Classification of bone fracture from 2D x-ray images using transfer learning and attention module on FracAtlas dataset

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Bachelor in Computer Science & Engineering

by

Sayeda Sanzida Ferdous Ruhi
CSE 072 08182
&
Fokrun Nahar
CSE 072 08192

Supervised by: Adnan Ferdous Ashrafi
Senior Lecturer

Department of Computer Science and Engineering
STAMFORD UNIVERSITY BANGLADESH

March 2024

Abstract

One of the most challenging research domain in application of computer vision in recent bio-medical problems is to detect and localize fracture in bones. Fracture may occur in many parts of our body. To easily classify it, X-radiation (X-ray), magnetic resonance imaging (MRI), or computed tomography (CT) etc. methods are used. One of the main research scope is to automatically detect these fractures without the intervention of orthopedic doctors in order to make the diagnosis more efficient and timely. In recent years, for this task many deep-learning approaches have been used to classify these fractures. In this study we deployed transfer learning approach of deep learning to efficiently classify bone fracture. This study aims to classify bone fractures of a fracture dataset named FracAtlas using an attenuated InceptionV3 model. Before adding Bottleneck attention module (BAM), we used several pre-trained models and find 89.05% accuracy. And after adding attention module, we achieved 90.2% accuracy in our study.

Keywords: Bone Fracture classification, transfer learning, deep learning, Inception V3, BAM attention module.

Approval

The Thesis Report ”Classification of bone fracture from 2D x-ray images using transfer learning and attention module on FracAtlas dataset” submitted by Sayeda Sanzida Ferdous Ruhi ID: CSE 072 08182, Fokrun Nahar ID: CSE 072 08192, to the Department of Computer Science & Engineering, Stamford University Bangladesh, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science (B.Sc.) in Computer Science & Engineering and approved as to its style and contents.

Board of Examiner’s Name, Signature, and Date:

....	....	....
(Board Member 1)	**(Board Member 2)**	**(Board Member 3)**
Date:	Date:	Date:

Supervisor’s Signature and Date:

....
Adnan Ferdous Ashrafi

Date:

Declaration

We, hereby, declare that the work presented in this Thesis is the outcome of the investigation performed by us under the supervision of Adnan Ferdous Ashrafi, Senior Lecturer, Department of Computer Science & Engineering, Stamford University Bangladesh. We also declare that no part of this Project and thereof has been or is being submitted elsewhere for the award of any degree or Diploma.

Signature and Date:

....
Sayeda Sanzida Ferdous Ruhi
Date:

....
Fokrun Nahar
Date:

Dedicated to our parents and teachers

Acknowledgments

We would like to take this opportunity to express our sincere gratitude to all those who have supported us throughout this research.

First and foremost, we would like to thank our thesis supervisor, Adnan Ferdous Ashrafi, for his invaluable guidance, support, and encouragement throughout this entire journey. His expertise and knowledge have been instrumental in shaping this thesis and pushing us to achieve our best.

We would also like to thank the faculty members and staff of the Department of Computer Science & Engineering, who have provided us with a rich academic environment, stimulating discussions, and resources to support our research.

We extend our appreciation to our family and friends for their unwavering support and encouragement throughout this journey. Their belief in us and our abilities has been a constant source of motivation.

Finally, we would like to acknowledge the countless individuals who have contributed to this thesis in ways both seen and unseen. Your support, encouragement, and assistance have been invaluable and deeply appreciated.

Thank you all for your contributions to this research and for being a part of this journey with us.

Table of Contents

List of Figures

List of Tables

1 Introduction

Bone fractures are one of the most common injuries nowadays. Every year, 2.7 million fractures occur across the EU6 nations, France, Germany, Italy, Spain, Sweden, and the UK [1]. A bone fracture occurs when there is a break or cracks in a bone, resulting in a disruption of its normal structure and function. Fracture creates hazards to one's daily life. and sometimes, it is proven life-threatening if the fracture occurs in the skull or in any other parts of the body that are directly connected to the cause of death.

Fracture detection is a vital job as it is one of the most challenging works in bio-medical subjects. Detecting fractures means detecting each single abnormality in bones and it requires more focus. This work needs to be done more accurately. We detect fractures of a body part from X-ray images of its locations. We also use MRI, CT, etc. methods to identify fractures. By identifying fractures, we can find out how damaged the bones are and what measures need to be taken to overcome the damage.

As fracture detection requires more focus and more accuracy, sometimes, it becomes tough for human beings to detect fractures. The slight changes in bones can elude the human eye. So for this reason, it is better to use AI models to identify these fractures as there is a huge possibility for AI models to subtly identify slight injuries on bones. Nowadays, many deep-learning algorithms are being used to identify fractures. Many pre-trained models can be found to identify fractures. And identifying fractures using these AI models can be proven useful as it saves our time. However, it is not fully sustainable to depend on AI to identify fractures as AI can be faulty in some cases. But overall, considering the usefulness of AI models, we can say that the use of AI in fracture classification is quite recommendable.

1.1 Motivation

AI models bring a new dimension of efficiency to the classification of fractures. With their ability to analyze vast amounts of medical imaging data swiftly and accurately, we can significantly reduce the time it takes to diagnose and categorize fractures. This not only enhances the speed of patient care but also allows us to allocate our time and resources more effectively. Moreover, the precision and accuracy that AI models offer are unparalleled. They can discern intricate patterns and subtleties in medical images that might elude the human eye. This level of detail ensures that we can provide more accurate diagnoses, leading to better-informed treatment plans and improved patient outcomes. Also, in the realm of education, incorporating AI models into fracture classification becomes a powerful tool for learning and skill development. In conclusion, the use of AI models in classifying fractures is not merely a technological trend but a fundamental shift that has the potential to redefine our approach to orthopedic care.

1.2 Research Outcome

The research outcome of an experiment depends on various factors including the quality and quantity of datasets. In this study, we have used an fracture image dataset to train, validate and test our model which is huge in number and the images are formed from X-ray image samples. We have created an attenuated model that takes input from X-ray images and classifies whether a bone is fractured or not. The model proposed in this study has higher accuracy with the FracAtlas [2] dataset and it is validated with the Bone Fracture [3] dataset and with the MURA [4] dataset which are also fracture image datasets.

1.3 Types of Tools

Classifying fractures is one of the daring deeds as it holds an important place in the bio-medical section. We determine fractures in bones after examining the X-ray images. It can be done both manually and using AI tools. There are several types of tools available for fracture classification, including computer vision libraries, machine learning frameworks, and pre-trained models. We have used the Keras applications to import pre-trained models to classify fractures in our dataset.

1.3.1 Pre-trained models

Keras [5] is a deep learning API written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch. Keras is simple, flexible and powerful. As a multi-framework API, Keras can be used to develop modular components that are compatible with any framework. The core data structures of Keras are layers and models.

Keras is a highly flexible framework, suitable to iterate on state-of-the-art research ideas. Keras follows the principle of progressive disclosure of complexity. Keras can be used to quickly develop new training procedures or state-of-the-art model architectures. The Keras framework offers a user-friendly interface for building and training neural networks as well as a selection of pre-trained models that can be applied to several different applications, including image classification and natural language processing.

1.3.2 Matplotlib

Matplotlib [6] is a comprehensive library for creating static, animated, and interactive visualizations in Python. It makes easy things easy and hard things possible. It creates publication-quality plots, and makes interactive figures that can zoom, pan, and update. It also customizes visual style and layout. It can be exported to many file formats. It is embedded in JupyterLab and Graphical User Interfaces. One can use a rich array of third-party packages built on Matplotlib.

1.3.3 Numpy

NumPy [7] is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

1.3.4 Pandas

Pandas [8] is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. Pandas is a fast and efficient DataFrame object for data manipulation with integrated indexing. It has tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format. It has intelligent data alignment and integrated handling of missing data. It gains automatic label-based alignment in computations and easily manipulates messy data into an orderly form. It gives flexible reshaping and pivoting of data sets. It has intelligent label-based slicing, fancy indexing, and subsetting of large data sets. It can insert and delete columns from data structures for size mutability. It aggregates or transforms data with a powerful group by engine allowing split-apply-combine operations on data sets. It has high performance in merging and joining data sets. Its hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure. Its benefits are time series functionality, date range generation and frequency conversion, moving window statistics, date shifting and lagging, creating domain-specific time offsets and joining time series without losing data. It is highly optimized for performance, with critical code paths written in Cython or C. Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

1.3.5 Tenserflow

TensorFlow [9] is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. It was developed by the Google Brain team for Google's internal use in research and production. Its languages are Python and C++. It offers multiple levels of abstraction. So you can choose the right one for your needs. It builds and trains models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. If one needs more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, one can use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

1.3.6 Open-CV

OpenCV [10] is known as cv2. cv2 (old interface in old OpenCV versions was named as cv) is the name that OpenCV developers chose when they created the binding generators. OpenCV is included as a submodule and the version is updated manually by maintainers when a new OpenCV release has been made. The cv2 module is the main module in OpenCV that provides developers with an easy-to-use interface for working with image and video processing functions. Using pre-built binaries is the quickest way to get a Python OpenCV environment up and running on Windows.

1.4 Chapter Summary

In this chapter, we provide insights into the feasibility of developing a fracture classification system. We also talk about the tools we use to create this system and what benefits we gain using these tools. We also talk about where we can get these tools and how we can use them to make our model well-fitted. We emphasize in this study what else can be done using these tools and how one can improve their classification system using these pre-trained models and tools.

2 Literature Review

A literature review is a critical examination and analysis of existing scholarly research, literature, and other sources related to a specific topic or research question. It is a key component of academic and research papers, theses, dissertations, and other scholarly works. The purpose of a literature review is to provide a comprehensive overview of the current state of knowledge on a particular subject, identify gaps or controversies in the existing literature, and highlight the significance of the proposed research. A well-conducted literature review is crucial for establishing the context and significance of a research project. It not only informs the reader about the existing body of knowledge but also helps the researcher position their work within the broader academic landscape.

2.1 KONet

KONet [11] is a robust detection method that utilizes a weighted ensemble approach to distinguish between normal and osteoporotic knee conditions, even when there are minor variations in the data. In this research, the researcher works with 7 pre-trained models and finds 2 models that are eligible for an ensemble approach. The model KONet got 97% accuracy ensembling 2 models: EfficientNetB0 and DenseNet121. KONet is the first ensemble-based approach for knee osteoporosis classification. In this research, InceptionV3 also used but its accuracy was only good for the training set and not for the testing and validation set. Whereas EfficientNetB0 and DenseNet121 were good enough in all sets. that's why, the researcher used these 2 models for the ensemble approach.

