Attentive Diagnostics: Enhancing Chest X-Ray Analysis Through Integrated Attention Mechanisms and SVM Classification

Juan David Gomez Villalba
Department of Computer Science
University of London

mjdgv1@student.london.ac.uk
https://github.com/Kayaba-Attribution/Attentive-Diagnostics-UoL-
Deep Learning On A Public Dataset

Final Project For CM3070 Computer Science Final Project

UNIVERSITY
OF LONDON

1. INTRODUCTION

Chest radiography is a fundamental diagnostic tool in medicine, playing a crucial role in detecting various thoracic diseases such as pneumonia, tuberculosis, and lung cancer. The accurate interpretation of chest X-rays (CXRs) is a complex task, often challenged by the subtle nature of imaging findings and the need for high-level expertise [19]. This complexity can lead to human errors and biases, especially in regions where experienced radiologists are scarce. Therefore, developing reliable, automated diagnostic tools is of paramount importance [18].

Recent advances in artificial intelligence (AI), particularly in deep learning, have shown promising results in medical image analysis. Convolutional Neural Networks (CNNs), due to their powerful feature extraction capabilities, have emerged as a key technology in this domain [13]. However, traditional CNN approaches often lack the ability to focus on specific, clinically relevant features in an image, a limitation that can lead to missed diagnoses or misinterpretation.

To address these challenges, our project, "Attentive Diagnostics," introduces an innovative approach that integrates attention mechanisms with CNNs, coupled with Support Vector Machine (SVM) classification. This integration not only aims to enhance the diagnostic accuracy and efficiency in analyzing CXRs but also introduces an AI-driven method to try to minimize human error and bias in medical image interpretation. The attention mechanism within the CNN allows the model to focus on the most relevant parts of an image, mimicking the way a radiologist might examine a radiograph, thus improving the model's interpretability and accuracy .

The use of SVMs as classifiers in this context is driven by their proven effectiveness in high-dimensional spaces and robust classification capabilities [20]. When combined with the rich feature representations extracted by the CNNs [13] and enhanced by the proposed attention mechanisms [21], the SVMs provide a more nuanced and accurate classification of potential pathologies present in CXRs. In validating our

approach, I will utilize comprehensive datasets, principally the NIHs ChestX-ray14[8] Dataset, which comprises approximately 120,000 images across 14 disease labels. This extensive dataset, along with the possible addition of CheXpert[8] and the COVID-19 Radiography Database[10,11], offers a robust platform for training and evaluating our model across a spectrum of clinical scenarios.

"Attentive Diagnostics" embarks on a practical journey in the field of medical diagnostics, propelled by the strategic application of AI. The project is centered around creating novel models by making use of established techniques known for their complementary benefits, evaluating an array of models that leverage both pre-trained weights and those trained from scratch, specifically tailored for CXR data, across diverse architectures, and systematically combining different techniques to critically evaluate the results. Our primary objective is to craft a tool that not only promises enhanced efficiency and precision but is also accessible, especially in under-resourced healthcare settings. By pushing the boundaries of the field, I aim to showcase the potential findings I may unravel.

2. LITERATURE REVIEW

2.1 CNNs Evolution Over Time

The evolution of CNNs over the years has been marked by significant milestones that have continually reshaped the landscape of image recognition and classification. The journey began with LeCun et al [1], 1998 groundbreaking paper, "Gradient-Based Learning Applied to Document Recognition," introducing LeNet-5.

This pioneering network effectively applied CNNs for character recognition, laying the groundwork for future deep learning developments. LeNet-5's architecture, featuring alternating layers of convolutional and subsampling layers, set a precedent for CNN design. Despite its early success, CNN research stagnated in the late 1990s and early 2000s. Seen as complex 'black boxes,' CNNs were less favored compared to handcrafted features and Support Vector Machines (SVMs), partly due to the lack of diverse image datasets and doubts about the backpropagation algorithm's efficiency.

A significant leap occurred in 2012 with Krizhevsky et al [2]. "ImageNet Classification with Deep Convolutional Neural Networks," commonly known as AlexNet. This model, using the ImageNet dataset[6]—a large repository of annotated images—marked a resurgence in neural networks within image classification.

AlexNet's deeper architecture incorporated ReLU (Rectified Linear Unit) activations, which helped mitigate the vanishing gradient problem by introducing non-saturation of gradients. Additionally, its use of dropout for regularization and GPU acceleration significantly improved the model training efficiency. AlexNet's success, underscored by a substantial reduction in error rates on the ImageNet challenge, highlighted the potential of deep CNNs.

In 2014, "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Simonyan and Zisserman introduced VGGNet [3]. VGGNet's architecture, featuring smaller convolutional filters, allowed for building deeper networks, which was key to enhancing learning capacity. The model's deep structure, albeit computationally intensive, became popular for feature extraction in various applications, affirming the trend towards deeper networks.

That same year, "Going Deeper with Convolutions," by Szegedy et al [4]. introduced GoogLeNet or Inception, revolutionizing the approach to network design. The inception module, a key innovation in GoogLeNet, allowed for increasing the depth and width of the network without a significant rise in computational cost. This module operated on a "network within a network" principle, applying different-sized filters to the same input and concatenating the results. This structure enabled efficient learning of representations at various scales, demonstrating a significant improvement in performance while maintaining manageable computational requirements.

In 2015, "Deep Residual Learning for Image Recognition" by He et al [5]. unveiled ResNet, a network that addressed the vanishing gradient problem through the introduction of residual connections. These connections facilitated the flow

of gradients through the network by introducing shortcut connections that skipped one or more layers. This innovation in CNNs, exemplified by ResNet's breakthrough performance in the ImageNet challenge, marked new performance records and underscored a significant shift towards deeper and more complex architectures.

These advancements in feature extraction and classification have paved the way for remarkable improvements in various fields, including medical imaging. However, the evolution of CNNs has also highlighted two main concerns: the high computational cost and memory requirements of deep and wide architectures. Such challenges are particularly pronounced in resource-constrained environments, where deploying state-of-the-art models may not be feasible [7].

This dilemma has led to a growing interest in the use of pre-trained networks, which can significantly improve the training speed and accuracy of new models, especially in specialized applications like chest radiograph classification.

2.2 CNNs for CXR Related To The Project

2.2.1 CNNs Architecture Comparison

In the constantly evolving field of medical imaging, particularly in chest radiograph (CXR) classification, CNNs have been at the forefront of research. A significant contribution to this area is the paper "Comparing Different Deep Learning Architectures for Classification of Chest Radiographs"[13] which conducts an in-depth comparison of 16 CNN architectures.

These architectures, initially trained on the ImageNet dataset, are evaluated based on their performance on two key datasets: CheXpert and the COVID-19 Image Data Collection.

The paper highlights the potential advantages of using pre-trained networks, which can substantially improve training speed and accuracy in medical image classification. One of the notable findings is the effectiveness of shallower networks, such as AlexNet (with eight layers), ResNet-34, and VGG-16. These models, which can be trained more efficiently and rapidly, demonstrate classification results comparable to, and occasionally superior to, more complex models like DenseNet-121 Fig 1, which was previously considered the standard for such tasks.

Network	Cardiomegaly	Effusion	Pooled
VGG-16	0.766	0.883	0.846
VGG-19	0.786	0.884	0.854
ResNet-18	0.822	0.911	0.868
ResNet-34	0.840	0.919	0.872
ResNet-152	0.836	0.937	0.882
DenseNet-121	0.828	0.926	0.869

Fig 1. Area under the Receiver Operating Characteristic Curve Performance metrics for different networks on Cardiomegaly, Effusion, and Pooled

This is particularly relevant in scenarios with limited hardware capabilities. Moreover, the research advocates for a 'human in the loop' approach during model training, where an expert can intervene and refine the model and its parameters. This methodology is crucial for enhancing model accuracy and reliability.

The primary performance metrics used in this study are the Area Under the Receiver Operating Characteristics curve (AUROC) and the Area Under the Precision-Recall Curve (AUPRC). These metrics are crucial as they provide insights into the model's ability to distinguish between different pathologies (AUROC) and its precision and recall balance (AUPRC). On the CheXpert dataset, the smaller models achieved AUROC values ranging from 0.83 to 0.88, which is comparable to the 0.889

AUROC achieved by the original authors of CheXpert using a DenseNet-121[12]. This marginal difference underlines the capability of smaller models in accurate pathology detection. For the COVID-19 Image Data Collection, all models showed excellent performance in detecting COVID-19 and non-COVID pneumonia, with AUROC values between 0.983 and 0.998, reinforcing the efficiency of simpler networks in specific imaging scenarios.

A potential limitation of the study is its reliance on only two publicly available datasets. This restriction may lead to overfitting, possibly affecting the generalizability of the results—a common issue in deep learning research.

Nonetheless, this paper makes a substantial contribution to the field, challenging the notion

that only complex, deep networks are suitable for CXR classification and opening up possibilities for using more accessible and efficient models in medical diagnostics.

2.2.2 *Integration of SVMs Classifiers in CNNs*

Building upon the evolving landscape of CNNs in medical imaging, particularly in chest X-ray (CXR) analysis, the study "Convolutional Neural Networks for the Classification of Chest X-rays in the IoT Era"[20] marks a significant advance in the field.

