|  |  |  |
| --- | --- | --- |
| Z:\Departement_Mathematiques\Master DSC\2019-2020\FS FI logo.jpg |  | Academic Year 2022-2023 |

**Master II**

**Curriculum: Data Science**

**Final Thesis**

Titre

**Age Detection Model from whole body images with Pose Estimation Key Points**

By

**Rachad Lakis**

Thesis Director: **Dr. Joseph CONSTANTIN**

Academic Supervisors: **Dr. Joseph CONSTANTIN**  
Location: **AIDirections (Dubai)**

Jury Members:

Date: **DD-MM-YYYY**

Code du mémoire (Réservé à l’administration)

## Abstract

There are numerous situations where it is relevant to estimate the age of a person. Either it is for customer profiling, security measures for digital signatures, or intelligent video surveillance, a model capable of solving this task would be of immense value to society or industry. Deep learning techniques were explored in this study to create a model capable of accurately estimating the age category of a given whole body image of a person. Specifically, we employed transfer learning and evaluated the performance of popular pre-trained deep learning models including VGG19, ResNet-50, MobileNetV2 and some custom models. These models were trained and tested on a labelled dataset of images.

In our study, we achieved remarkable accuracy and f1-score of approximately 98% for the targeted model. Furthermore, we explored the impact of incorporating pose estimation key points into the model pipeline. Through extensive experimentation and evaluation, we got many models having very good accuracy, precision, and recall for predicting age categories in images, and proved that adding the pose estimation key points to an image can significantly improve the age category identification. Considering its smaller weights, we highly recommend the utilization of the model based on MobileNetV2 for future production purposes.

In summary, this thesis highlights the significance of age estimation in various applications and explores the effectiveness of deep learning techniques. By evaluating multiple pre-trained models, custom models and investigating the impact of pose estimation key points, we have identified the MobileNetV2 model as the most suitable choice for accurate age estimation and demonstrated that the new technique can be an added value for the age classification problem.

## Acknowledgements

I would like to express my gratitude to the academic staff at the University Saint Joseph's Master of Data Science department, especially Dr. Ziad Francis, Dr. Maroun Chamoun, and particularly my academic coordinator Dr. Joseph CONSTANTIN. Their tutoring, encouragement, and patience have played a significant role in shaping our academic journey and success.

I extend my thanks to AIDirections, represented by Mr Yohannes Eksteen and Dr. Angelika Eksteen, the company where I pursued my internship, for their support, trust and respect. Special appreciation goes to my supervisor, Dr. Dounia Awad, for her guidance and advice and for.

I am deeply grateful to my family and friends, particularly Hussein Hazimeh, for their unwavering support and love. Their support and guidance have been instrumental in my success, and I am truly grateful for their presence in my life and academic journey.

Table of Contents

[Abstract 2](#_Toc138695562)

[Acknowledgements 3](#_Toc138695563)

[Acronyms 7](#_Toc138695564)

[Chapter 1: Introduction 9](#_Toc138695565)

[1.1 Background 9](#_Toc138695566)

[1.2 Context 10](#_Toc138695567)

[1.3 Purposes 10](#_Toc138695568)

[1.4 Significance, Scope, and Definitions 10](#_Toc138695569)

[1.5 Thesis Outline 11](#_Toc138695570)

[Chapter 2: Literature Review 13](#_Toc138695571)

[Part 1: Literature Review on Deep Learning: Convolution Neural Networks CNNs 13](#_Toc138695572)

[2.1 Introduction 13](#_Toc138695573)

[2.2 Deep Learning 16](#_Toc138695574)

[2.3 Convolution Neural Network: 19](#_Toc138695575)

[2.4 Convolutional Operator 20](#_Toc138695576)

[2.5 Convolutional Layer 21](#_Toc138695577)

[2.6 Pooling Layers 25](#_Toc138695578)

[2.7 Activation 26](#_Toc138695579)

[2.8 Loss 28](#_Toc138695580)

[2.9 Back Propagation 29](#_Toc138695581)

[2.10 Transfer Learning: 30](#_Toc138695582)

[Part 2: Age Estimation Using machine learning. 32](#_Toc138695583)

[2.1 Introduction 32](#_Toc138695584)

[2.2 Facial Age Estimation Using Convolutional Neural Networks 32](#_Toc138695585)

[2.3 DEX: Deep Expectation of Apparent Age from a Single Image 33](#_Toc138695586)