2.2 Recognition and Classification of Knee Osteoporosis and Osteoarthritis

The study [12] is regarding knee pain. In this study, there are 2 cases of knee pain. The first case is Knee Osteoarthritis which is the most common joint disorder based on Kellgren-Lawrence grading to distinguish from 0 to 4 levels of severity and the second case is Knee Osteoporosis which can progress without symptoms until a broken bone occurs. There have been several researches on knee pain. The previous researches show great performance on Knee Osteoarthritis but this research shows great performance on Knee Osteoporosis. for this research, the researcher used OA dataset consisting 372 images. In this research, the researcher works with 2models: VGG16 (Visual Geometry Group) and Late Fushion Model. For VGG16, the researcher got 82% accuracy. And for Late Fushion, the researcher got 77% accuracy. Though VGG16 got enough accuracy, but this model takes a total of 16 layers only.

2.3 Attention mechanism-based deep learning method

In this paper [13], a dataset, consisting of 4346 anteroposterior, lateral and oblique hand X-rays, is built from many orthopedic cases. Specifically, it contains a lot of hairline fractures. During wrist and finger fracture detection, hairline fractures in X-rays are difficult to detect with state-of-the-art methods in that small object detection is a challenging task for deep learning-based ways. So in this study, the researcher proposed GAN for manual windowing enhancement. In this research, the researcher works with ResNeXt and WristNet model and got ap 56.6% for their proposed model WristNet. Though wrist and finger fracture detection is challenging due to hairline, but still the result this model gets is not commendable.

2.4 Classification of Shoulder X-ray Images with Deep Learning Ensemble Models

This study [14] is regarding shoulder fracture. Fractures occur in the shoulder area for various reasons. To diagnose these fractures, data gathered from X-ray, MRI, and CT are used. This study aims to help physicians by classifying shoulder images taken from X-ray devices as fracture or non-fracture with artificial intelligence. In this study, several pre-trained models were used. The inceptionV3 model was also used in this study and got high accuracy for the training set but was inefficient in the testing set. Thus, after experiments, the researcher proposed 2 ensemble-based models: EL1 (consisting of ResNet50, denseNet169 and DenseNet201) and EL2 (consisting of ResNet34, DenseNet169 and Densenet201). The researcher got 2 accuracies for 2 ensemble models. EL1 got 84.55% and EL2 got 84.72%.

2.5 Dense Dilated Attentive Network for Automatic Classification

This study [15] is regarding the automatic classification of femur trochanteric fracture. It is very valuable in clinical diagnosis practice. Developing a high classification performance system is challenging due to the various locations, shapes, and contextual information of the fracture regions. To tackle this challenge, the researcher used DDA network in this study. The dataset used in this study has 3 classes. To handle multiclass, the researcher proposed DDA Network that uses DDA layer which aims to learn multiscale. Though the researcher used 2 instances in each batch whereas the dataset has 3 classes, still the AUC the proposed network got is 97%.

2.6 Fuzzy Rank-Based Ensemble Model

This study [16] is about the diagnosis of knee osteoporosis. In detecting diseases using X-ray images, it is problematic to manually examine X-rays for osteoporosis as well as to identify the essential components and choose elevated classifiers. To categorize x-ray pictures of knee joints into normal, osteopenia, and osteoporosis condition categories, authors present a process in this investigation that uses three convolutional neural networks (CNN) architectures, i.e., Inception v3, Xception, and ResNet 18, to create an ensemble-based classifier model. The suggested ensemble approach employs a fuzzy rank-based unification of classifiers. Though the researcher didn't take the standard ranking system's flaws into account, but still got 93.5% accuracy for their proposed ensemble model.

2.7 Classification of Atypical Femur Fracture with Deep Neural Networks

Atypical Femur Fracture (AFF) is a type of stress fracture that occurs in conjunction with prolonged bisphosphonate treatment. It is very rarely identified from Normal Femur Fracture (NFF) correctly on the first diagnostic X-ray examination. This study [17] aims at developing an algorithm based on deep neural networks to assist clinicians with the diagnosis of atypical femur fracture. In this study, the dataset that was used was consist of 796 images, which was divided into 2 classes: 397 for AFF and 399 for NFF. With the fully automatic diagnosis pipeline, the researcher achieved diagnosis accuracy of 82.7%, 89.4%, 90.5% with VGG19, InceptionV3 and ResNet50, respectively. With the interactive diagnostic pipeline, the diagnosis accuracy was improved to 92.2%, 93.4% and 94.4% respectively.

3 Methodology

The term methodology defines what things have been done to complete a research. This section contains choosing datasets, pre-processing datasets, working with pre-trained models, using attention modules, validating models, etc. This section also contains the experiments information that has been used to determine the model's efficiency.

For this research purpose, we have chosen an image dataset and resized its shape as per our needs. We then applied the data in pre-trained models to see which model provides the best result. After getting the efficient model, we find an attention module to customize our model and increase its efficiency. To validate the model's efficiency, we find two more datasets and implement our customized model on those datasets to find whether our model works well on those or not.

3.1 Dataset Description

The name of the dataset used in our research is FracAtlas [2]. It is an X-ray image dataset. The majority of the existing X-ray datasets are either too small in size to be used in training machine learning models or lack proper annotation to be used in localization and segmentation. This creates a limitation while comparing different state-of-the-art (SOTA) methods as each study uses its own set of data that is not made public. It also hinders the development of machine-learning algorithms for classification, localization and segmentation. But this dataset consists of 4,083 images which have been manually annotated for classification, localization and segmentation of bone fractures with the help of 2 expert radiologists and later validated by a medical officer. The data were collected from 3 major hospitals in Bangladesh. There are 717 images with 922 instances of fractures. Each of the fracture instances has its own mask and bounding box, whereas the scans also have global labels for classification tasks. The dataset is divided into 2 classes: one is Fractured and the other is Non-fractured. The dataset consists of images of different aged people's bones. The age of subjects in the dataset ranges from 8 months to

Table 3.1: Image Numbers In Each Class In The Dataset

Fractured	Non-fractured
717	3366

78 years old. Also, the gender distribution for abnormal studies is 85.4% and 14.6% between males and females respectively. The gender ratio for the whole dataset (normal + abnormal cases) is 62% male and 38% female approximately.

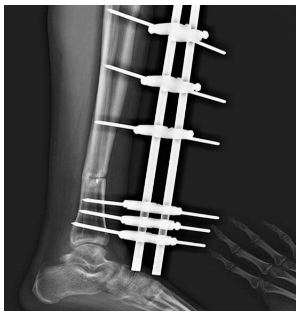
3.1.1 Dataset Source

The dataset was provided by Springer Science and Business Media LLC. It is a group work. The authors are Iftekharul Abedeen, Md. Ashiqur Rahman, Fatema Zohra Prottyasha, Tasnim Ahmed, Tareque Mohmud Chowdhury and Swakkhar Shatabda. The journal was published in ”Scientific Data” in August, 2023. And the dataset was downloaded from FigShare.

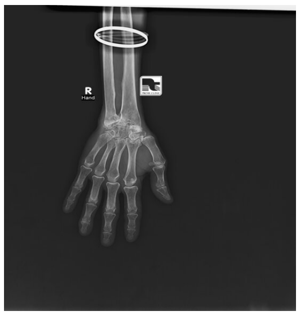
3.1.2 Images

The image of the dataset is divided into 2 classes. One of them is Fractured and the other one is Non-fractured. The total data collected at an early was amounted to 14,000 images. Then from this raw data, they isolated 4,083 images for future research. Of these images, 717 are Fractured images and 3366 are Non-fractured images. All of the images are in JPG format. There are 99 images with Orthopedic Fixation Devices (hardware) in them. The dataset has a total of 1,538 scans of the hand and among them, 437 are fractured. There is a total of 2,272 leg and hand scans, 338 hip scans and 349 shoulder scans. Among these, the number of scans belonging fractured class is 263, 63 and 63 for the leg and hand, hip and shoulder regions respectively. The dataset comprises a total of 2,503 frontal, 1,492 lateral, and 418 oblique view images, each pertaining to different organs. Whereas the ‘Fractured’ class includes 438 frontal, 325 lateral, and 45 oblique view images. Conversely, the ‘Non-fractured’ class encompasses a total of 2,065 frontal, 1,167 lateral, and 373 oblique views.

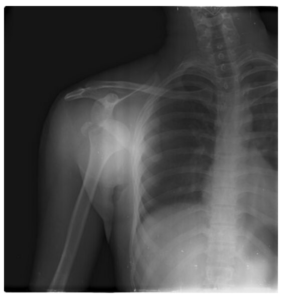
From the fractured images, three images of leg, hand and shoulder having fracture in different portion are given below to show fracture in these parts of body:



(a) Image showing fracture in a portion of leg



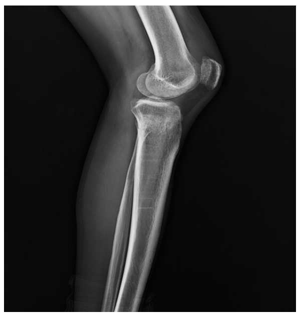
(b) Image showing fracture in a portion of hand



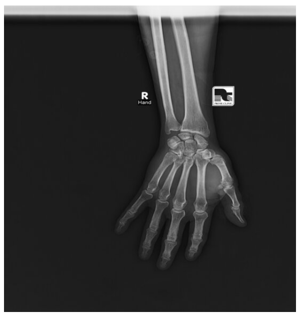
(c) Image showing fracture in a portion of shoulder

Figure 3.1: Images showing fracture in different parts of body

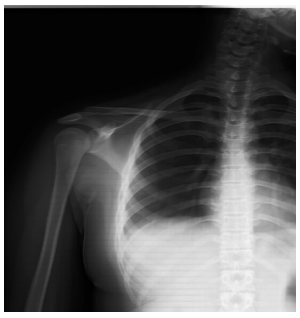
From the non-fractured images, three images of leg, hand and shoulder having no fracture are given below to show normal form of these parts of body:



(a) Image showing no fracture in leg



(b) Image showing no fracture in hand



(c) Image showing no fracture in shoulder

Figure 3.2: Image showing no fracture in different parts of body

3.1.3 Image Resolution

Image resolution is the term used to describe the number of pixels in an image and is typically stated as the image's width times height, both in pixels. The degree of clarity and detail in an image depends on its resolution. More pixels mean more detail and clarity, and higher resolution means more pixels. In this dataset, the resolution of the images are all same. The images are all in the same shape. For this dataset, the resolution is 2304×2880.