This paper notably builds on earlier findings that simpler networks like AlexNet, ResNet-34, and VGG-16 can rival or even outperform more complex architectures such as DenseNet-121 (Figure 3). In addressing a critical need in the medical domain, where a shortage of radiologists and the time-intensive nature of X-ray examinations pose significant challenges, this research introduces innovative deep learning methods for the rapid and automated classification of chest X-ray images.

Utilizing the ChestX-ray14 dataset, the study pioneers an approach that integrates Support Vector Machines (SVM) with AlexNet and

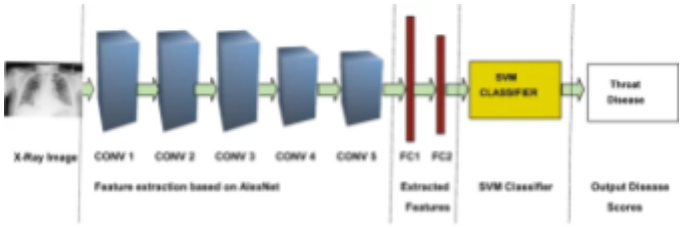

Figure 2. SVM-AlexNet Method Architecture Diagram taken from the original paper [20]

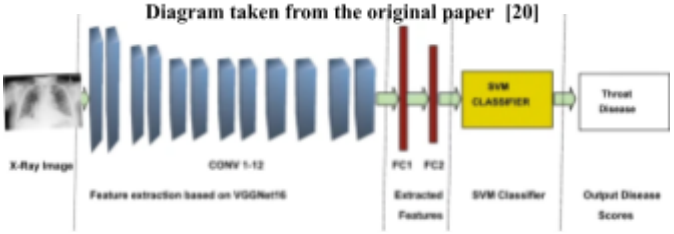

Figure 3. SVM-VGGNet16 Method Architecture Diagram taken from the original paper [20]

VGG-16 (Figures 2 and 3), a departure from the traditional use of Softmax classifiers in CNNs. This fusion of SVMs with more streamlined yet effective networks highlights a promising avenue for enhancing the accuracy and speed of lung disease diagnosis via CXRs.

A standout feature of this research is the remarkable performance metrics of the proposed SVM-AlexNet and SVM-VGGNet16 models on the ChestX-ray14 dataset. These models achieved average Area Under the Curve (AUC) values between 97% to 98%, signifying a high degree of precision in disease classification (Figure 4).

Models	Avg
AlexNet+SVM	0.98
VGG16+SVM	0.97
Wang et al.	0.73
Yao et al.	0.76
Gundel et al.	0.79
ResNet	0.72
ChestNet	0.76
CheXNet	0.83

Fig 4. The performance comparison of the proposed models (AUC) [20]

The integration of SVMs not only preserves the computational efficiency of these networks but also significantly amplifies their classification prowess. The SVM's approach to margin-based loss minimization and regularization plays a crucial role in this enhancement, offering a more robust classification than models based on Softmax.

However, the study acknowledges some limitations. It relies exclusively on the ChestX-ray14 dataset for benchmarking, which may affect the broader applicability of the findings. Furthermore, the computational demands of running and fine-tuning CNN models alongside SVM classifiers could pose challenges in less powerful computing environments, such as mobile devices.

Combining the efficiency of simpler network architectures with the robust classification ability of SVMs, effectively addresses the urgent need for rapid and precise diagnostic tools in medical contexts with limited radiologist availability.

These innovative deep learning methods hold significant promise for revolutionizing CXR analysis, thereby enhancing the accessibility and efficacy of advanced medical diagnostics.

2.2.3 Integration of Attention Mechanisms in CNNs

The paper "Studying the Effects of Self-Attention for Medical Image Analysis"[21] focuses on enhancing medical image classification by integrating self-attention mechanisms into CNNs.

This research, addressing the limitations of standard CNNs in emulating the focused attention of medical specialists, employs various state-of-the-art self-attention models like Squeeze-and-Excitation (SE), Convolutional Block Attention Module (CBAM) Fig 5, 6, and Global Context Network (GCNet).

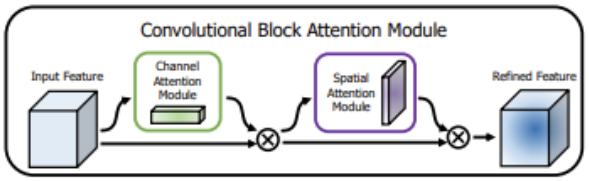

Fig 5. : Overall Convolutional Block Attention Module (CBAM). Diagram taken from the original paper [22].

The module comprises channel and spatial sub-modules, enhancing features within deep network convolutional blocks through a process described by:

$$F' = M_c(F) \otimes F, \quad F'' = M_s(F') \otimes F'$$

Given an intermediate feature $F \in \mathbb{R}^{C \times H \times W}$, it produces a 1D channel attention map $M_c \in \mathbb{R}^{C \times 1 \times 1}$ and a 2D spatial attention map $M_s \in \mathbb{R}^{1 \times H \times W}$, with $\otimes$ denoting element-wise multiplication [23].

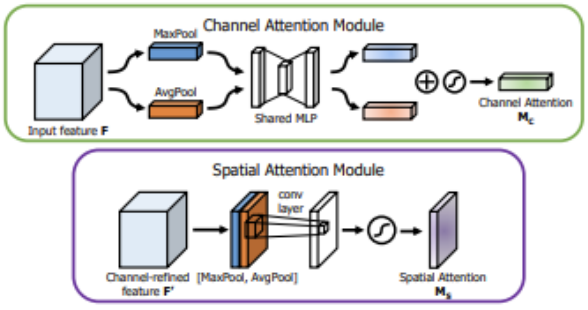

Fig 6. : Channel and spatial attention sub-modules. Diagram taken from the original paper [22].

These models are assessed across various medical imaging datasets, including skin, CXR, MRI, and CT images. The integration of self-attention mechanisms allows CNNs to concentrate on semantically significant regions, thereby potentially increasing the robustness and accuracy of medical image analysis by concentrating on more clinically significant feature regions [21].

The research employed the straightforward ResNet-18 architecture as a base, modifying it to incorporate different attention mechanisms; the research offers a comprehensive comparison of these techniques across a range of medical imaging tasks. These mechanisms were integrated into the residual blocks of the ResNet-18 model. The attention-augmented models were compared against the standard ResNet-18 model architecture Fig 7.

Model	Test 1	Test 2	Test 3	Mean AUC-ROC
ResNet-18 [13]	93.50%	93.77%	92.59%	93.28%
ResNet-18 + SE [15]	95.20%	94.83%	95.15%	95.06% (+1.78)
ResNet-18 + CBAM [34]	95.28%	94.73%	95.26%	95.09% (+1.81)
ResNet-18 + GC [7]	93.32%	93.86%	94.28%	93.82% (+0.54)

Fig 7. Quantitative CXR Dataset Results. Notice ResNet-18+CBAM performance.

The models were evaluated both quantitatively and qualitatively. Quantitative evaluations involved running each model variant across three individual tests and calculating the average area under the curve of the receiver operating characteristic (AUC-ROC), a robust metric for classification models.

Qualitatively, Grad-CAM activation heat-maps were visualized for each model on different dataset image samples Fig 8. These visualizations were then reviewed by medical specialists who provided insights into which models and attention mechanisms focused on the most clinically relevant regions.

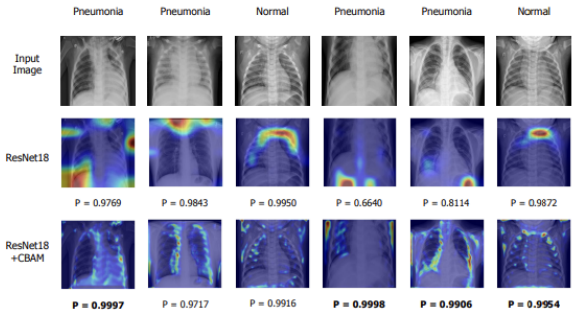

Fig 8. Grad-CAM activation heat-maps for Ground Image, ResNet-18 and ResNet18+CBAM

The study concluded that self-attention mechanisms enabled standard CNN models to focus more on semantically important features, improving AUC-ROC for medical vision task

accuracy on a range of medical images.

However, the research highlighted the need for further validation with other datasets to establish a consistent trend of improved accuracy with attention.

The findings emphasize the potential of self-attention mechanisms to enhance the capability of AI-driven diagnostic tools, paving the way for more advanced applications in the field.

3. DESIGN

3.1 CXRs And Databases Used

Chest X-rays (CXRs) are one of the most common and widely used diagnostic imaging tests in the medical field. They are radiographic images of the chest used to diagnose conditions affecting the chest, its contents, and nearby structures. CXRs play a crucial role in detecting various lung diseases, heart conditions, and chest-related traumas.

In the analysis of chest X-rays (CXR), several significant datasets have emerged and have been used to advance the application of CNNs for diagnostic accuracy and efficiency. Our project will utilize three key datasets, each offering unique contributions to the field of medical imaging.

The ChestX-ray14[8] dataset, released by the National Institutes of Health (NIH), is one of the largest publicly available chest X-ray datasets. It comprises over 112,000 frontal-view X-ray images from more than 30,000 patients, with each image labeled for up to 14 different thoracic pathologies.

This dataset will serve as the primary resource for our project, offering an extensive collection of labeled images crucial for training CNNs to identify and classify a wide spectrum of chest diseases. Its comprehensive nature provides a valuable benchmark for numerous research studies in medical imaging.