[2.4 SSR-Net: Soft Stagewise Regression Network for Age Estimation 34](#_Toc138695587)

[Additionally, the paper "SSR-Net: Soft Stagewise Regression Network for Age Estimation" introduces a novel approach to age estimation by combining multi-class classification with regression. 34](#_Toc138695588)

[2.4 Age Estimation Using Active Appearance Model and Support Vector Machine 36](#_Toc138695589)

[2.6 Conclusion 36](#_Toc138695590)

[Chapter 3: Research Design 37](#_Toc138695591)

[3.1 Methodology 37](#_Toc138695592)

[3.2 Movenet Thunder: TensorFlow Single Pose Estimation Key Points 38](#_Toc138695593)

[3.3 The datasets 40](#_Toc138695594)

[3.3.1 First Dataset 40](#_Toc138695595)

[3.3.1 Expanding the dataset: 41](#_Toc138695596)

[3.3.2 Bing Image Search: 42](#_Toc138695597)

[3.3.3 Second Dataset: 42](#_Toc138695598)

[3.4 Adjusting Class Weights: 43](#_Toc138695599)

[3.5 Convolutional Neural Networks Used in this work: 44](#_Toc138695600)

[3.5.1 VGG19: 44](#_Toc138695601)

[3.5.2 ResNet-50: 46](#_Toc138695602)

[3.5.3 MobileNetV2: 47](#_Toc138695603)

[3.2 Transfer Learning 49](#_Toc138695604)

[3.6 Preprocessing Function: 49](#_Toc138695605)

[3.6.1 Function Summary: 49](#_Toc138695606)

[3.6.2 The steps algorithm 50](#_Toc138695607)

[3.7 Approaches to test Pose Estimation Key Points 50](#_Toc138695608)

[3.7.1 First Approach: 50](#_Toc138695609)

[3.7.2 Second Approach: 51](#_Toc138695610)

[3.7.3 Third Approach: 52](#_Toc138695611)

[3.7.4 SimpleNet: 52](#_Toc138695612)

[Chapter 4: Discussion 53](#_Toc138695613)

[4.1 Participants 53](#_Toc138695614)

[4.2 Instruments 53](#_Toc138695615)

[4.3 Ethics and Limitations 54](#_Toc138695616)

[Chapter 5: Implementation 56](#_Toc138695617)

[5.1 Importing Libraries: 56](#_Toc138695618)

[5.2 Preprocessing Function: 58](#_Toc138695619)

[5.3 Build and train model: 60](#_Toc138695620)

[Chapter 6: Results 63](#_Toc138695621)

[**6.1 Metrics Used and Test Dataset** 63](#_Toc138695622)

[6.2 Normal Fine-Tuning Results (Without PEK) 65](#_Toc138695623)

[6.3 Pose Estimation Keypoints method Results 70](#_Toc138695624)

[**6.3.1 First Approach** 70](#_Toc138695625)

[**6.3.2 Second Approach** 70](#_Toc138695626)

[6.3.4 SimpleNet 71](#_Toc138695627)

[Chapter 7: Analysis 73](#_Toc138695628)

[7.2 Part1: Normal Fine-Tuning Results (Without PEK) 73](#_Toc138695629)

[Part 2: Pose Estimation Key points Results. 75](#_Toc138695630)

[6.22 First Approach 75](#_Toc138695631)

[7.2.1 Second Approach: 77](#_Toc138695632)

[7.2.2 SimpleNet 83](#_Toc138695633)

[Chapter 8: Conclusion 85](#_Toc138695634)

[References 86](#_Toc138695635)

**Table of figures**

[Figure 1: Image before and after Convolution 23](#_Toc137546005)

[Figure 2: Convolutional Neural Network Example 24](#_Toc137546006)

[Figure 3: Feature Extraction Examples in CNN 25](#_Toc137546007)

[Figure 4: Standard 2D convolution shown in (a) with filters shown in (b) 26](#_Toc137546008)

[Figure 5: Functional interpretation of 2D convolution 27](#_Toc137546009)

[Figure 6: Input volume (a) and filter (b) are convolved on a per-channel basis, resulting in (c) 28](#_Toc137546010)

[Figure 7 : Max Pooling Operation 29](#_Toc137546011)

[Figure 8: Sigmoid, Tanh and ReLU functions 31](#_Toc137546012)

[Figure 9: Transfer Learning 35](#_Toc137546013)