3.2 Dataset preprocessing

Dataset preprocessing is a crucial step in the data science and machine learning pipeline. It involves cleaning, transforming, and organizing raw data into a format suitable for analysis and model training. Dataset preprocessing for images entails cleaning up and converting raw image data into a format appropriate for computer vision tasks. The steps we had taken for preprocessing are given below:

3.2.1 Data splitting

Data splitting is a crucial step in the machine learning workflow, where you divide your dataset into different subsets to train, validate, and test your model. The typical splits include a training set, a validation set, and a test set. To split our dataset into these 3 sets, first we import the cv2 module and split our dataset using its functions. First we create 3 directories for these 3 sets and split our dataset into those 3 directories. The dataset provider already splits the fracture images into 3 sets. So we just split the non-fractured images into 3 sets. Following splits of fracture images, we split 80% of non-fractured images into training set, 8.5% of non-fracture images into test set and 11.5% of non-fracture images into validation set.

Table 3.2: Split Of Fractured And Non-fractured Classes Into 3 Sets

Training Set	Validation Set	Test Set
80%	11.5%	8.5%

3.2.2 Corrupted Image Removing

When working with large datasets for training deep learning models, it's not uncommon to encounter corrupted or unreadable image files. Feeding such images to the model can cause training to halt unexpectedly, leading to wasted time and resources. Therefore, it's essential to validate the integrity of your dataset before training. While using our dataset in codes, we found that there were corrupted images in the dataset which were causing the training to halt unexpectedly. Then we preprocessed the dataset and removed the corrupted images from both fractured and non-fractured images from the previously splitted 3 sets.

3.2.3 Image resizing

Image resizing is a common preprocessing step in computer vision tasks, especially when dealing with image datasets of varying dimensions. Resizing images helps ensure uniformity and can be essential for compatibility with neural network architectures. In our research, we have resized the image shapes from 2304×2880 to 224×224. In deep learning, the 224×224 picture size is most frequently used as an input size for Convolutional Neural Network (CNN) models. Using this image size has the following advantages:

- Computational Effectiveness: Training and inference are faster when we use smaller size images as we need fewer computations. This is crucial when we work with huge models and scarce computational resources.

- Improved Generalization: Choosing smaller size images can result in improved generalization as the model is compelled to concentrate on the most crucial aspects of the input data instead of becoming overburdened by irrelevant information.

- Compatibility with Pre-trained Models: As we are using several well-known pre-trained models, including InceptionV3, EfficientNetV2B2, DenseNet121, EfficientNetB1 and InceptionV4, were developed using this size as their training data. Utilizing this size as an input size makes it simpler to apply these models to transfer learning.

- Standardization: It makes it simpler to compare results from various models and datasets when they are all inputted with the same size. This makes it possible to evaluate various algorithms more consistently and to compare their effectiveness directly.

3.3 Implementation of pre-trained models

Implementing pre-trained models involves using a model that has already been trained on a large dataset for a specific task, such as image classification or natural language processing, and then adapting it to your own specific task. This is commonly referred to as transfer learning.

3.3.1 InceptionV3

Inception-v3 [18] is a pre-trained convolutional neural network that is 48 layers deep, which is a version of the network already trained on more than a million images from the ImageNet database. This pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299. The model extracts general features from input images in the first part and classifies them based on those features in the second part.

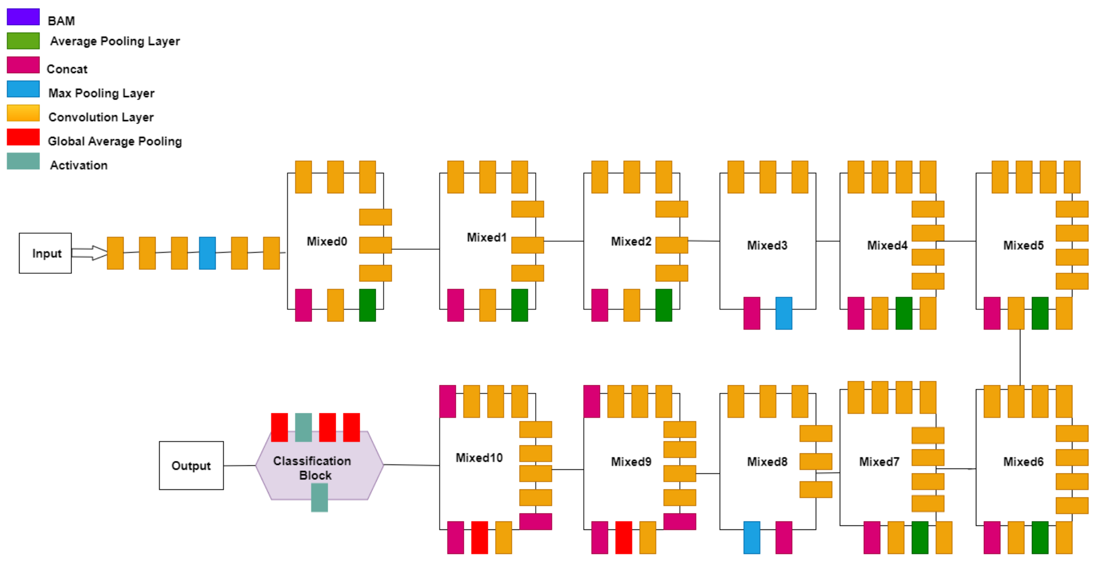


Figure 3.3: InceptionV3 Architecture

3.3.2 EfficientNetV2B2

EfficientNetV2B2 [19] is a type of convolutional neural network that has faster training speed and better parameter efficiency. EfficientNetV2 goes one step further than EfficientNet to increase training speed and parameter efficiency. This network is generated by using a combination of scaling (width, depth, resolution) and neural architecture search. The main goal is to optimize training speed and parameter efficiency.

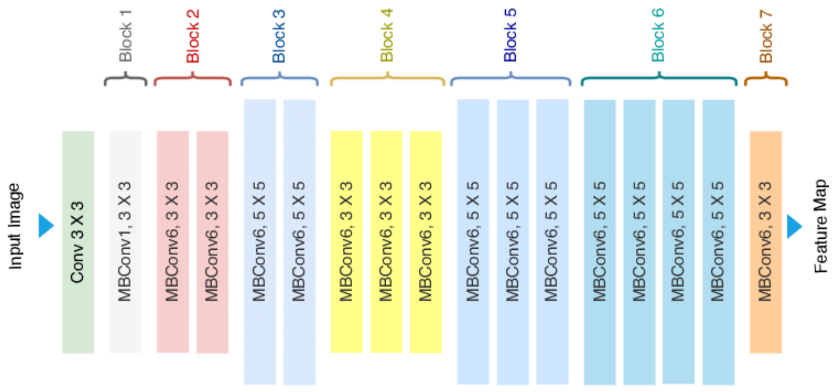


Figure 3.4: EfficientNetV2B2 Architecture

3.3.3 DenseNet121

The DenseNet121 [20] model is one of the DenseNet group of models designed to perform image classification. The authors originally trained the models on Torch, but then converted them into Caffe format. All DenseNet models have been pre-trained on the ImageNet image database. The Dense Convolutional Network (DenseNet) connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections - one between each layer and its subsequent layer - our network has L(L+1)/2 direct connections. For each layer, the feature maps of all preceding layers are used as inputs, and its own feature maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. We evaluate our proposed architecture on four highly competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet). DenseNets obtain significant improvements over the state-of-the-art on most of them, whilst requiring less memory and computation to achieve high performance.

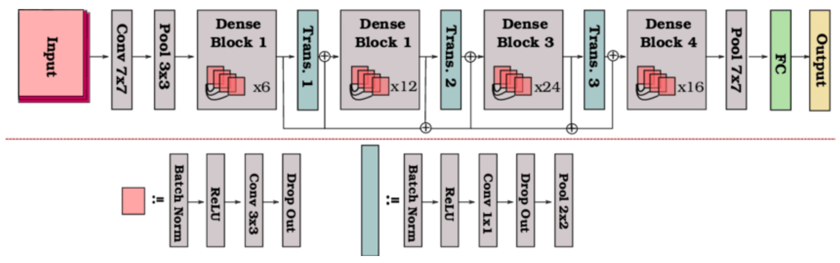


Figure 3.5: DenseNet121 Architecture

3.3.4 EfficientNetB1

EfficientNetB1 model [21] trained on ImageNet-1k at resolution 240x240. It was introduced in the paper EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks by Mingxing Tan and Quoc V. Le, and first released in this repository. The keras efficientnet function will return the model of keras image classification that has been pre-trained with imagenet and optionally loaded with weights. It is a model that achieves state-of-the-art accuracy on common image classification and imagenet for transfer learning tasks.

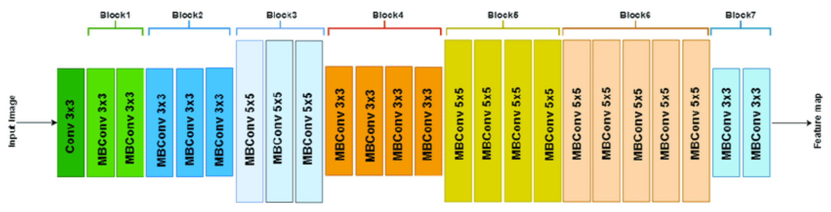


Figure 3.6: EfficientNetB1 Architecture

3.3.5 Inception-v4

Inception-v4 [22] is a convolutional neural network architecture that builds on previous iterations of the Inception family by simplifying the architecture and using more inception modules than InceptionV3. Inception-v4, introduced in 2016, further improved the Inception architecture by introducing residual connections and ”grid size reduction” modules. The residual connections allowed the network to learn residual mappings, which helped improve the network’s accuracy and convergence speed.

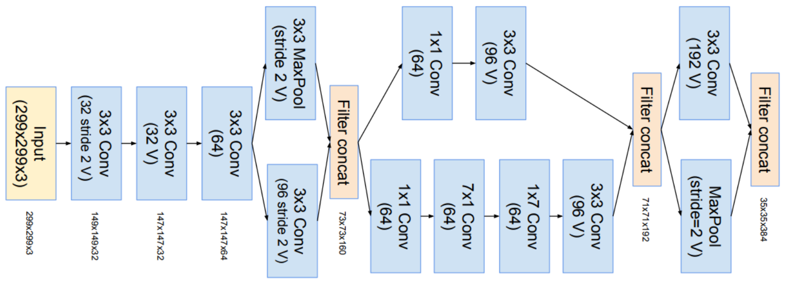


Figure 3.7: Inception-v4 Architecture

3.4 *Proposed Methodoloy*

The model we have chosen to use attention module is InceptionV3 as we have already gained a good result using the pre-trained model. So we have done various experiments on model and added our chosen attention module to different layers of the model. After doing several experiments, we chose 2 experiments that have the same result and that enhance our model's accuracy. In one of the experiments, we have added 3 attention modules that have fully connected layers. And in another experiment, we have added 2 attention modules that have fully connected layers.