In addition to ChestX-ray14[8], if time permits I will utilize the CheXpert dataset, developed by Stanford University. This dataset is tailored for chest radiograph interpretation and consists of 224,316 chest radiographs from 65,240 patients, each annotated for the presence of 14 observations as per radiology reports.

CheXpert is distinguished by its competition mode, which includes hidden test sets designed to challenge the development of models capable of generalizing effectively on unseen data. This dataset will be employed as a supplementary resource to verify results and conduct specific tests, particularly useful for its role in training models to interpret and classify ambiguous findings in medical imaging.

In addition, if there's any spare time, I can make use of the COVID-19 Radiography Database [10,11], developed by researchers at Qatar University, to bolster the development of automated COVID-19 detection tools. This database consists of 33,920 chest X-ray images, categorized into three different classes. Due to its relatively smaller size, it's well-suited for quick iterations and experimentation with various model architectures.

3.2 Objective, Domain and Users

This project aims to develop a novel CNN architecture for CXR classification, capitalizing on advancements in self-attention mechanisms and SVMs for improved diagnostic precision. It addresses the critical need in the domain of medical imaging, especially in resource-limited settings, by offering a solution that combines efficiency with high-level expertise.

Our users, primarily radiologists and healthcare professionals, require reliable tools for quick and accurate diagnosis. The approach involves preparing and analyzing data for deep learning, employing transfer learning and fine-tuning with pretrained models, and exploring various model architectures informed by literature reviews.

The project will further innovate by integrating SVM for classification and incorporating attention mechanisms, evaluated for efficacy. The culmination of this project is to blend the best-performing models from each category, creating a combined approach that utilizes both self-attention and SVM, aiming to set a new benchmark in the accuracy and efficiency of medical image analysis.

This comprehensive strategy is designed to produce a tool that not only accelerates diagnosis but also increases its precision, thereby significantly contributing to the healthcare sector, especially in regions where such advancements are most needed.

3.3 Solutions and Architectures

In designing the architecture for this project, the choice of tools and models is driven by the specific needs of medical imaging and informed by extensive literature review. Utilizing the TensorFlow and Keras frameworks [16] with Python allows for leveraging their robust, flexible capabilities, particularly beneficial for complex tasks like medical image analysis.

The decision to use pre-trained model weights from Keras Applications aligns with the project's objective of efficient and accurate CXR classification. This approach is supported by insights from authoritative sources like Chollet's "Deep Learning With Python"[14] and Geron's "Hands-on Machine Learning with scikit-learn, keras & tensorflow"[15], along with relevant academic coursework.

The project will adopt a structured approach that will make use of the well-labeled nature of medical imaging datasets like ChestX-ray14[8] and CheXpert[9]. Focusing on AlexNet[1], but mainly ResNet[5], and VGG-16[3] with a combination of pre-trained and from the ground up architectures is a strategic choice, aligning with the literature suggesting the potential advantages of pre-trained networks while also not undervaluing training directly from the data in enhancing training speed and accuracy.

The effectiveness of shallower networks, as highlighted in recent studies, further validates this choice, indicating their efficiency and potential to achieve results on par with or better than more complex models like DenseNet-121.

Data preparation will be thorough, involving collection, cleaning, exploration, and preparation of gathered data. Feature extraction and transfer learning, followed by fine-tuning with the prepared data, will be crucial steps. The integration of SVM classifiers, as opposed to traditional softmax classifiers, is based on literature findings demonstrating their superior performance in terms of AUC, especially when combined with shallow networks like AlexNet and VGG-18.

The use of SVMs, known for their margin-based loss minimization and regularization, is expected to provide a more robust classification capability. Finally, the project will explore the integration of state-of-the-art attention modules in CNNs, aiming to emulate the focused attention of medical specialists.

The Convolutional Block Attention Module (CBAM) will be employed based on its demonstrated effectiveness in recent studies. This integration seeks to enhance the model's capability to highlight clinically relevant features in CXRs, aligning with the project's goal of supporting medical professionals in accurate and efficient diagnostics.

3.4 Performance Metrics and Evaluation

The evaluation strategy for this project will be comprehensive and aligned with the objective of achieving high diagnostic accuracy in CXR classification. To ensure the developed models meet the required standards, I will employ a set of performance metrics, primarily focusing on the Area Under the Curve of Receiver Operating Characteristics (AUC-ROC) alongside loss, accuracy and precision-recall curves.

These metrics are critical as AUC-ROC provides a measure of the model's ability to distinguish between classes (e.g., diseased vs. healthy), essential in medical diagnostics. The precision-recall curve and F-1 values, on the other hand, will give insight into the balance between the true positive rate and the positive predictive value, crucial for medical applications where false negatives can have serious implications.

To establish a benchmark for performance, a minimum threshold will be set, derived from the baseline performance of a basic CNN implementation trained on thedatasets. This threshold will act as a reference point to gauge the enhancements brought by the integration of self-attention mechanisms and SVMs. The evaluation process will not only assess the overall

accuracy but also scrutinize the model's ability to generalize across different datasets and under varying conditions. This will involve testing the models on diverse CXR datasets, including CheXpert and ChestX-ray14, to ensure the robustness and adaptability of the solutions.

Furthermore, the testing and evaluation phase will be iterative, involving continuous refinement of the models based on the performance metrics. This process will include adjusting model parameters, re-assessing the integration of attention mechanisms and SVMs, and potentially exploring additional datasets or augmentation techniques to improve performance.

The final evaluation will encapsulate a thorough analysis, comparing the novel CNN architecture against existing benchmarks and standards in the field, ensuring the project aligns with its primary goal of enhancing CXR classification accuracy in a clinically meaningful way.

I will develop custom functions to generate and store reports, graphs, and model architecture visuals in designated experiment directories. These resources will be utilized for comparative analysis of the most promising models, enhancing the evaluation process.

3.5 Work Plan

The project is structured into six key sprints, meticulously planned and visualized through a waterfall Gantt chart. This chart outlines the interdependencies of various tasks, ensuring a coherent flow from one phase to the next. Each sprint incorporates a phase of performance analysis or a documentation segment to consolidate findings or elucidate model insights.

3.5.1 Prototype Phase

The initial phase involves downloading and processing the CheXray-14 Sample Dataset for model application. Addressing the challenge of mapping images to multiple labels, the dataset will undergo extensive analysis and preparation.

A deep learning image pipeline will be created, considering the vast quantity of images. Custom functions will be developed for efficient data handling. The datasets for training and validation will be created.

A rudimentary CNN will serve as a benchmark, trained on the dataset. The core task in this phase is to perform transfer learning from a pretrained ImageNet model, fine-tuning it with the ChestXray-14 sample dataset using Keras. This involves loading pretrained weights, freezing layers, customizing the input/output layers, and training on the new dataset.

Fine-tuning will involve partial unfreezing of the base model and training at a low learning rate to prevent overfitting. Performance metrics will be applied to evaluate the models, with results and necessary visuals duly recorded.

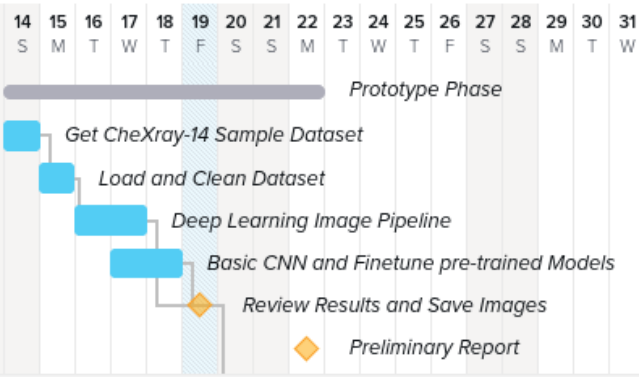

Fig 9 . *Prototype Phase Gantt Chart*

3.5.2 Better Environment Setup

Considering the computational demands of pre-trained, deep models like VGG16 from keras with 138.4M params, an upgrade from my home GPU setup is imperative. Services offering computational resources and storage, such as Google Colab and WANB, will be explored, particularly those providing student free-tiers.

The optimal platform will be chosen to handle the full ChestXray-14 and CheXpert datasets with no bottlenecks. Organizational structures will be implemented for storing model weights, statistics, images, etc. in order to make comparisons between experiments.

Enhanced computational resources will enable extensive transfer learning and fine-tuning on larger models with complete datasets. Performance analyses will be conducted, with results saved. Tentative models include AlexNet, ResNet, and VGG architectures.

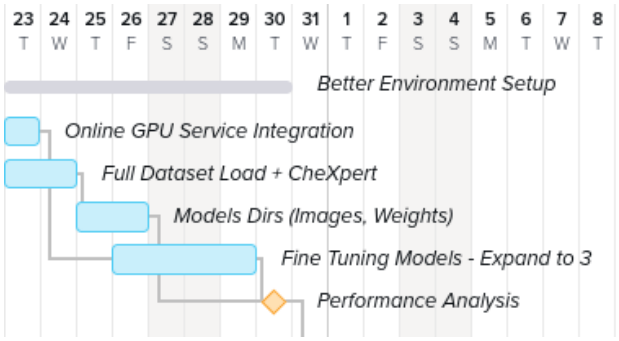

Fig 10 . *Better Environment Setup Phase Gantt Chart*

3.5.3 SVMs Implementation

To perform the SVMs classifier integration I will review the research paper, understand the key concepts and draft an overall plan of the task.