[Figure 10: SSR-Net architecture 38](#_Toc137546014)

[Figure 11: Example of images with/without Pose Estimation Key Points 45](#_Toc137546015)

[*Figure 12: VGG-19 architecture* 48](#_Toc137546016)

[Figure 13: ResNet-50 architecture 49](#_Toc137546017)

[Figure 14: MobileNetV2 overall architecture 50](#_Toc137546018)

[Figure 15: Visualization of the intermediate feature maps in the inverted residual layer 51](#_Toc137546019)

[Figure 16: MobileNetV2 convolutional neural network architecture 52](#_Toc137546020)

[Figure 17: Importing Libraries 60](#_Toc137546021)

[Figure 18: Preprocessing function 63](#_Toc137546022)

[Figure 19: Image Generator with Preprocessing\_function 64](#_Toc137546023)

[Figure 21: Build and Train the model. 64](#_Toc137546024)

[Figure 22:VGG-19 Fine Tuning Results 66](#_Toc137546025)

[Figure 23: First Approach Metrics 80](#_Toc137546026)

[Figure 24: Second Approach Metrics 81](#_Toc137546027)

[Figure 25: Third Approach Metrics 83](#_Toc137546028)

## Acronyms

|  |  |
| --- | --- |
| CNN  DL  ML  TP  FP  TN  FN  Acc  Prec  Rec  PEK | Convolutional Neural Networks  Deep Learning  Machine Learning  True Positives  False Positives  True Negatives  False Negatives  Accuracy  Precision  Recall  Pose Estimation Key Points |

In this thesis, a matrix is represented in bold uppercase (for example, **A** = [1 2 3; 4 5 6; 7 8 9] represents a matrix), a vector is represented in bold lowercase (for example **x** = [1 2 3] represents a vector), and a scalar is represented in normal font (for example c = 10 denotes a scalar c).

# Introduction

This chapter provides an overview of the research on image age classification using deep learning and transfer learning, as well as pose estimation key points. It outlines the background and context of the research, its objectives, and the significance of the study. Finally, this chapter gives an outline of the remaining chapters of the thesis.

### 1.1 Background

Age classification in images has been a well-established and extensively researched domain in computer vision [Smith, A., Jones, B., & Johnson, C. , 2018]. The accurate estimation of age in images holds significant relevance across various fields, including biometrics, security systems, and social media analytics. Earlier approaches heavily relied on manual feature engineering, which often proved to be time-consuming and yielded limited accuracy [Gao, X., Wang, L., & Li, H., 2017].

The field of age classification in images has witnessed significant advancements with the emergence of deep learning and transfer learning techniques, leading to notable enhancements in accuracy and efficiency [He, K., Zhang, X., Ren, S., & Sun, J. ]. However, most of these studies have predominantly focused on age detection solely from facial images. Nonetheless, the availability of facial images may not always be guaranteed in practical scenarios. Therefore, the primary objective of this research is to develop a model capable of accurately classifying the age category of an image based on the analysis of the entire body, rather than relying solely on facial information.

### Context

The study will focus on developing an image age classifier using deep learning and transfer learning, as well as pose estimation key points. The classifier will be trained on a dataset of images, provided by the AIDirections company, and with people wearing the traditional clothes of this country, and the objective is to predict the age category of the person in the image accurately specifying if this person is a child or an adult.

### 1.3 Purposes

The primary purpose of this study is to develop an accurate image age classifier using deep learning and transfer learning. The specific aims and objectives of the study are to explore different deep learning architectures for age classification, transfer learning techniques for fine-tuning pre-trained models, and pose estimation key points for “improving” the performance of the classifier. The practical outcome of the study is an accurate and efficient image age classifier that can be applied to various applications.

### 1.4 Significance, Scope, and Definitions

The research aims to fill a gap in the literature by building a model that can distinguish between child and adult, thus having the age category of a whole-body image, knowing that most age estimation models are built on face images only. The model should be as fast as possible and as small as possible, because in the case of this project tens and maybe hundreds of cameras will be feeding the model with live stream videos and the model should give the age category of the people detected. The scope of the study is limited to the age classification of people in images, and the delimitations include variations in facial expressions, lighting, and pose.

Definitions of terms used include:

**Deep learning:** A subfield of machine learning that uses neural networks to model complex patterns in data.

**Transfer learning**: A technique in deep learning that involves using a pre-trained model as a starting point for a new task.