3.4.1 Fully Connected layers

In our 2 experiments, we have added attention modules that have fully connected layers.

In our first experiment, we have added 3 attention modules. The first one is added after "mixed 2" layer and it connects with 288 channels of the previous layer. The second one is added after "mixed 6" and it connects with 768 channels of the previous layer. And the last one is added after "mixed 10" and it connects with 2048 channels of the previous layer.

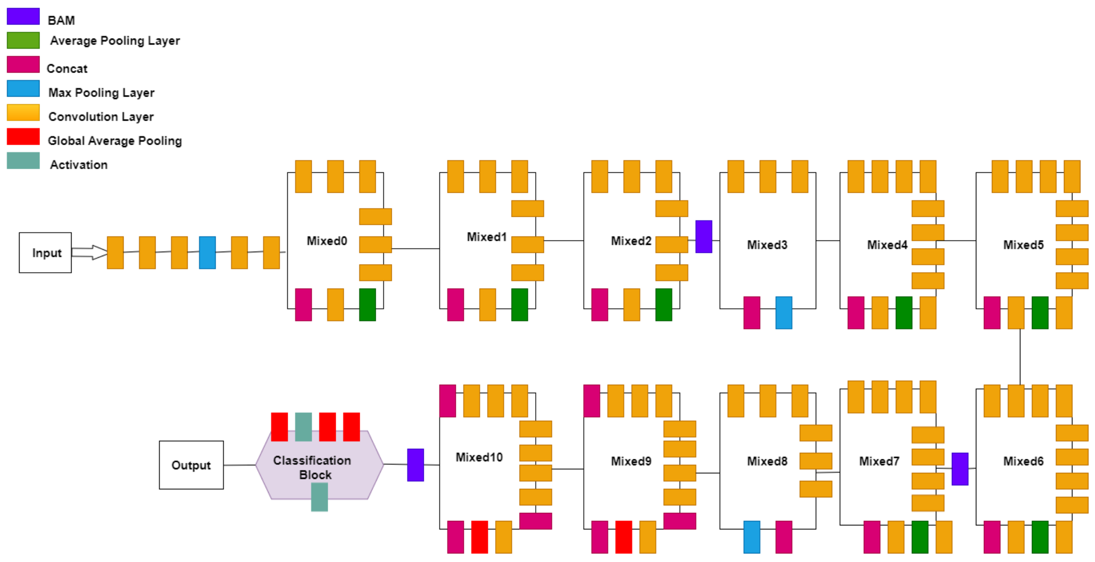


Figure 3.8: First experiment of inceptionV3 model after adding BAM

In our second experiment, we have added 2 attention modules. The first one is added after ”mixed 4” and it connects with 768 channels of the previous layer. And the last one is added after the average pooling of ”mixed 10” and it connects with 2048 channels of the previous layer.

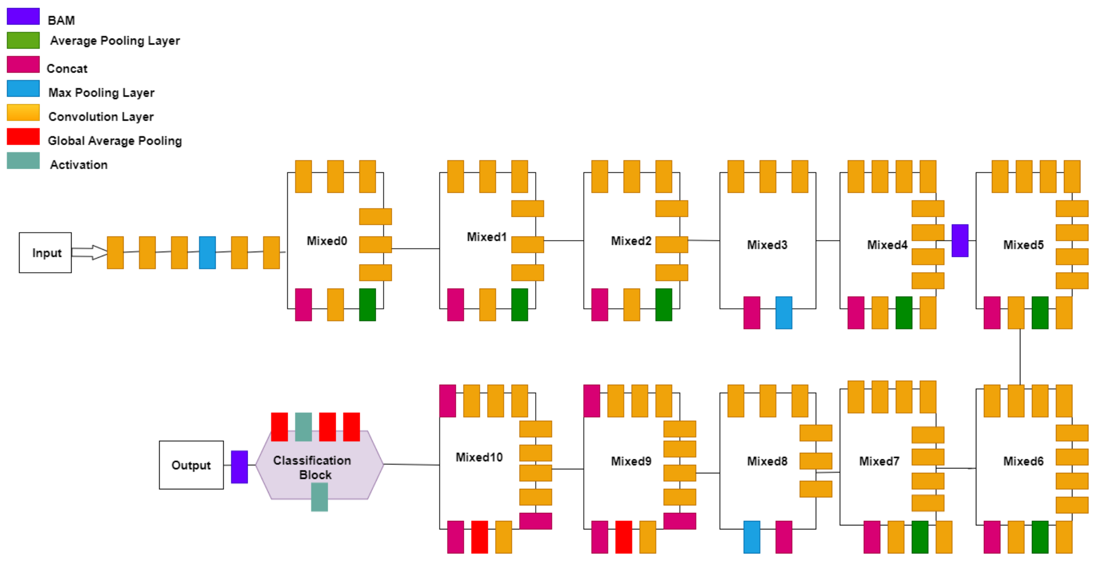


Figure 3.9: Second experiment of inceptionV3 model after adding BAM

3.4.2 BAM attention module

We have chosen BAM as our attention module and added it in our experiments. After adding BAM in our model, it enhances our model's accuracy and also improves the results of other metrics.

Bottleneck Attention Module (BAM) [23] is introduced by Korea Advanced Institute of Science and Technology (KAIST). BAM is a new module that is designed to be integrated with any feed-forward CNNs. This module infers an attention map along two separate pathways, channel and spatial. It is placed at each bottleneck of models where the downsampling of feature maps occurs. BAM constructs a hierarchical attention at bottlenecks with a number of parameters and it is trainable in an end-to-end manner jointly with any feed-forward models. On the CIFAR-100 and ImageNet classification tasks, authors observe performance improvements over baseline networks by placing BAM. The improvement can be observed when multiple BAM modules located at different bottlenecks build a hierarchical attention.

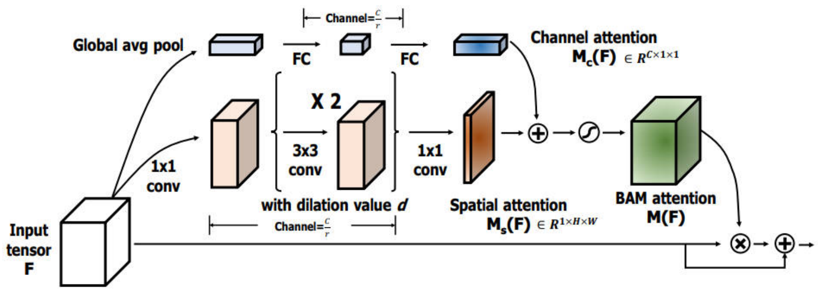


Figure 3.10: BAM Architecture

3.5 Evaluation metrics

Evaluation metrics are quantitative measures used to assess the performance and effectiveness of a statistical or machine learning model. These metrics provide insights into how well the model is performing and help in comparing different models or algorithms. For our research, we have used 4 metrics as our evaluation metric. We have used Precision, Recall, F1-Score and Accuracy as our evaluation metric. We have calculated TP, TS, FP, TN and FN to measure our model's result. The abbreviations TP, TS, FP, TN, and FN stand for true positive, total samples, false positive, true negative, and false negative respectively. Here is how we calculated these metrics:

3.5.1 Accuracy

One parameter for assessing our classification models is accuracy. The ratio of correctly predicted instances to the total instances is Accuracy. The percentage of predictions that our model correctly predicted is known as accuracy. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{TN + TP}{TP + FP + TN + FN} \tag{3.1}$$

3.5.2 Precision

One parameter for assessing our classification models is precision. Precision is determined by dividing the total number of positively identified samples by the proportion of accurately classified positive samples. It is the proportion of true positive predictions among all positive predictions. Formally, precision has the following definition:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

3.5.3 Recall

One parameter for assessing our classification models is recall. Recall is the proportion of true positive predictions among all actual positives. The model's capacity to identify positive samples is gauged by the recall. More positive samples are found when a recall is higher. Formally, recall has the following definition:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

3.5.4 F1-score

One parameter for assessing our classification models is f1-score. A statistic used to assess a machine learning model's performance is called the F-score, also known as the F1 score or F-measure. It generates a single score that incorporates recall and precision. It is the harmonic mean of precision and recall. It balances precision and recall. Formally, f1-score has the following definition:

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

4 Result Analysis and Discussion

In this chapter, we will discuss about all pre-trained models test accuracy, test precision, test recall and test f1-score. We will also discuss why we chose our model and how much better result we get after applying our attention module. We will also discuss about how our model affects any other dataset for validation and how much it works better after applying BAM.

4.1 Results obtained

In this section, we have added all the results that we have gathered from implementing pre-trained models. We have also added what accuracy we get after adding the attention module. And how much the experiments affect other datasets.

4.1.1 Result after implementing models

The comparison of results was done after doing several experiments using imagenet and without using imagenet. The results from the models are shown in the following tables to see which model has the better result in which section. Here:

- Test Accuracy: TA
- Test F1 score: TF1
- Test Recall: TR
- Test precision: TP

The first table contains results of models using imagenet:

Table 4.1: Comparison Of Results Of Models Using Imagenet

Model	TA	TF1	TR	TP
InceptionV3	0.9193	0.9201	0.9201	0.9201
EfficientNetV2B2	0.9049	0.9083	0.9083	0.9083
DenseNet121	0.9135	0.9145	0.9145	0.9145
Inception-v4	0.8904	0.8901	0.8901	0.8901
EfficientNetB1	0.8991	0.8985	0.8985	0.8985

The second table contains results of models without using imagenet:

Table 4.2: Comparison Of Results Of Models Without Using Imagenet

Model	TA	TF1	TR	TP
InceptionV3	0.8905	0.8923	0.8923	0.8923
EfficientNetV2B2	0.8646	0.8652	0.8652	0.8652
DenseNet121	0.8905	0.8902	0.8902	0.8902
Inception-v4	0.8731	0.8735	0.8735	0.8735
EfficientNetB1	0.8472	0.8485	0.8485	0.8485

From the above 2 tables, we can see that InceptionV3 was the best suitable for our dataset as in both times using imagenet and without using imagenet, the models provided the maximum test accuracy, test precision, test recall and test f1-score.