Initial steps will involve using the CNN models as feature extractors, followed by training an SVM with the extracted features and labels, utilizing sklearn's SVM libraries.

A basic CNN and CIFAR10 dataset will first be employed to test the SVM approach. Successful implementation will then be extended to the basic CNN and later on to the fine-tuned models, followed by performance analysis.

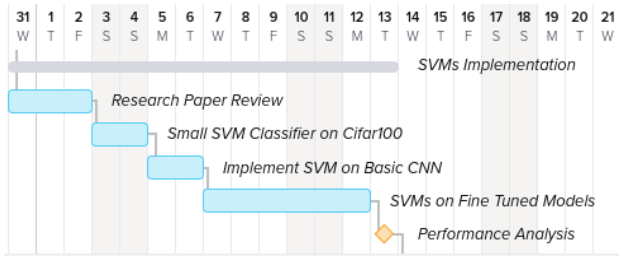

Fig 10 . *SVMs Implementation Gantt Chart*

3.5.4 Attention CBAM Implementation

To perform the CBAM attention mechanism integration, following the SVM section approach, I will begin by reviewing the research paper, understand in detail the inner workings and draft an overall idea of the task.

The unique nature of the attention block requires manual integration between custom layers, varying according to the fine-tuned models architecture. A basic CNN will first be enhanced with CBAM, using past implementations as a reference.

This will then be extended to the fine-tuned models, followed by performance evaluation.

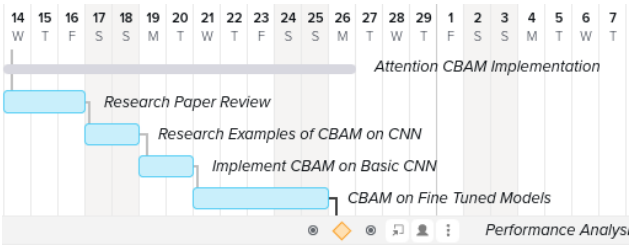

Fig 11. *Attention CBAM Implementation Gantt Chart*

3.5.5 CBAM and SVM Integration

After CBAM and SVM have been integrated, tested and evaluated for performance on all the model architectures that I have, a comparison will be made to choose the model that will be used for the final ensemble.

After it is chosen the SVM classifier will be integrated to the CBAM version of the model, given that only the output layers will be affected. The model will be trained and evaluated.

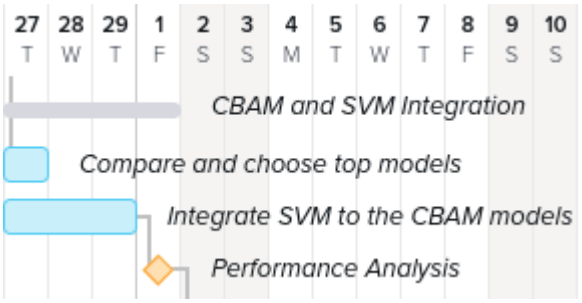

Fig 12. *CBAM and SVM Integration Gantt Chart*

3.5.5 Results Compilation & Final Writing

Finally with all the experiment information disponible, the performance data, etc, I will create the final tables with the results, and choose the figures to display.

There will be a literature review to see if the experimental results were in line with the chosen recommendations. and lastly the report will be updated with the latest information.

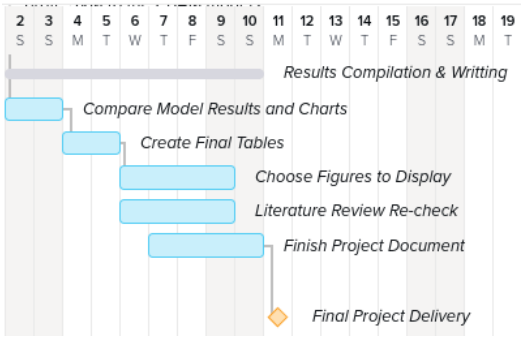

Fig 13. *Results Compilation & Final Writing Gantt Chart*

4. IMPLEMENTATION

4.1 Dataset Creation And Pipeline

Given the size of the CheXray-14 dataset, comprising 120,000 images of 1024x1024 resolution, and the computational limitations of my GeForce GTX 1660 hardware with 32 GB RAM, I initially used a random sample of 5,606 images and their corresponding labels.

Jupyter Notebooks facilitated data processing and model training. Attempts to use Kaggle and Google Vertex AI were hindered by CPU bottlenecks during image processing, leading to the exclusive use of one dataset.

The labels were processed, cleaned, and visualized, with multi-label challenges addressed through individual column processing for each image and direct one-hot encoding. The dataset included validation and testing TXT files for creating necessary training and testing splits, saved as efficient pickle files.

Further research into TensorFlow data structures led to developing a custom pipeline utilizing `tf.data.Dataset.from_tensor_slices` with parallel computing, custom batch sizes, and optimized image loading and pre-fetch functions.

```python
def parse_function(filename, label):
    """Read, decode, resize, and normalize image, returning image and label."""
    image_string = tf.io.read_file(filename)
    image_decoded = tf.image.decode_png(image_string, channels=CHANNELS)
    image_resized = tf.image.resize(image_decoded, [IMG_SIZE, IMG_SIZE])
    image_normalized = image_resized / 255.0
    return image_normalized, label

def create_dataset(filenames, labels):
    """Create a TensorFlow dataset from image paths and labels."""
    dataset = tf.data.Dataset.from_tensor_slices((filenames, labels))
    dataset = dataset.map(parse_function, num_parallel_calls=NUM_CORES)
    dataset = dataset.batch(BATCH_SIZE).prefetch(buffer_size=AUTOTUNE)
    return dataset

# Create TensorFlow datasets
train_ds = create_dataset(X_train, y_train)
val_ds = create_dataset(X_val, y_val)
test_ds = create_dataset(X_test, y_test)
```

Fig 13. Custom Optimized Data Pipeline

A custom Python file was created for dataset optimization, dynamic image path creation, and sys path inclusion. The project also addressed the challenge of an unbalanced dataset initially through the integration of custom weighted sampling which increased the recall and precision significantly.

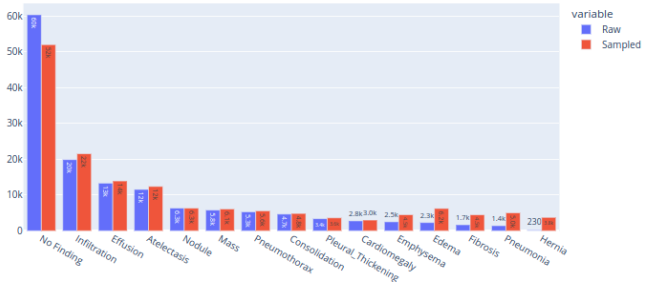

Fig 14. Dataset after custom weighted sampling

During our exploration, I encountered challenges with achieving high recall and precision metrics on the test set, so I opted to follow the approach of the CheXpert team, by expanding the dataset for training and validation, while reserving a smaller, distinct dataset for testing. This was complemented by ensuring no patient data overlap between datasets.

Dataset	Size	Shape
Train	86,524	(86524, 15)
Validation	14,668	(14668, 15)
Test	450	(450, 15)

Fig 15. Final Dataset Split for model training and evaluation

Despite these adjustments, I faced persistent challenges in metric performance, rooted in our task's multi-label, multi-class nature and the one-hot encoding of image labels.

	precision	recall	f1-score	support
Atelectasis	0.00	0.00	0.00	102
Cardiomegaly	0.00	0.00	0.00	30
Consolidation	0.00	0.00	0.00	45
Edema	0.00	0.00	0.00	23
Effusion	0.00	0.00	0.00	123
Emphysema	0.00	0.00	0.00	32
Fibrosis	0.00	0.00	0.00	18
Hernia	0.00	0.00	0.00	4
Infiltration	0.00	0.00	0.00	186
Mass	0.00	0.00	0.00	57
No_Finding	0.55	0.60	0.57	609
Nodule	0.00	0.00	0.00	65
Pleural_Thickening	0.00	0.00	0.00	39
Pneumonia	0.00	0.00	0.00	11
Pneumothorax	0.00	0.00	0.00	69

Fig 16. Bad Results When Evaluating the test set.

4.1.1 MultilabelStratifiedShuffleSplit

To address this challenge, I attempted to generate stratified training and validation splits from the dataset, aiming to ensure proportional representation of each class in the training, validation and test sets. My focus was misdirected; instead of concentrating on aspects of the problem that could significantly impact outcomes, such as the training loss function, I prioritized the dataset's split.

```
from iterstrat.ml_stratifiers import MultilabelStratifiedShuffleSplit

msss = MultilabelStratifiedShuffleSplit(n_splits=1, test_size=0.15, random_state=24)

train_data = train_df['Image Index']
train_labels = train_df[train_df.columns[3:]].values

for train_index, val_index in msss.split(np.zeros(train_data.shape[0]), train_labels):
    X_train, X_val = train_data.iloc[train_index], train_data.iloc[val_index]
    y_train, y_val = train_labels[train_index], train_labels[val_index]

X_test = test_df['Image Index']
y_test = test_df[test_df.columns[2:]].values
```

Fig 17. Custom Stratified Dataset Split Logic.