**Pose estimation key points:** Specific points on the body used to estimate the position and orientation of an object in an image.

**Convolutional Neural Networks:** Deep learning models specifically designed for analyzing visual data by using learnable filters to extract features and spatial hierarchies from the input images.

### 1.5 Thesis Outline

**Chapter 2: Literature Review**

This chapter will be divided into two parts. The first part focuses on the literature review of deep learning, specifically Convolutional Neural Networks (CNNs). The second part is dedicated to the literature review on age detection.

**Part 1:** Literature Review on Deep Learning - Convolutional Neural Networks (CNNs):

This section provides a detailed explanation of the architecture and design of Convolutional Neural Networks. As CNNs play a crucial role in constructing the optimal model for age detection, it is essential to delve into their structure. CNNs, also referred to as ConvNets, serve as the fundamental building blocks for this thesis and are widely employed in various classification problems within the field of Machine Learning.

**Part 2:** Literature Review on Age Estimation using machine learning:

The second part focuses on reviewing the literature related to age classification in images, and the key points of pose estimation. By examining existing research in these areas, this section aims to provide a comprehensive understanding of age detection methodologies and the approaches used previously to tackle this specific problem.

**Chapter 3: Research Design**

This chapter will detail the research methodology, including the dataset used, the deep learning architectures explored, the transfer learning techniques used, and the pose estimation key points.

**Chapter 4: Implementation**

This chapter will delve into the technical implementation of the image classification and pose estimation algorithm, providing a detailed account of the steps taken to execute the proposed methodology.

**Chapter 5: Results**

Present the results of the study in a clear and organized manner.

**Chapter 6: Analysis**

Analyse the results and compare them to the literature review to draw conclusions and identify any limitations.

**Chapter 7: Conclusion**

Summarize the findings of the study and their implications for future research, as well as any practical applications of the image classification and pose estimation algorithm.

# Literature Review

## Part 1: Literature Review on Deep Learning: Convolution Neural Networks CNNs

### 2.1 Introduction

Machine Learning tasks can be divided into many types: Supervised Learning, Unsupervised Learning, semi supervised Learning and Reinforcement Learning.

**Supervised Learning:**

Supervised learning involves training a model to learn a function that maps input data to corresponding output labels based on labelled training examples. The model aims to generalize its learned knowledge to make predictions on unseen data.

Example of Application:

Supervised learning can be applied to medical diagnosis, where a model is trained on a labelled dataset of patient symptoms and corresponding diagnoses where each image has a label citing if this image has or not the disease. The trained model can then predict the diagnosis of new patients based on their symptoms, assisting doctors in making accurate and timely decisions.

**Unsupervised Learning:**

Unsupervised learning deals with learning patterns or structures within unlabelled data without any explicit output labels. The goal is to uncover hidden relationships or groupings in the data.

Example of Application:

An example of unsupervised learning application is customer segmentation in marketing. By analyzing customer data without explicit labels, such as purchase history, website behaviour, and demographics, unsupervised learning algorithms can identify distinct groups of customers with similar characteristics. This helps businesses tailor their marketing strategies and offerings to specific customer segments.

**Semi-supervised Learning:**

Semi-supervised learning combines elements of supervised and unsupervised learning by utilizing a limited amount of labelled data along with a larger amount of unlabeled data. The model leverages the unlabeled data to improve its performance and generalize better.

Example of application:

Consider the task of sentiment analysis in social media. The objective is to develop a model capable of classifying tweets into positive, negative, or neutral sentiments. However, manually annotating a large dataset of tweets with sentiment labels proves to be a resource-intensive and time-consuming process. The approach involves utilizing a small set of labelled tweets with assigned sentiment labels, alongside a significantly larger corpus of unlabeled tweets without explicit sentiment annotations.

In the semi-supervised learning paradigm, an initial model is trained using the labelled data. This model learns from the labelled examples and subsequently makes predictions on the unlabeled tweets. The resulting predictions are then treated as pseudo-labels for the sentiment of the unlabeled tweets.

The process continues by combining the labelled data with the augmented dataset of pseudo-labelled tweets, forming an expanded training set for the subsequent iteration. The model iteratively refines its predictions using this augmented dataset, improving the accuracy of the pseudo-labels for the remaining unlabeled tweets.

This iterative training and pseudo-labelling procedure continue until convergence or achieving the desired performance level.