After finding our suitable model, we need to add BAM in some of the layers of the model to upgrade the model's efficiency. So, we have added BAM attention module in some of its layers. The following table shows the result of our experiments:

Table 4.3: Results Of 2 Experiments Of InceptionV3 After Adding BAM

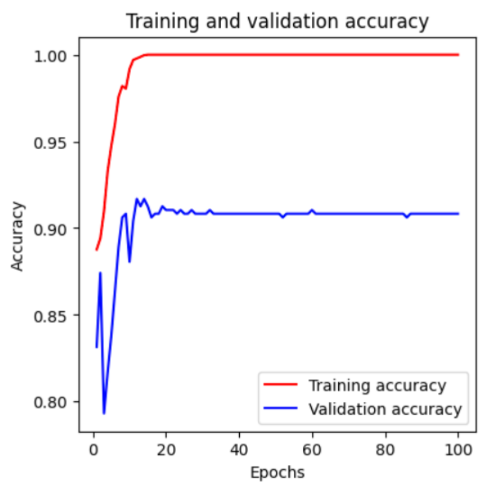
Experiment No.	TA	TF1	TR	TP
Experiment1	0.9020	0.9034	0.9034	0.9034
Experiment2	0.9020	0.9022	0.9022	0.9022

From the above table, we can see that after adding BAM as attention module, our model's test accuracy, test precision, test recall and test f1-score got better than previous results.

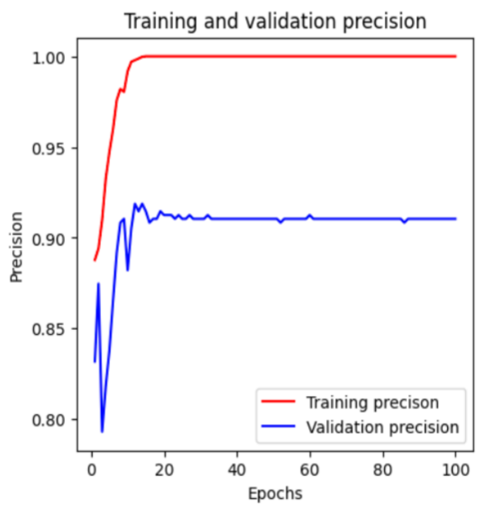
4.2 Graphs showing results of various evaluation metrics

In this section, we are providing the graphs of pre-trained models showing the results. We are also providing graphs showing the results of InceptionV3 after adding BAM attention module.

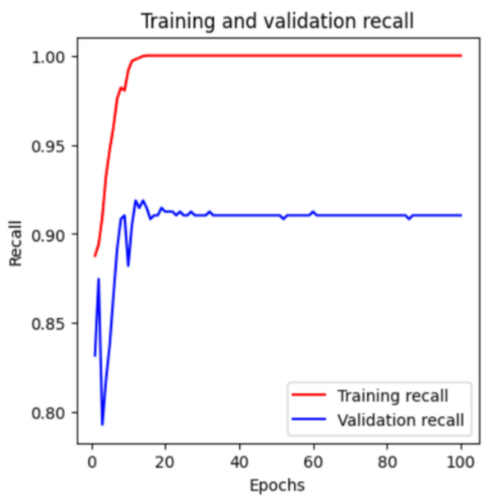
The graphs given below show the results of pre-trained model InceptionV3 using imagenet:



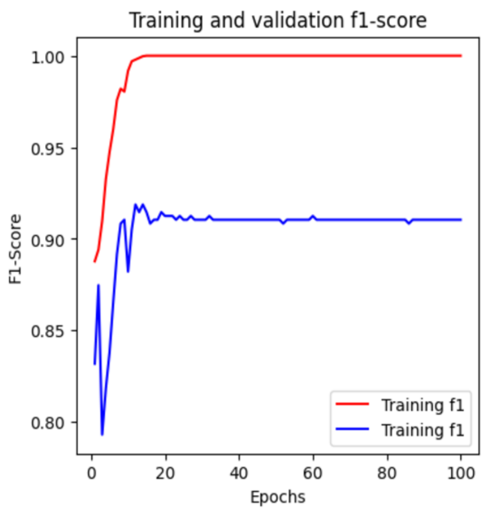
(a) Accuracy of InceptionV3 model using imagenet



(b) Precision of InceptionV3 model using imagenet



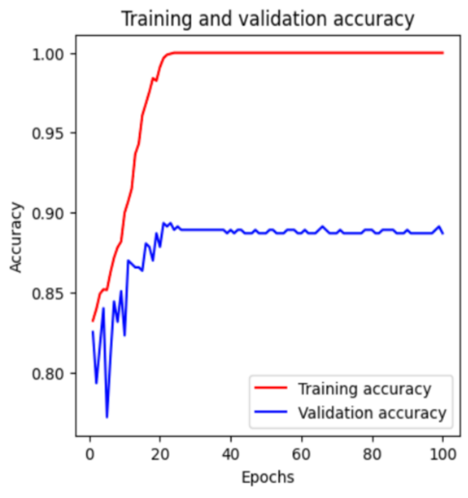
(c) Recall of InceptionV3 model using imagenet



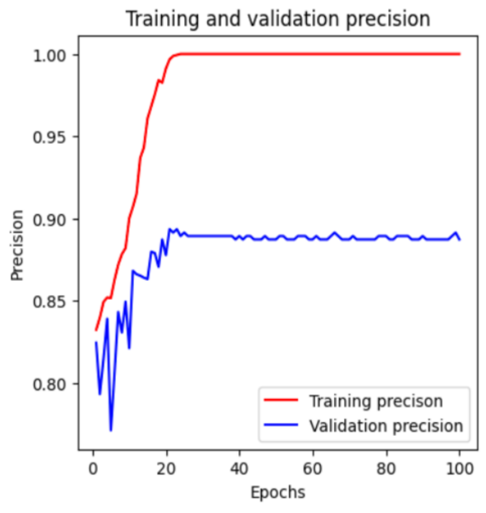
(d) F1-score of InceptionV3 model using imagenet

Figure 4.1: Results of pre-trained model InceptionV3 using imagenet

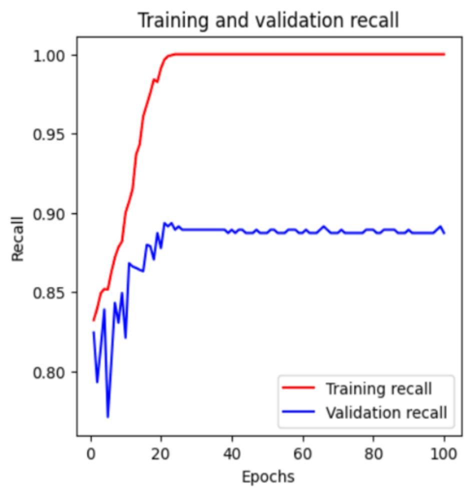
The graphs given below show the results of pre-trained model InceptionV3 without using imagenet:



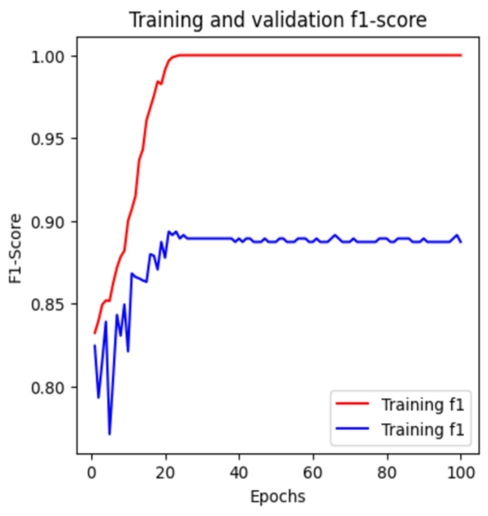
(a) Accuracy of InceptionV3 model without using imagenet



(b) Precision of InceptionV3 model without using imagenet



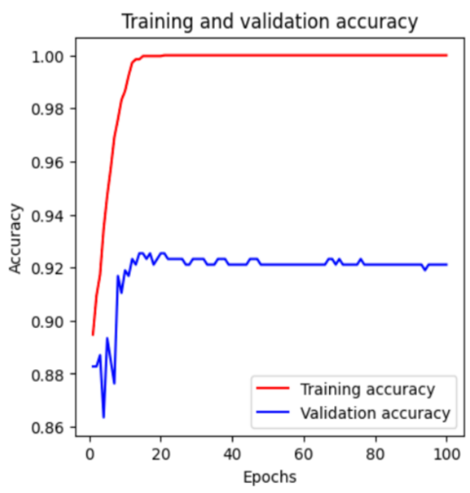
(c) Recall of InceptionV3 model without using imagenet



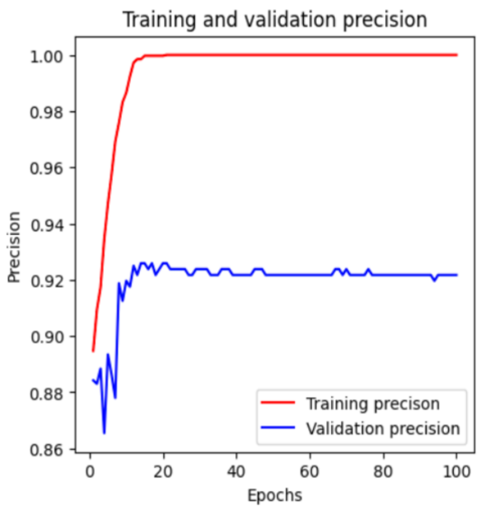
(d) F1-score of InceptionV3 model without using imagenet

Figure 4.2: Results of pre-trained model InceptionV3 without using imagenet

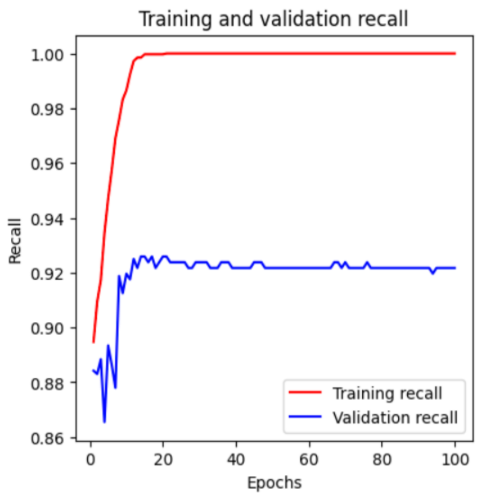
The graphs given below show the results of pre-trained model DenseNet121 using imagenet:



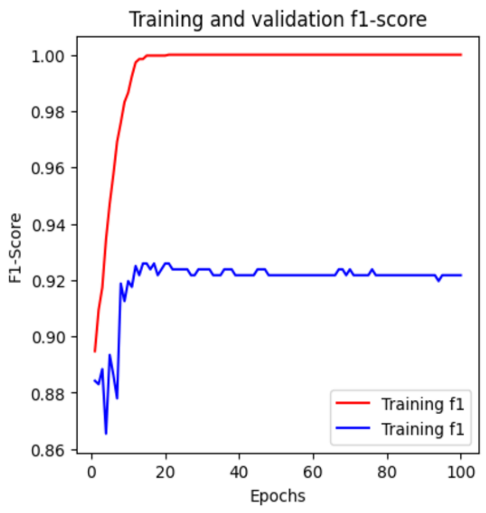
(a) Accuracy of DenseNet121 model using imagenet