A critical aspect of this setup was the disproportionate representation of '0's (indicating the absence of a condition) compared to '1's (indicating its presence), which skewed our model training and introduced a significant training bias.

4.2.2 Custom Weighted Cross Entropy Function

To address this imbalance and enhance our model's ability to learn from the underrepresented positive classes ('1's), I created a custom weighted cross entropy function. This strategy aimed to recalibrate the model's focus, ensuring that learning from less frequent, positive instances was emphasized adequately.

```
weights = {0: 1.,1: 5.}

def weighted_cross_entropy_fn(y_true, y_pred):
    tf_y_true = tf.cast(y_true, dtype=y_pred.dtype)
    tf_y_pred = tf.cast(y_pred, dtype=y_pred.dtype)
    weights_v = tf.where(tf.equal(tf_y_true, 1), weights[1], weights[0])
    weights_v = tf.cast(weights_v, dtype=y_pred.dtype)
    ce = K.binary_crossentropy(tf_y_true, tf_y_pred, from_logits=False)
    loss = K.mean(tf.multiply(ce, weights_v))
    return loss

def get_weighted_loss():
    def loss(y_true, y_pred):
        return weighted_cross_entropy_fn(y_true, y_pred)
    return loss
```

Fig 18. Custom Weighted Cross Entropy Function

This custom function intricately adjusts the loss based on the label of each instance, applying a higher penalty for misclassifications of the minority class ('1's). By implementing this tailored approach, I aimed to mitigate the imbalance's adverse effects, fostering a more equitable and effective learning process. This approach proved to be successful and is used on most of the later models.

4.2 Initial Models and Experiments

4.2.1 Callbacks And Others

I consistently used a set of callbacks and metrics, including the Adam optimizer for its efficiency in handling sparse gradients and its adaptability, alongside BinaryCrossentropy loss alongside with my custom weighted implementation with a sigmoid activation function, optimizing for multi-label outcomes.

Callbacks included ModelCheckpoint, facilitating continuous model saving and leveraging previous best weights for enhanced and continuing training; EarlyStopping, preventing overfitting by halting training when improvements cease; and ReduceLROnPlateau, adjusting the learning rate to optimize performance.

```python
# https://keras.io/api/callbacks/model_checkpoint/
checkpoint = ModelCheckpoint(
    weight_path,
    monitor='val_loss',
    verbose=1,
    save_best_only=True,
    mode='min',
    save_weights_only = True)

# https://keras.io/api/callbacks/early_stopping/
earlystop = EarlyStopping(
    monitor = 'val_loss',
    min_delta = 1e-4,
    patience = 5,
    mode = 'min',
    restore_best_weights = True,
    verbose = 1)

# https://keras.io/api/callbacks/reduce_lr_on_plateau/
reduceLROnPlat = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.1,
    patience=2,
    verbose=1,
    mode='auto',
    min_delta=1e-4,
    cooldown=1,
    min_lr=1e-6)
```

Fig 19. Callbacks used and parameters

Metrics like accuracy, precision, recall, and AUC were used for a comprehensive assessment of model performance, essential for evaluating the model's ability to distinguish between classes accurately.

```python
callbacks_list = [checkpoint, earlystop, reduceLROnPlat]


Basic_CNN.compile(optimizer='adam',
                  loss=keras.losses.BinaryCrossentropy(label_smoothing=0.0),
                    metrics=[
                        keras.metrics.BinaryAccuracy(name='accuracy'),
                        keras.metrics.Precision(name='precision'),
                        keras.metrics.Recall(name='recall'),
                        keras.metrics.AUC(name='auc', multi_label=True)])

Basic_CNN.summary()
```

Fig 20. Metric Used and Compile Step

The goal was to train all models for a minimum of 10 epochs; however, due to computational and time limitations, some models underwent shorter training periods. Regardless, each model was assessed on both the validation and test datasets to ensure thorough evaluation.

4.2.2 Baseline CNN

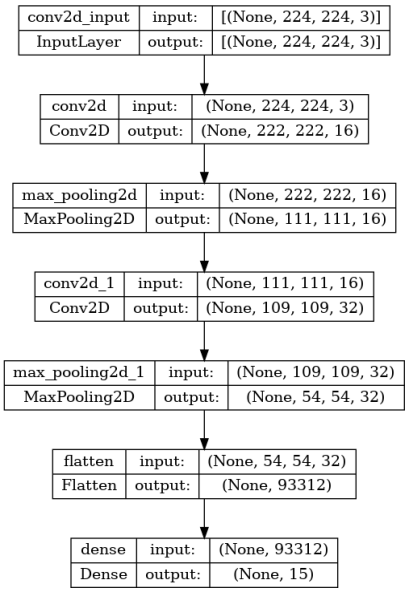

Fig 22. Baseline CNN Model

The baseline CNN model Basic_CNN consists of 2 blocks of Conv2D for feature extraction combined with MaxPooling2D that reduces the spatial dimensions, a flatten layer to transform the 3D output into a 1D array, and a dense layer for classification. The Basic_CNN was trained over 12 epochs and had some issues with overfitting almost immediately, due to the simplicity of the network and the lack of dropout layers which led to very high accuracy, while missing the recall and precision on many labels. Still the AUC values were surprisingly good.

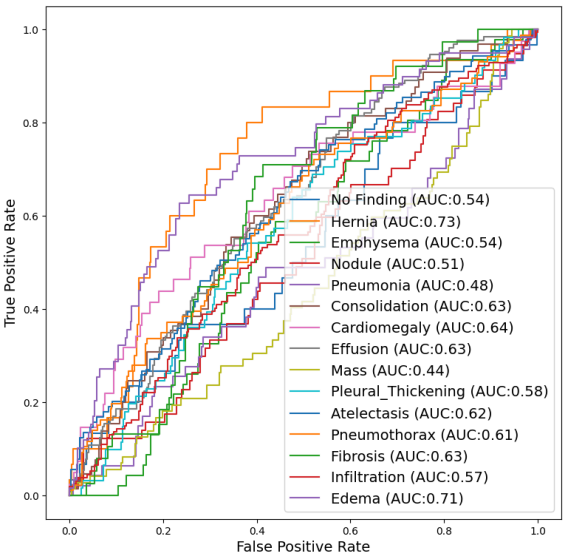

Fig 23. AUC-ROC Values for Basic_CNN, explained on Evaluation

4.2.3 Pre-trained Models & Transfer Learning

The pre-trained models VGG16, ResNet50 were initially planned to be used but once again due memory computational constraints a smaller model MobileNet[17] with only 4.3M parameters was chosen. The procedure for transfer learning was executed as stated on section 5.3.1. The resulting model architecture is as follows:

This new model from MobileNet with ImageNet weights -MobileNet_finetuned- is compiled with the same arguments as the Basic_CNN model, and trained over five of epochs.

To finetune the model I set all the layers to trainable and train with a low training rate value. The fine tuned m model seems to begin to overfit early on. Given the computational constraints I were not able to run the training for more than 5 epochs.

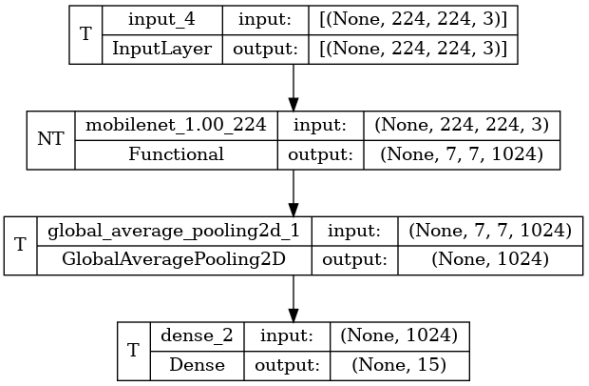

Fig 23. MobileNet Pre-trained model integration

4.4 Prototype Models Summary Table

The MobileNet_finetuned model shows the best overall performance, with the highest accuracy, precision, AUC, and PRC scores. The Base_CCN has the highest loss and the lowest scores for precision, recall, AUC, and PRC, indicating it may be the least effective model among the three. The MobileNet_transfer model, while not as effective as the fine tuned version, still outperforms the Base_CCN across all metrics except for loss, where it is marginally better.

Although it is a good start, for the implementation of the CBAM attention mechanism the ResNet architecture will be more suitable.

Model	Loss	Accuracy	Precision	Recall	AUC	PRC
(Sample) Base_CCN	0.87099	0.91390	0.47831	0.28096	0.70891	0.29641
(Sample) MobileNet_transfer	0.20629	0.92911	0.65850	0.32343	0.85797	0.47473
(Sample) MobileNet_finetuned	0.20041	0.93102	0.68103	0.33546	0.87162	0.50567
(Full) Base_CCN	0.12220	0.95500	0.81710	0.61620	0.95890	0.79480
(Full) ResNet18	1.31170	0.91550	0.51340	0.39662	0.68051	0.29074

Fig 24. Preliminary Results Notice Mobile Fine Tuned has the best results overall.

4.5 ResNet-18 and Implementation

Following the outlined work plan, the implementation was initially intended to use Keras's version of ResNet. However, Keras provides weights only for ResNet-32 and more complex variants, which my computer lacks the capacity to train efficiently. Consequently, the necessity to develop a simplified version of ResNet from scratch became apparent.

