), 1072-1081.
- [41] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, 8, 1-74.
- [42] Teuwen, J., & Moriakov, N. (2020). Convolutional neural networks. In *Handbook of medical image computing and computer assisted intervention* (pp. 481-501). Academic Press.

- [43] Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., & Cottrell, G. (2018, March). Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)* (pp. 1451-1460). Ieee.
- [44] Kuo, C. C. J. (2016). Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41, 406-413.
- [45] Özyurt, F. (2020). A fused CNN model for WBC detection with MRMR feature selection and extreme learning machine. *Soft Computing*, 24(11), 8163-8172.
- [46] Paymode, A. S., & Malode, V. B. (2022). Transfer learning for multi-crop leaf disease image classification using convolutional neural network VGG. *Artificial Intelligence in Agriculture*, 6, 23-33.
- [47] Chlap, P., Min, H., Vandenberg, N., Dowling, J., Holloway, L., & Haworth, A. (2021). A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology*, 65(5), 545-563.
- [48] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T. Y., Shlens, J., & Le, Q. V. (2020). Learning data augmentation strategies for object detection. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16* (pp. 566-583). Springer International Publishing.
- [49] Bora, J., Dehingia, S., Boruah, A., Chetia, A. A., & Gogoi, D. (2023). Real-time assamese sign language recognition using mediapipe and deep learning. *Procedia Computer Science*, 218, 1384-1393.
- [50] Namani, S., Akkapeddi, L. S., & Bantu, S. (2022, March). Performance Analysis of VGG-19 Deep Learning Model for COVID-19 Detection. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 781-787). IEEE.
- [51] Berrabia, N., Fekhari, A., & Hentabli, H. (2021). *Novel Attention Deep Learning Model for Cross Language Abstractive Text Summarization* (Doctoral dissertation).
- [52] Qin, J., Yuan, S., Chen, J., Amor, B. B., Fang, Y., Hoang-Xuan, N., ... & Wang, H. (2022). SHREC'22 track: Sketch-based 3D shape retrieval in the wild. *Computers & Graphics*, 107, 104-115.

- [53] Hasan, A. K. H. (2022). *Characteristics of machine learning algorithms to improve student evaluation* (Master's thesis, Altınbaş Üniversitesi/Lisansüstü Eğitim Enstitüsü).
- [54] Mou, L., Ghamisi, P., & Zhu, X. X. (2017). Deep recurrent neural networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7), 3639-3655.