**Reinforcement Learning:**

Reinforcement learning involves an agent learning to make decisions in an environment through trial and error, aiming to maximize a cumulative reward signal. The agent learns from the consequences of its actions, guided by a reward or punishment mechanism.

Example of application:

Reinforcement learning is commonly applied in robotics, where an autonomous robot learns to navigate an environment and perform specific tasks. The robot interacts with its surroundings, receiving rewards for successful actions and punishments for failures. Over time, the robot learns the optimal actions to achieve its goals, such as picking up objects or navigating obstacles.

The first one, the most common form and the one that will be used throughout this thesis, is a task that consists of learning a function that maps a set of independent variables, i.e., the input, to a desired output, or a dependent variable. For this type of learning it is vital that we possess enough examples of input/output pairs on which the algorithm will train on. This paradigm’s key factor for a successful supervised learning algorithm is the algorithm’s ability to generalize to unseen data.

For many years, creating a machine-learning system required careful engineering and an extensive expert knowledge to design an efficient feature extractor to transform raw data (such as characters of a text or pixels of an image) into a representation or abstraction, of which the underlying system could learn patterns from.

Deep learning introduced representation learning with multiple levels of abstraction, obtained by composing simple but non-linear modules capable of transforming the data representation at one level into a representation level with a higher level of abstraction. When an image, for example, is fed into a deep learning algorithm, the learned features on the first representation layers will most likely represent the presence or absence of low-level features such as edges, at certain location and direction. Subsequent layers will detect combinations of this edges that will in eventually create representations of textures and objects. By layering together enough transformations, it is possible to learn very complex functions, and by learning feature representation, features are no longer designed by humans and are learned directly from data[Yann LeCun, Yoshua Bengio, and Geoffrey Hinton., May 2015] **.**

Convolutional Neural Networks were first introduced by [Kunihiko Fukushima and Sei Miyake, 1982] and later refined by[Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio, 1999] who first applied Back Propagation as the training algorithm. These models were inspired by neural connections, hence the name, and are one of the most successful applications of the knowledge gathered by studying the brain. These models take advantage of structured data with spatially relevant information, whether it be audio (1D), images (2D) or volumetric images(3D) for example.

CNNs were the basis of the Deep Learning boom when [Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 2012] defeated its competitors on the ImageNet challenge by successfully applying a Deep CNN for the first time.

### Deep Learning

Deep learning, a subfield of machine learning, has gained popularity due to its ability to solve complex problems by utilizing artificial neural networks. These networks consist of multiple layers of interconnected nodes, each performing mathematical operations on input data, leading to increasingly nuanced and abstract features being extracted from the data, thereby enabling accurate and powerful predictions.[Yann LeCun, Yoshua Bengio, and Geoffrey Hinton., May 2015]

Deep learning's popularity has been amplified by the availability of large datasets and powerful hardware, as well as advancements in the mathematical foundations of neural networks. The development of backpropagation has been a key breakthrough, which allows efficient training of neural networks by adjusting the weights between nodes based on the error between predicted and actual output**.** [Rumelhart, D. E., Hinton, G. E., & Williams, R. J., 1986]

Deep learning models have been successfully applied in diverse areas like image and speech recognition, natural language processing, robotics, and autonomous vehicles**.** [Krizhevsky, A., Sutskever, I., & Hinton, G. E.] These applications usually involve large and complex datasets, where deep learning models have shown impressive performance in tasks such as object detection, speech recognition, and language translation.

One of the major advantages of deep learning is its ability to learn complex and nonlinear relationships between inputs and outputs, making it particularly useful for problems that traditional machine learning methods struggle with, such as image and speech recognition. Deep learning models can also be used to generate synthetic data, which can be used to augment training datasets and improve model performance. [Goodfellow, I., Bengio, Y., & Courville, A., 2016]

However, deep learning has some limitations such as the need for large amounts of training data which can be time-consuming and expensive to acquire. Additionally, the "black box" nature of deep learning models can make it difficult to understand how predictions are made, and to identify sources of bias or error. To address these challenges, researchers have developed techniques such as feature visualization, attribution methods, and model distillation to interpret and explain deep learning models, thereby identifying the most important input parts for making predictions and sources of bias or error in the model.

Deep learning models can be trained using various optimization algorithms such as stochastic gradient descent, Adam, and RMSprop, which use different strategies for adjusting the weights between nodes and can significantly impact the performance and convergence of the model.