This endeavor, also conveniently facilitates the integration of the CBAM (Convolutional Block Attention Module), specifically channel and spatial attention mechanisms, directly within the residual blocks. One drawback of this approach is the inability to utilize pre-trained ImageNet weights, which implies that training will require more time.

A key feature of ResNet is the utilization of a tensor identity at the start of each block, which is then added back before the final ReLU layer. Challenges arise when the stride of the convolution alters the image dimensions, making the addition operation infeasible. To address this, downsampling and batch normalization are employed to enable tensor summation.

Below is the visual diagram illustrating this downsampling process:

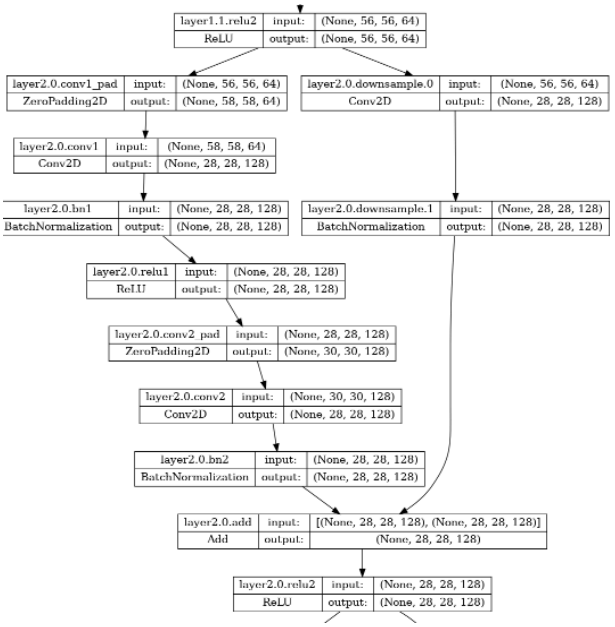

Fig 25. Diagram for the downsampling process

4.5.1 resnet_layer Function

It acts as the foundational building block for the network, implementing the sequence of convolution, batch normalization, and activation.

It applies spatial convolution over images. The num_filters, kernel_size, and strides parameters control the layer's complexity, receptive field, and downsampling, respectively.

The use of 'he_normal' initialization and L2 regularization helps in efficient training and combating overfitting, it normalizes the activations from the previous layer, improving stability and speed of training, finally, introduces non-linearity into the network, allowing it to learn complex patterns.

```python
def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):
    """
    2D Convolution-Batch Normalization-Activation stack builder.
    """
    conv = Conv2D(num_filters,
                  kernel_size=kernel_size,
                  strides=strides,
                  padding='same',
                  kernel_initializer='he_normal',
                  kernel_regularizer=l2(1e-4))

    x = inputs
    if conv_first:
        x = conv(x)
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
    else:
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
        x = conv(x)
    return x
```

Fig 26. ResNet Layer Function

4.5.2 resnet_block Function

Constructs a ResNet block that can optionally include CBAM for attention-based feature refinement. It optionally applies a stride-2 convolution to reduce spatial dimensions, matching the shortcut connection dimensions when necessary, merges the block's input and output, facilitating deep network training by allowing gradients to flow directly through the network.

```python
def resnet_block(inputs, num_filters, CBAM=False,
downsample=False):
    """
    Creates a ResNet block with an optional Convolutional
    Block Attention Module (CBAM).
    """
    x = inputs
    if downsample:
        inputs = resnet_layer(inputs=inputs,
        num_filters=num_filters, kernel_size=1, strides=2,
        activation=None, batch_normalization=True)

    x = resnet_layer(inputs=x, num_filters=num_filters,
    strides=2 if downsample else 1)
    x = resnet_layer(inputs=x, num_filters=num_filters,
    activation=None)

    if CBAM:
        x = cbam_block(x)  # Applies CBAM if flagged

    x = Add()([x, inputs])
    x = Activation('relu')(x)
    return x
```

Fig 27. Resnet_block Function

4.5.3 resnet_18 Function

Builds the ResNet-18 model, layering ResNet blocks sequentially to form the deep architecture. Starts with a larger convolution and max pooling to reduce input dimensionality while preserving essential features, then implements four stages of ResNet blocks, doubling the number of filters and applying downsampling at the start of stages 2-4 to progressively refine and compress the feature maps. Reduces each feature map to a single value, focusing the model on the most salient features and ends up with a dense layer for classification, where the sigmoid activation suits multi-label tasks by predicting independent probabilities for each class.

```
def resnet_18(input_shape, num_classes=10, use_cbam=False):
    """
    Builds a ResNet-18 model with an optional CBAM.
    """
    inputs = Input(shape=input_shape)
    x = resnet_layer(inputs=inputs, num_filters=64, kernel_size=7,
    strides=2, activation='relu', batch_normalization=True)
    x = MaxPooling2D(pool_size=(3, 3), strides=2, padding='same')(x)

    # Defines each stage of ResNet-18
    for stage in [(64, False), (128, True), (256, True), (512, True)]:
        num_filters, downsample = stage
        x = resnet_block(x, num_filters, CBAM=use_cbam,
        downsample=downsample)
        x = resnet_block(x, num_filters, CBAM=use_cbam)

    x = GlobalAveragePooling2D()(x)
    outputs = Dense(num_classes, activation='sigmoid',
    kernel_initializer='he_normal')(x)

    model = Model(inputs=inputs, outputs=outputs)
    return model
```

Fig 28. ResNet-18 Function Model Initialization, Input and Outputs

4.6 ResNet-18 Attention Implementation

4.6.1 Convolutional Block Attention Module (CBAM)

The Convolutional Block Attention Module (CBAM) enhances the ResNet architecture by introducing attention mechanisms that allow the model to focus on the most informative features in both channel and spatial dimensions. CBAM is integrated directly into the residual blocks of the ResNet-18, providing a modular and efficient method to refine feature maps generated by convolutional layers.

```
def cbam_block(cbam_feature, ratio=8):
    cbam_feature = channel_attention(cbam_feature, ratio)
    cbam_feature = spatial_attention(cbam_feature)
    return cbam_feature
```

Fig 29. CBAM Block combines channel & spatial attention.

4.6.2 Channel Attention

The channel attention mechanism focuses on 'what' is meaningful given an input image by emphasizing important features and suppressing less useful ones across channel dimensions. This is achieved through a combination of global average pooling and global max pooling, followed by shared dense layers to capture channel-wise dependencies. The resulting feature map is scaled by a sigmoid activation to ensure that channel attention values are between 0 and 1, effectively modulating the input feature map.

```
def channel_attention(input_feature, ratio=8):
    channel_axis = 1 if K.image_data_format() == "channels_first" else -1
    channel = input_feature.shape[channel_axis]

    shared_layer_one = Dense(channel//ratio, activation='relu', kernel_initializer='he_normal',
    use_bias=True, bias_initializer='zeros')
    shared_layer_two = Dense(channel, kernel_initializer='he_normal', use_bias=True,
    bias_initializer='zeros')

    avg_pool = GlobalAveragePooling2D()(input_feature)
    avg_pool = Reshape((1,1,channel))(avg_pool)
    avg_pool = shared_layer_one(avg_pool)
    avg_pool = shared_layer_two(avg_pool)

    max_pool = GlobalMaxPooling2D()(input_feature)
    max_pool = Reshape((1,1,channel))(max_pool)
    max_pool = shared_layer_one(max_pool)
    max_pool = shared_layer_two(max_pool)

    cbam_feature = Add()([avg_pool, max_pool])
    cbam_feature = Activation('sigmoid')(cbam_feature)

    if K.image_data_format() == "channels_first":
        cbam_feature = Permute((3, 1, 2))(cbam_feature)

    return multiply([input_feature, cbam_feature])
```

Fig 30. Channel Attention Function

4.6.3 Spatial Attention

Spatial attention, on the other hand, focuses on 'where' is an informative part of an image by highlighting spatial locations with salient features. It operates by applying average pooling and max pooling across the channel dimension and concatenating the results. A convolutional layer then processes this combined feature map, and a sigmoid activation function scales the output, producing a spatial attention map. This map is used to refine the input feature map by emphasizing important spatial regions.

```
def spatial_attention(input_feature):
    kernel_size = 7

    if K.image_data_format() == "channels_first":
        cbam_feature = Permute((2, 3, 1))(input_feature)
    else:
        cbam_feature = input_feature

    avg_pool = Lambda(lambda x: K.mean(x, axis=3, keepdims=True))(cbam_feature)
    max_pool = Lambda(lambda x: K.max(x, axis=3, keepdims=True))(cbam_feature)
    concat = Concatenate(axis=3)([avg_pool, max_pool])
    cbam_feature = Conv2D(filters=1, kernel_size=kernel_size, strides=1, padding='same',
    activation='sigmoid', kernel_initializer='he_normal', use_bias=False)(concat)

    if K.image_data_format() == "channels_first":
        cbam_feature = Permute((3, 1, 2))(cbam_feature)

    return multiply([input_feature, cbam_feature])
```

Fig 31. Spatial Attention Function

5. EVALUATION

5.7 Issues with SVMs Integration

In our project, we initially considered employing a Support Vector Machine (SVM) for classification, given its robustness in handling high-dimensional data and its effectiveness in various machine learning tasks.