(b) Precision of DenseNet121 model using imagenet



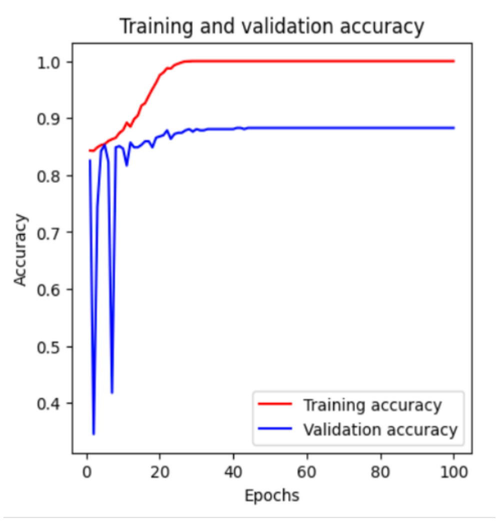
(c) Recall of DenseNet121 model using imagenet



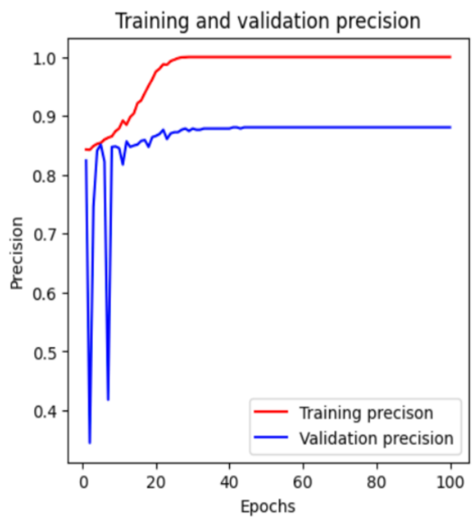
(d) F1-score of DenseNet121 model using imagenet

Figure 4.3: Results of pre-trained model DenseNet121 using imagenet

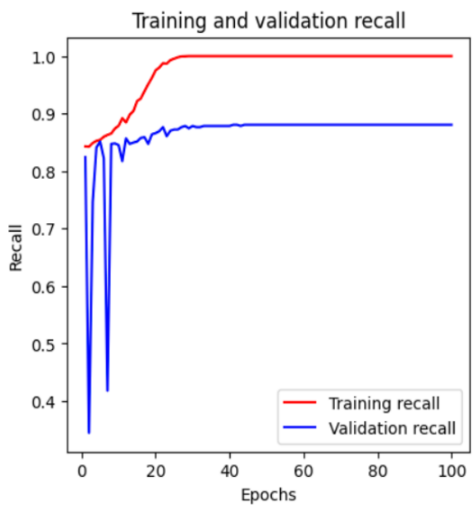
The graphs given below show the results of pre-trained model DenseNet121 without using imagenet:



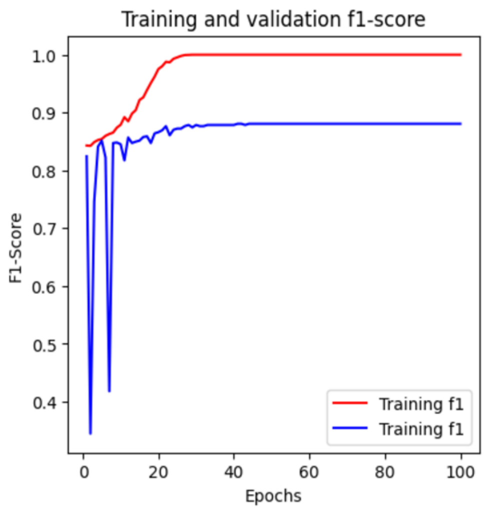
(a) Accuracy of DenseNet121 model without using imagenet



(b) Precision of DenseNet121 model without using imagenet



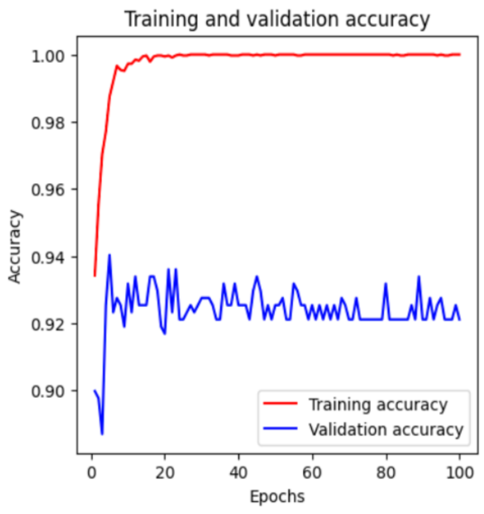
(c) Recall of DenseNet121 model without using imagenet



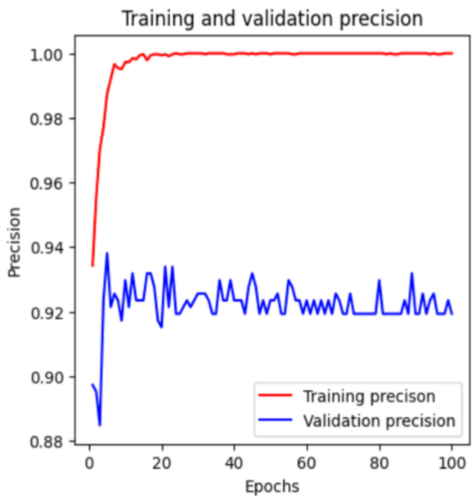
(d) F1-score of DenseNet121 model without using imagenet

Figure 4.4: Results of pre-trained model DenseNet121 without using imagenet

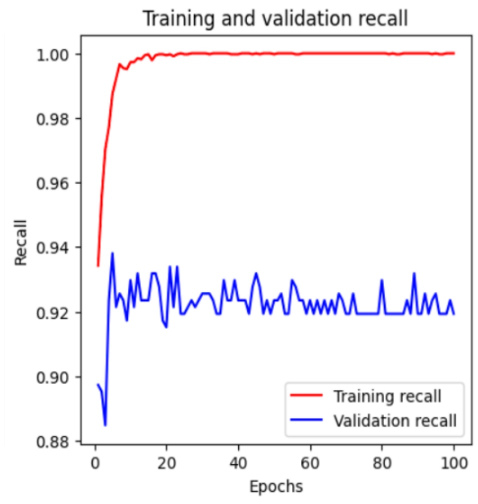
The graphs given below show the results of pre-trained model EfficientNetV2B2 using imagenet:



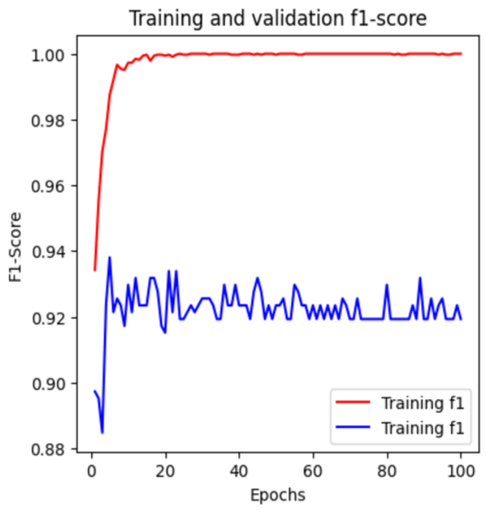
(a) Accuracy of EfficientNetV2B2 model using imagenet



(b) Precision of EfficientNetV2B2 model using imagenet



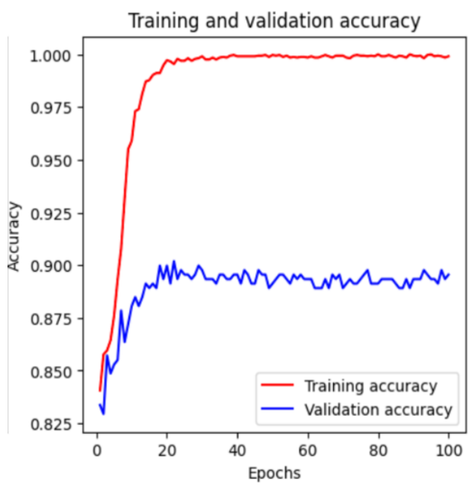
(c) Recall of EfficientNetV2B2 model using imagenet



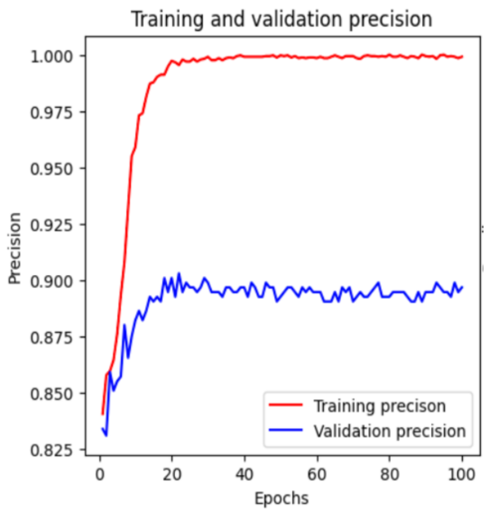
(d) F1-score of EfficientNetV2B2 model using imagenet

Figure 4.5: Results of pre-trained model EfficientNetV2B2 using imagenet

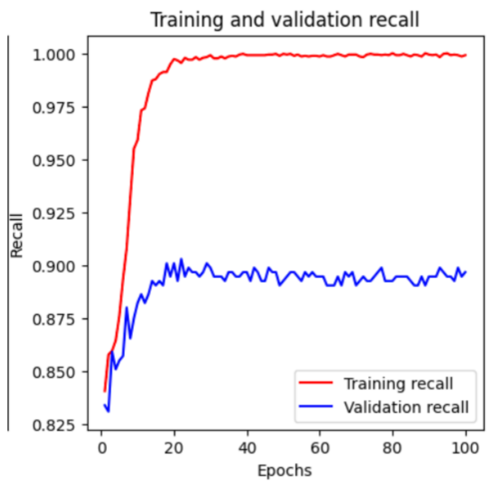
The graphs given below show the results of pre-trained model EfficientNetV2B2 without using imagenet:



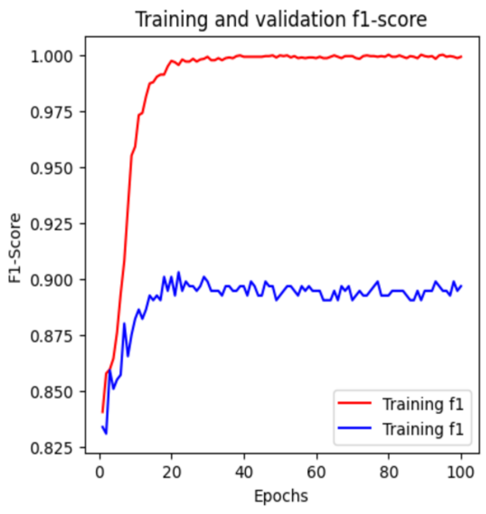
(a) Accuracy of EfficientNetV2B2 model without using imagenet



(b) Precision of EfficientNetV2B2 model without using imagenet



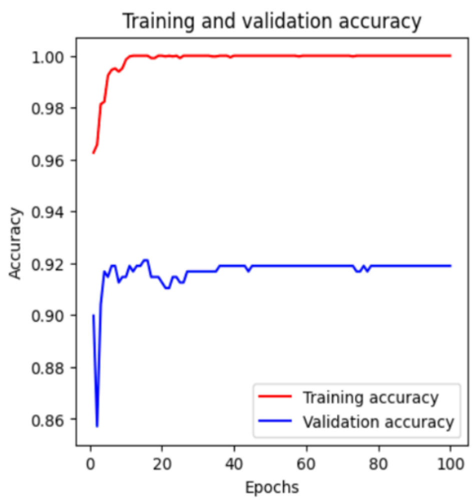
(c) Recall of EfficientNetV2B2 model without using imagenet



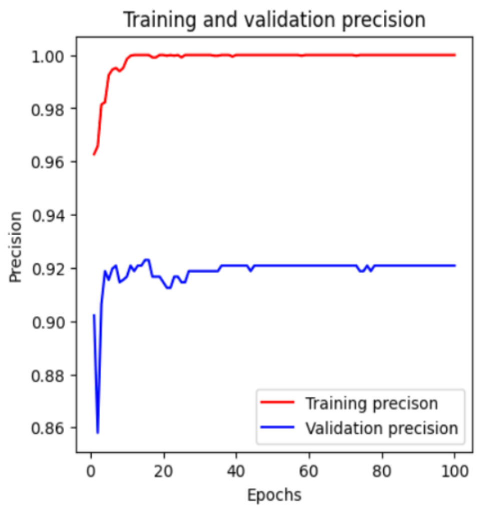
(d) F1-score of EfficientNetV2B2 model without using imagenet

Figure 4.6: Results of pre-trained model EfficientNetV2B2 without using imagenet

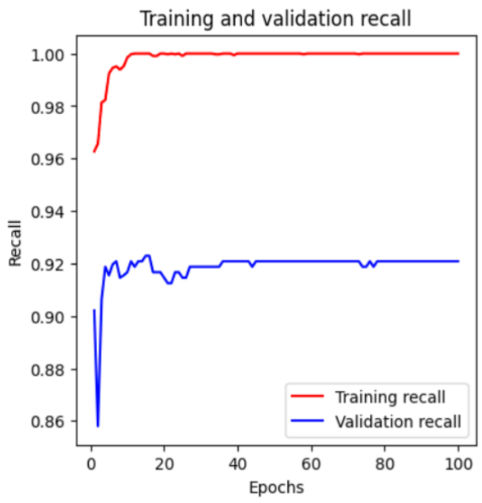
The graphs given below show the results of pre-trained model EfficientNetB1 using imagenet:



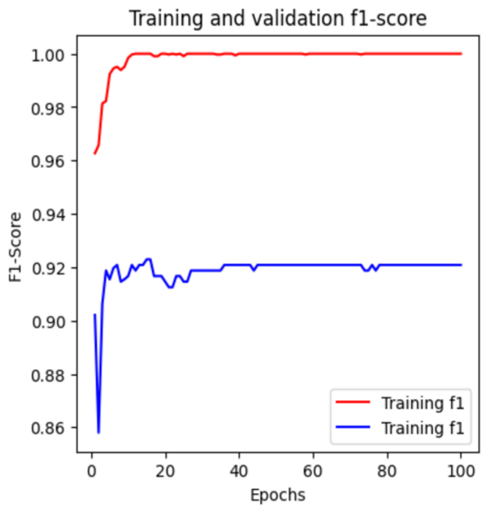
(a) Accuracy of EfficientNetB1 model using imagenet



(b) Precision of EfficientNetB1 model using imagenet



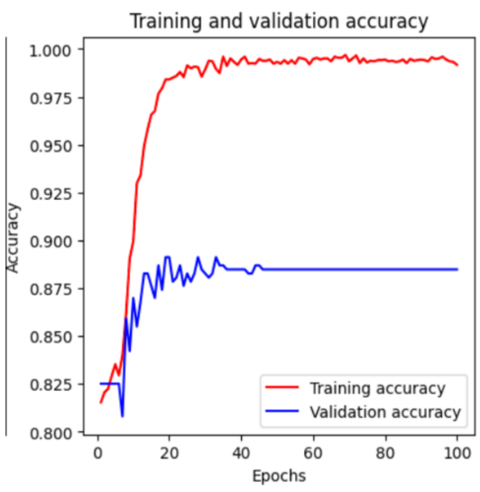
(c) Recall of EfficientNetB1 model using imagenet



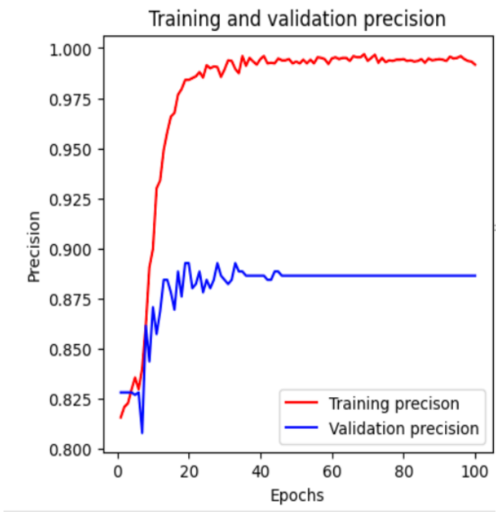
(d) F1-score of EfficientNetB1 model using imagenet

Figure 4.7: Results of pre-trained model EfficientNetB1 using imagenet

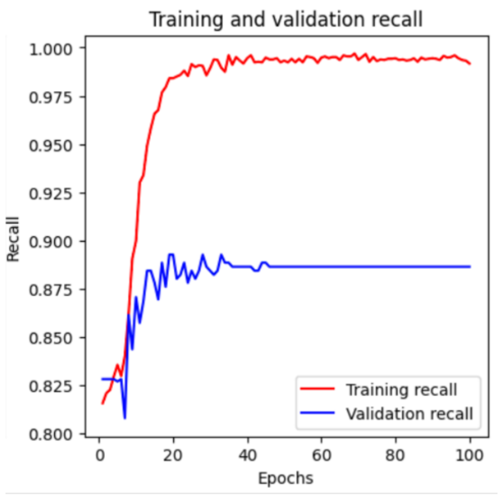
The graphs given below show the results of pre-trained model EfficientNetB1 without using imagenet:



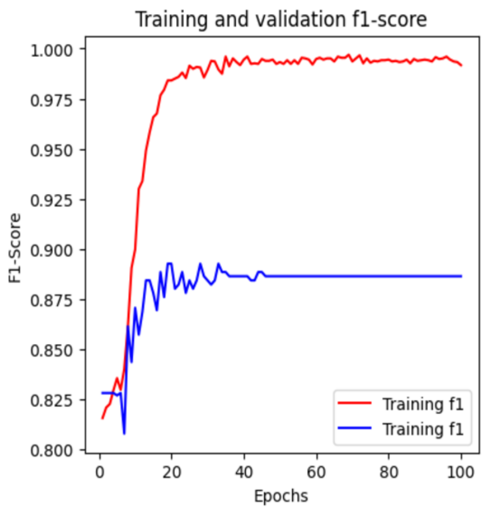
(a) Accuracy of EfficientNetB1 model without using imagenet



(b) Precision of EfficientNetB1 model without using imagenet



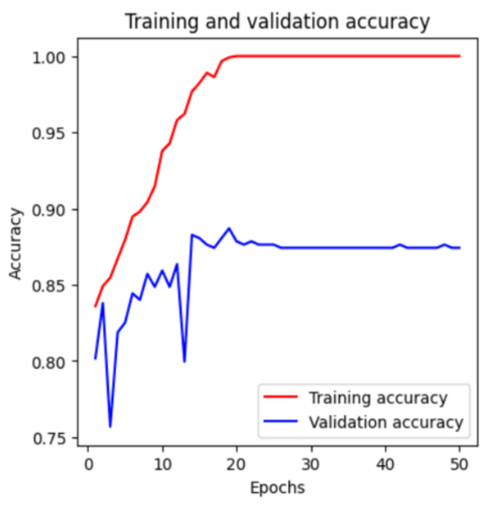
(c) Recall of EfficientNetB1 model without using imagenet



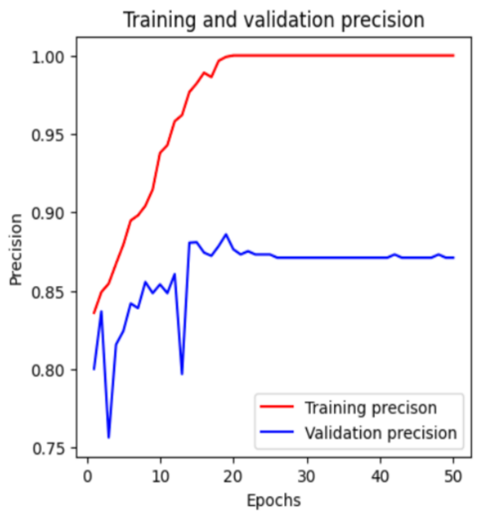
(d) F1-score of EfficientNetB1 model without using imagenet

Figure 4.8: Results of pre-trained model EfficientNetB1 without using imagenet

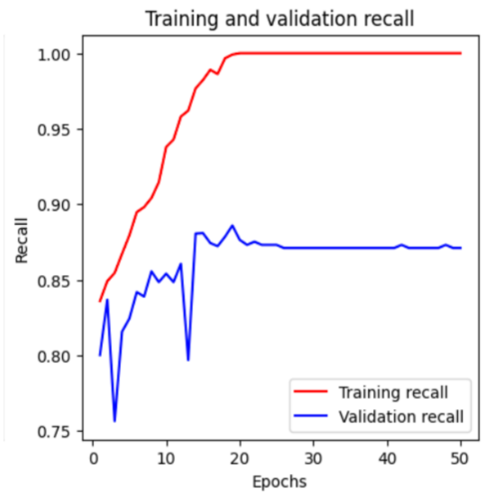
The graphs given below show the results of experiments after adding BAM:



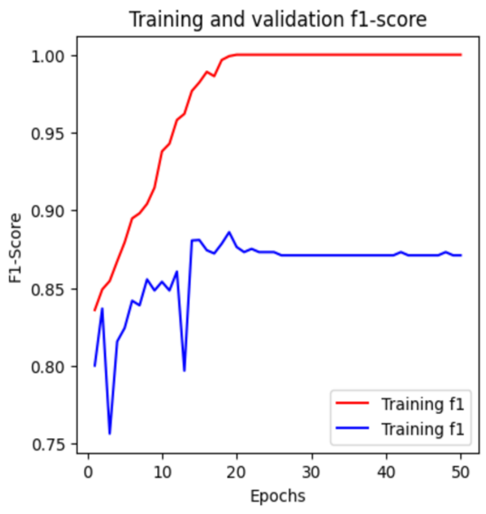
(a) Accuracy of Experiment1 model without using imagenet



(b) Precision of Experiment1 model without using imagenet

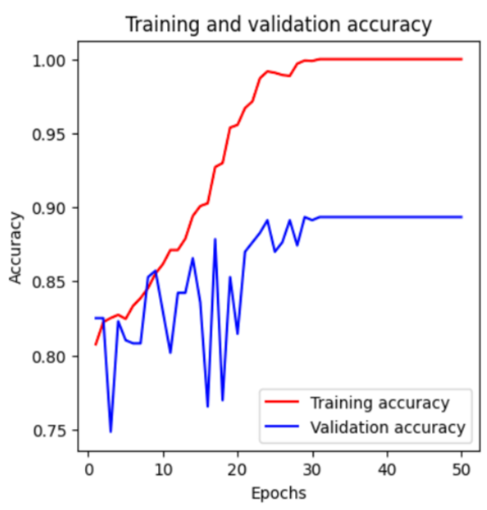


(c) Recall of Experiment1 model without using imagenet

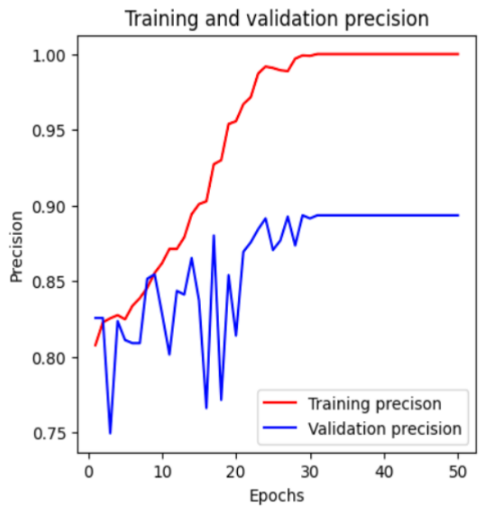


(d) F1-score of Experiment1 model without using imagenet

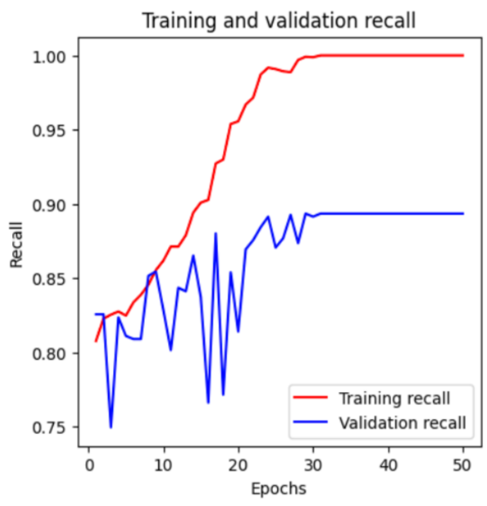
Figure 4.9: Results of pre-trained model Experiment1 without using imagenet



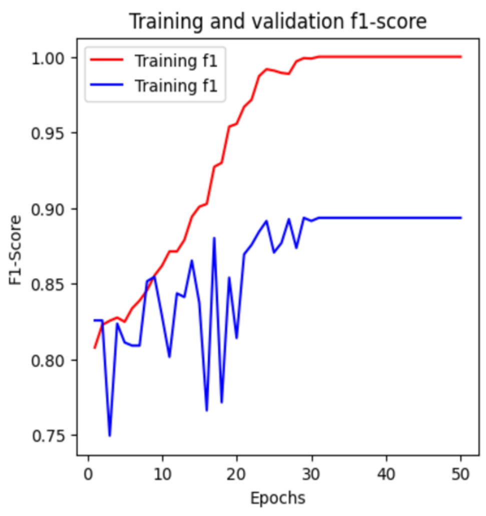
(a) Accuracy of Experiment2 model without using imagenet



(b) Precision of Experiment2 model without using imagenet



(c) Recall of Experiment2 model without using imagenet



(d) F1-score of Experiment2 model without using imagenet

Figure 4.10: Results of pre-trained model Experiment2 without using imagenet

4.3 Findings

From the tables and figures provided above, we can see that InceptionV3 provides us with better results than the other models and that is 89.04%. We can see from the table that even DenseNet121 provides us the same accuracy but but it is less efficient in terms of precision, recall and f1-score. InceptionV3 has higher percentage in terms of these evaluating metrics. So we chose InceptionV3 as our model and add BAM as attention module in it. After adding attention module BAM inside its layers, it provides us more accuracy and that is 90.2% and also increases the other three evaluating metrics..

4.4 Chapter Summary

In this chapter, we compare models results and show their accuracy. We also see that InceptionV3 provides better accuracy than other models whether using imagenet or without using imagenet. We also show in this chapter that all the evaluation metrics get better than previous results after adding BAM.

5 Conclusion

Fracture detection is one of the most challenging problems of bio-medical problems as it directly impacts human life. Fractures in bones can make one's life challenging and problematic and also cause many hazards in life. So detecting these fractures and identifying their area is always important.

In this study, we use FracAtlas [2] dataset containing a significant number of images. This dataset contains fracture images of different areas and also it shows the normal form of that area. This dataset is compatible with classification, segmentation and localization tasks. This study is about classification tasks.

For the classification task, first, we run some pre-trained models and find out that InceptionV3 is more suitable than other models. Then we find an attention module named BAM that increases the accuracy of our model. But, the model still has some limitations. There is still a lot of scope to do better work on it. We have achieved enough accuracy on the test set, but couldn't reach the pre-trained model's accuracy level using imagenet. However, when compared with other models and the accuracy level of InceptionV3 without using imagenet, we can say that we get a good result.

5.1 Limitations

Every research work requires difficulties and even after all these hardships, the research might have limitations. Even we faced a lot of difficulties while working on this paper. We have tried to solve the problems and give our best. But still, there are some limitations that our paper couldn't resolve. The limitations are given below:

- We find out that our model is over-fitted in some cases. Though it classified fractures on the test set and provided a pretty good result, but still there was seen an over-fitting tendency in some cases.

- Our model couldn't beat the result of pre-trained InceptionV3 model using imagenet. Though after adding BAM, our model's accuracy got better compared to this model's previous accuracy without using imagenet. But still, we couldn't achieve the model's accuracy that it got by using imagenet which was 91.93%

- Our model is acceptable for binary class only. The dataset we worked with was a binary dataset. The dataset was divided into only 2 classes. Thus, our model was created only to classify binary classes. And it is not competent to work with multi-classes.

5.2 Future Works

As our model has some limitations. So in the future, we would like to solve these issues and upgrade our model.

- Our future goal is to make our model better and handle its over-fitting problem.

- We want to increase its accuracy and beat the result that this model got by using imagenet.

- We would like to create a model that can classify multi-class datasets and can classify fractures according to their region.

- We also want to develop a software that is incorporated with this model which can take X-ray images as input and can classify the images as fractured or non-fractured considering the input.

References

[1] F. Borgström, L. Karlsson, G. Ortsäter, N. Norton, P. Halbout, C. Cooper, M. Lorentzon, E. V. McCloskey, N. C. Harvey, M. K. Javaid, and J. A. Kanis, "Fragility fractures in europe: burden, management and opportunities," *Archives of Osteoporosis*, vol. 15, p. 59, 12 2020.

[2] I. Abedeen, M. A. Rahman, F. Z. Prottyasha, T. Ahmed, T. M. Chowdhury, and S. Shatabda, "Fracatlas: A dataset for fracture classification, localization and segmentation of musculoskeletal radiographs," *Scientific Data*, vol. 10, no. 1, p. 521, 2023.

[3] "Bone fracture." [Online]. Available: https://www.kaggle.com/datasets/ahmedashrafahmed/bone-fracture

[4] P. Rajpurkar, J. Irvin, A. Bagul, D. Ding, T. Duan, H. Mehta, B. Yang, K. Zhu, D. Laird, R. L. Ball, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, "Mura," 2018. [Online]. Available: https://arxiv.org/abs/1712.06957

[5] K. Team, "keras," 2015. [Online]. Available: https://github.com/keras-team/keras

[6] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[7] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[8] T. pandas development team, “pandas-dev/pandas: Pandas,” Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[10] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[11] M. J. A. R. S. A. U. S. T. K. Whangbo, “Konet: Toward a weighted ensemble learning model for knee osteoporosis classification,” *https://ieeexplore.ieee.org/abstract/document/10379081*, 2024.

[12] J. B. Tsai Shih Yang, “Recognition and classification of knee osteoporosis and osteoarthritis severity using deep learning techniques,” *https://norma.ncirl.ie/6343/*, 2022.

[13] W. Wang, W. Huang, Q. Lu, J. Chen, M. Zhang, and J. Q. . Y. Zhang, “Attention mechanism-based deep learning method for hairline fracture detection in hand x-rays,” *https://link.springer.com/article/10.1007/s00521-022-07412-0*, 2022.

[14] F. Uysal, F. Hardalac, O. Peker, T. Tolunay, and N. Tokgoz, “Classification of shoulder x-ray images with deep learning ensemble models,” *https://www.mdpi.com/2076-3417/11/6/2723/htm*, 2021.

[15] K. Yuxiang, Y. Jie, Z. Ren, W. Cao, Y. Zhang, and Q. Dong, “Dense dilated attentive network for automatic classification of femur trochanteric fracture,” *https://www.hindawi.com/journals/sp/2021/1929800/*, 2021.

[16] S. Kumar, P. Goswami, and S. Batra, “Fuzzy rank-based ensemble model for accurate diagnosis of osteoporosis in knee radiographs,” *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 4, 2023.

[17] Y. Chen, “Classification of atypical femur fracture with deep neural networks,” 2019.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[19] M. Tan and Q. Le, “Efficientnetv2: Smaller models and faster training,” in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[21] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[23] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, “Bam: Bottleneck attention module,” *arXiv preprint arXiv:1807.06514*, 2018.