The approach intended to utilize a pipeline integrating SVM with feature representations extracted from Convolutional Neural Networks (CNNs), leveraging the discriminative capabilities of CNNs for feature extraction combined with the classification strength of SVMs. Specifically, the strategy involved:

5.7.1 Feature Extraction

Utilizing CNNs like the Base Model and ResNet to extract features from the image data. This step is critical as SVMs themselves do not perform feature extraction but rely on input features for classification. Given that our CNN models (Base Model and ResNet) do not include dense layers before the output, we planned to extract features from the final convolutional layers, which capture high-level visual patterns relevant for classification.

5.7.2 SVM Classification

With features extracted, we aimed to employ an SVM classifier to perform the final classification. The rationale behind this choice lies in SVM's capability to find the optimal hyperplane that separates the data points of different classes in the feature space, potentially offering a robust classification mechanism.

To optimize the SVM's performance, we considered using Grid Search CV for hyperparameter tuning, exploring combinations of the C parameter (which controls the trade-off between achieving a low error on the training data and maintaining simplicity of the decision surface) and the gamma parameter (which defines the influence of data points on the decision boundary) with an RBF kernel to find the best model configuration.

5.7.3 Grid Search for Hyperparameter Optimization

The planned approach included using GridSearchCV from scikit-learn to automate the search for the best hyperparameters (C and gamma) for the SVM classifier. This exhaustive search strategy was expected to iteratively train multiple SVM models with different combinations of C and gamma values on a subset of the extracted features, evaluating each model's performance to select the optimal configuration.

5.7.3 Challenges and Impracticalities

Despite the theoretical advantages of combining CNNs with SVMs for classification, practical challenges related to computational resources and the complexity of implementing such a pipeline became evident. The extensive memory requirement for computing and storing the SVM's kernel matrix, especially for a dataset of our scale, along with the absence of GPU acceleration for SVM training, made this approach impractical.

Furthermore, the additional computational overhead for extracting and handling high-dimensional feature representations from CNNs for SVM input posed significant hurdles, given our hardware constraints.

While the combined use of CNNs for feature extraction and SVMs for classification presented a promising approach on paper, the memory and computational requirements, along with the complexity of the pipeline and the lack of direct support for multi-label classification in SVMs, led us to conclude that this strategy was not feasible given our project's constraints.

5.7 Evaluation Strategy

5.7.1 Performance Metrics Used

The evaluation strategy for this project will be comprehensive and aligned with the objective of achieving high diagnostic accuracy in CXR classification.

To ensure the developed models meet the required standards, I will primarily focus on the Area Under the Curve of Receiver Operating Characteristics (AUC-ROC) alongside loss, accuracy and precision-recall, and F-1 Values.

These metrics are critical as AUC-ROC provides a measure of the model's ability to distinguish

between classes (e.g., diseased vs. healthy), essential in medical diagnostics.

The precision-recall and F-1 values, on the other hand, will give insight into the balance between the true positive rate and the positive predictive value, crucial for medical applications where false negatives can have serious implications.

Finally, the weighted average is crucial for addressing the class imbalance in the dataset, where some conditions are more common than others.

It adjusts the impact of each class based on its frequency, offering a more balanced view of the model's performance across all conditions. This method ensures our performance metrics reflect the model's effectiveness and fairness in diagnosing both prevalent and rare chest conditions, aligning with our goal to develop a reliable and equitable diagnostic tool.

5.7.2 ROC Curves and AUC Analysis

To evaluate the diagnostic capability of our model across the various pathologies in chest X-ray images, we employ Receiver Operating Characteristic (ROC) curves. These curves graphically illustrate the model's performance by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

The code snippet provided forms part of our analysis, generating ROC curves for each class (or pathology) within our multi-label dataset.

```
# create plot
fig, c_ax = plt.subplots(1,1, figsize = (9, 9))
for (i, label) in enumerate(ALL_LABELS):
    fpr, tpr, thresholds = roc_curve(y_test[:,i].astype
    (int), quick_model_predictions[:,i])
    c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  %
    (label, auc(fpr, tpr)))

# Set labels for plot
c_ax.legend()
c_ax.set_xlabel('False Positive Rate')
c_ax.set_ylabel('True Positive Rate')
```

Fig 32. Function to generate ROC Curves and AUC values

.

This approach allows us to visually assess not just the overall performance of the model, but also its capability to accurately diagnose specific conditions within the chest X-rays, a crucial aspect in medical imaging analysis.

5.7.3 Visualizing Model Performance

Following the described methodology, multiple figures can be showcased to highlight the model's performance. Each figure represents the ROC curves for different pathologies, annotated with their respective AUC scores.

The Basic CNN model has the lowest AUC (Area Under Curve) scores among the three models, indicating its performance in distinguishing between the different pathologies is less accurate.

AUC values are quite close to 0.5 for many pathologies, which suggests performance only slightly better than random guessing for those classes.

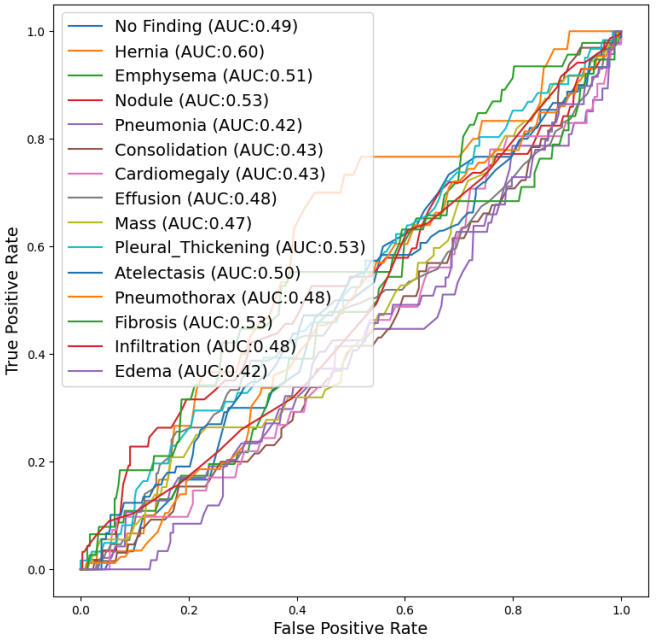

Fig 33. Basic CNN AUC-ROC Values

The ResNet-18 model shows improved performance over the Basic CNN, with higher AUC scores across most pathologies. This suggests better discriminative power for identifying the presence of specific conditions in the X-ray images.

However, some pathologies like 'Mass' and 'Pleural_Thickening' still hover around an AUC of 0.50, indicating room for improvement.

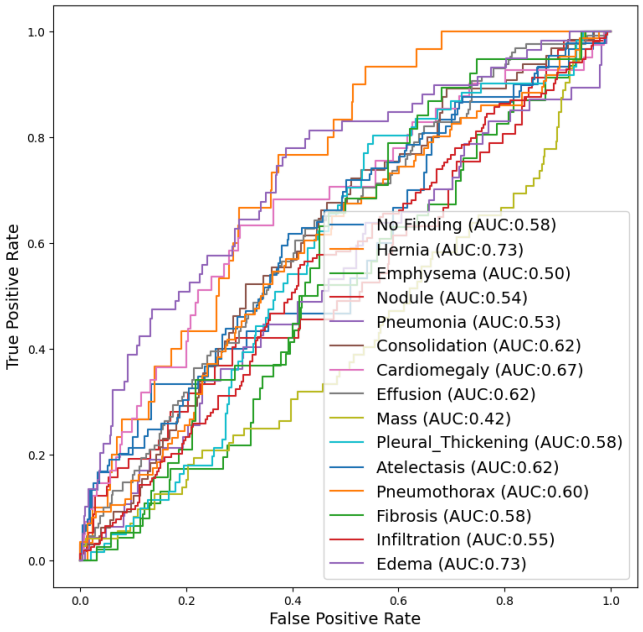

Fig 34.. ResNet-18 AUC-ROC Values

The addition of CBAM (Convolutional Block Attention Module) to ResNet-18 further improves the AUC scores for nearly all pathologies, suggesting that the attention mechanism helps the model focus on more relevant features for each class.

This model demonstrates the highest AUC scores, particularly for pathologies like 'Cardiomegaly' and 'Edema', indicating a strong ability to correctly identify these conditions.

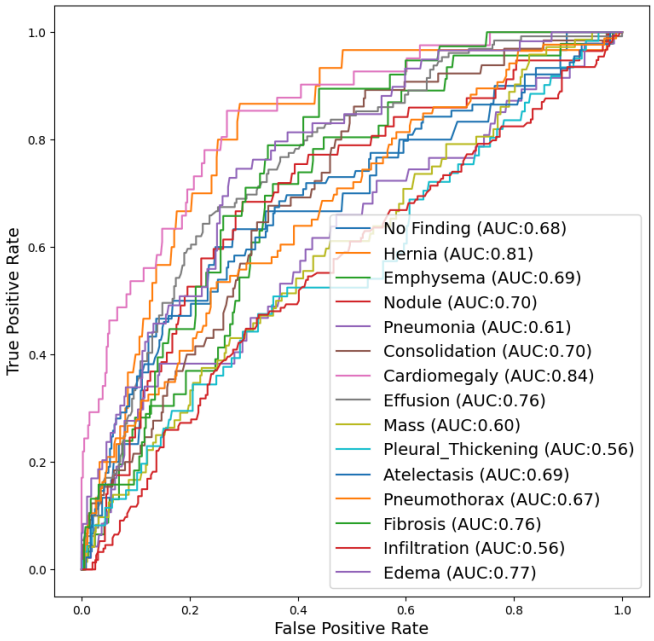

Fig 35. ResNet-18 + CBAM AUC-ROC Values

The integration of attention mechanisms via CBAM with ResNet-18 leads to the most promising results. It not only improves the model's overall ability to differentiate between pathologies but also shows that focusing the model's attention on specific areas of an image can significantly enhance performance. Despite this improvement, some pathologies still present challenges, and as such, there is potential for further optimization either through model architecture adjustments, additional data, or more advanced preprocessing techniques.

5.7.4 Weighted Average Results

Model	Precision (%)	Recall (%)	F1-Score (%)
Basic CNN	11	33	16
ResNet-18	25	30	19
ResNet-18 + CBAM	22	43	27

Fig 36.

The results from the weighted averages of the precision, recall, and F1-score offer an insightful perspective on the performance of each model. The Basic CNN achieved a precision of 11%, recall of 33%, and an F1-score of 16%. While the precision is low, the model has a relatively higher recall, suggesting that it is more capable of identifying true positive cases, but at the expense of correctly classifying negative cases.

ResNet-18 demonstrated a balanced performance with a slight increase in precision to 25% and a marginal reduction in recall to 30%, leading to a modest F1-score of 19%. This indicates an improvement in the model's ability to correctly predict positive cases while reducing false positives compared to the Basic CNN.

The integration of CBAM with ResNet-18 resulted in a notable increase in recall to 43% while maintaining a reasonable precision rate of 22%. This enhancement in recall suggests that the attention mechanisms in CBAM help the model to better identify relevant features for class prediction, leading to a higher F1-score of 27%. It is a significant improvement, showing that the model with CBAM is more adept at identifying true positives and is more balanced overall in

terms of precision and recall.

5.7.4 predict_pathologies Function

The predict_pathologies function pre-processes a specified image from a dataset, adjusts it for the model input, and then employs the model to predict the likelihood of each pathology.

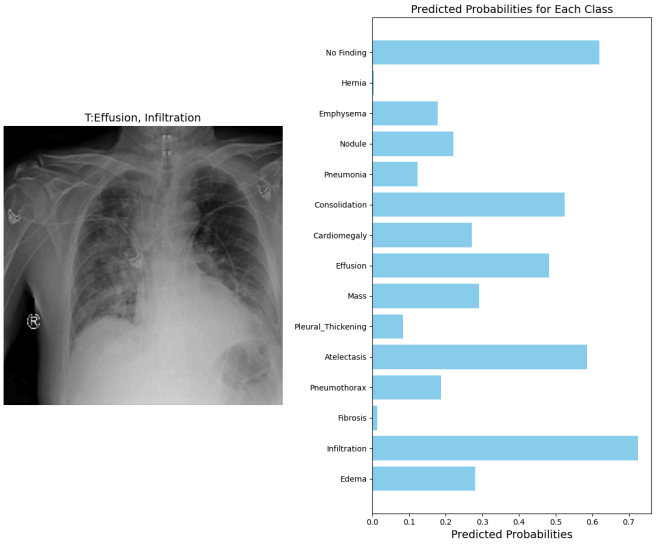

Fig 37. Output of the predict_pathologies function.

The model outputs probabilities that indicate its confidence in the presence of each condition in the image.

Visually, the function presents both the image and the model's predictions side by side. It displays the image on the left with its true labels for reference. On the right, it features a horizontal bar chart that represents the model's predicted probabilities for each class label, allowing for an immediate visual assessment of the model's performance.

This dual presentation of image and data is particularly insightful. It not only quantifies the model's predictions but also qualitatively showcases the alignment between predicted probabilities and actual conditions. This is crucial for tasks like medical diagnostics, where understanding model certainty can inform decisions.

6. Conclusion

Our project "Attentive Diagnostics" embarked on enhancing the accuracy and efficiency of Chest X-ray (CXR) analysis using a novel AI-driven method. The introduction of attention mechanisms with CNNs aimed to mimic a radiologist's focus, potentially minimizing human error and bias in interpretations.

However, we steered away from leveraging Support Vector Machine (SVM) classification due to the impracticality of its integration with our extensive dataset and computational constraints. The evaluation demonstrated the project's alignment with the ambitious goal of providing high diagnostic accuracy.

The AUC-ROC metric confirmed the model's capability to distinguish between various pathologies, while precision-recall and F-1 values offered insight into the balance between the true positive rate and predictive value—critical aspects for medical applications where the cost of false negatives is substantial. The adoption of a weighted average evaluation approach ensured fairness in assessing performance across prevalent and rare conditions.

In implementation, the integration of CBAM with the ResNet-18 model notably enhanced performance. This approach focused the model's attention on relevant image features, as evidenced by the increased AUC scores, especially for conditions like 'Cardiomegaly' and 'Edema'. The predict_pathologies function was a testament to the project's success, providing a clear and interactive visual assessment of model predictions alongside the actual image data.

Despite facing challenges such as computational limitations, the inability to utilize SVMs, and the requirement for additional steps in feature extraction due to the architecture of the CNN models, the project adapted and overcame these hurdles through meticulous strategy and methodical model selection.

The thoroughness of the evaluation, the comprehensive coverage of performance metrics, and the innovative use of attention mechanisms within CNNs underscored our systematic approach to the project, guiding us to refine a diagnostic tool that seeks to improve upon the current standards for automated chest radiography, mindful of the challenges that lie ahead.

THANK YOU FOR READING!

6. REFERENCES

[1] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 86, 11 (Nov. 1998), 2278-2324. DOI: https://doi.org/10.1109/5.726791.

[2] Krizhevsky, A., Sutskever, I., and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

[3] Simonyan, K., and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. 2015. Going Deeper with Convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'15), 1-9.

[5] He, K., Zhang, X., Ren, S., and Sun, J. 2016. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'16), 770-778. DOI: 10.1109/CVPR.2016.90.

[6] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[7] Architectures of Deep Convolutional Neural Networks. Artificial Intelligence Review. DOI: https://doi.org/10.1007/s10462-020-09825-6. arXiv:1901.06032 [cs.CV]. Pattern Recognition Lab, DCIS, PIEAS, Nilore, Islamabad 45650, Pakistan; Deep Learning Lab, Center for Mathematical Sciences, PIEAS, Nilore, Islamabad 45650, Pakistan.

[8] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, Ronald M. Summers.

ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly- Supervised

Classification and Localization of Common Thorax Diseases, IEEE CVPR, pp. 3462-3471,2017

[9] Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., Seekins, J., Mong, D. A., Halabi, S. S., Sandberg, J. K., Jones, R., Larson, D. B., Langlotz, C. P., Patel, B. N., Lungren, M. P., Ng, A. Y. 2019. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. arXiv:1901.07031. [Online]. Available: https://arxiv.org/abs/1901.07031

[10] Chowdhury, M.E.H., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M.A., Mahbub, Z.B., Islam, K.R., Khan, M.S., Iqbal, A., Emadi, N.A., Reaz, M.B.I., Islam, M.T. 2020. Can AI help in screening Viral and COVID-19 pneumonia? IEEE Access, 8, pp. 132665-132676. [Online]. Available: https://ieeexplore.ieee.org/document/9144185

[11] Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S., Chowdhury, M.E. 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images. Computers in Biology and Medicine, [Online]. Available: https://doi.org/10.1016/j.compbiomed.2021.104319

[12] Irvin, J. et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. Proc. AAAI Conf. Artif. Intell.33, 590–597 (2019).

[13] Bressem, K.K., Adams, L.C., Erxleben, C. *et al.* Comparing different deep learning architectures for classification of chest radiographs. *Sci Rep* 10, 13590 (2020). https://doi.org/10.1038/s41598-020-70479-z

[14] Francois Chollet. 2017. Deep Learning with Python (1st. ed.). Manning Publications Co., USA.

[15] Aurelien Geron. 2019. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd. ed.). O'Reilly Media, Inc.

[16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: a system for large-scale machine learning. In Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'16). USENIX Association, USA,

265–283.

[17] Team, Keras. Keras Documentation: Keras applications, Keras. Available at: https://keras.io/api/applications/ (Accessed: 19 January 2024).

[18] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., Lungren, M. P., & Ng, A. Y. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. arXiv:1711.05225

[19] Hou, J., Gao, T. Explainable DCNN based chest X-ray image analysis and classification for COVID-19 pneumonia detection. Sci Rep 11, 16071 (2021). https://doi.org/10.1038/s41598-021-95680-6

[20] Almezhghwi, K., Serte, S. & Al-Turjman, F. Convolutional neural networks for the classification of chest X-rays in the IoT era. Multimed Tools Appl 80, 29051–29065 (2021). https://doi.org/10.1007/s11042-021-10907-y

[21] Rao, A., Park, J., Woo, S., Lee, J.-Y., & Aalami, O. (2021). Studying the Effects of Self-Attention for Medical Image Analysis. In Proceedings of the ICCV 2021 CVAMD Workshop on Image and Video Processing (eess.IV); Computer Vision and Pattern Recognition (cs.CV). arXiv:2109.01486 [eess.IV].

[22] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In Proceedings of the European conference on computer vision (ECCV), pages 3–19, 2018.

[23] Kobiso. 2021. Research on Convolutional Block Attention Module (CBAM). Kobiso's Blog. Available at https://kobiso.github.io//research/research-CBAM/.