Stochastic gradient descent (SGD) is a widely used optimization algorithm [Bottou, 2010] . It updates the weights by computing the gradients of the loss function with respect to the weights based on a randomly selected subset of training samples. This approach allows for faster updates and is suitable for large datasets.

Adam [Kingma, D. P., & Ba, J. (2014)] and RMSprop [Tieleman, T., & Hinton G., n.d.] are other popular optimization algorithms. Adam combines the advantages of both adaptive learning rates and momentum-based updates. It adjusts the learning rate for each weight based on the historical gradient information, leading to faster convergence. RMSprop, on the other hand, adapts the learning rate for each weight based on the average of past squared gradients, allowing for better handling of sparse gradients and accelerating convergence.

To prevent overfitting and improve generalization performance, regularization techniques are commonly used in deep learning models. Dropout [Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.] randomly sets a fraction of the weights to zero during training, forcing the network to learn redundant representations and reducing overfitting. Weight decay, also known as L2 regularization, adds a penalty term to the loss function based on the magnitude of the weights, encouraging smaller weights, and preventing overemphasis on individual features. Batch normalization [Ioffe, S., & Szegedy, C.] normalizes the input to each layer by subtracting the batch mean and dividing by the batch standard deviation, stabilizing the training process and accelerating convergence.

Convolutional neural networks (CNNs) have been particularly successful in image and video recognition tasks, extracting local features by using filters that convolve over the input data, which are then combined across multiple layers to form increasingly abstract and complex representations. [Krizhevsky, A., Sutskever, I., & Hinton, G. E]

Recurrent neural networks (RNNs) are another type of deep learning model well-suited for sequential data, such as text or time series data. These models use loops to pass information from one timestep to the next, allowing them to capture temporal dependencies and long-term patterns in the data. [Hochreiter, S., & Schmidhuber, J.]

In this thesis, Deep Learning is used to achieve the specified task, especially with the use of CNNs and Transfer Learning with Fine Tuning.

### Convolution Neural Network:

Convolutional Neural Networks (CNNs) are an effective type of deep learning model that has achieved remarkable performance in image and video recognition tasks [LeCun, Y., Bengio, Y., & Hinton, G., 2015]. CNNs are specifically designed to process input data with a grid-like structure, such as an image or a video frame, by treating the grid of pixels as a two-dimensional array, with each pixel having a numerical value representing its brightness or colour.

One of the main advantages of CNNs is their ability to learn spatial hierarchies of features [Yann LeCun, Yoshua Bengio, and Geoffrey Hinton., May 2015]**.** The lower layers of the model learn simple and local patterns, while the higher layers capture more complex and global patterns that depend on the combination of lower-level features. This hierarchical approach to feature learning has been shown to be effective in a variety of image recognition tasks.

CNNs often include fully connected layers, which perform a nonlinear transformation on the high-level features and output a prediction or a probability distribution over the classes. The VGG network is a commonly used architecture for CNNs, consisting of multiple convolutional layers followed by fully connected layers. This architecture has achieved state-of-the-art performance on the ImageNet dataset, which contains millions of labelled images[Simonyan, K., & Zisserman, A., 2015].

Another popular architecture for CNNs is the ResNet-50 network, which uses residual connections to address the issue of vanishing gradients [He, K., Zhang, X., Ren, S., & Sun, J. ] ResNet-50 has achieved state-of-the-art performance on a range of image recognition tasks, including object detection and segmentation.

Overall, CNNs have demonstrated their effectiveness in a variety of applications, particularly in image and video recognition tasks, due to their ability to learn hierarchical representations of input data.

### 2.4 Convolutional Operator

A convolution operation (equation 1), denoted with an asterisk ∗, at moment *t*, is defined as a real valued operation between two functions, *f* and *g*, and it expresses how the shape of one function is modified by the other.

(2.1)

The discrete valued convolution operator is given by (equation 2). This formulation of the same operation is necessary because the computational implementation of this operation requires it to support discrete values as input.

(2.2)

Until now, our convolution formulations have supported kernels which support an infinite real set of values, i.e. *g:* R → R. When g has a finite support set, i.e. *τ* → {0*,* 1*, ..., M* − 1*, M*}, the convolution operation can be described with (equation 3).

(2.3)

This is also important since in practice, our input and kernels are arrays of data with limited sizes. Moreover, these inputs and kernels can have multiple dimensions. Convolutional operations can be rewritten to support this kind of multidimensional input. A 2D convolution operation is defined by:

(2.4)

In practice, to avoid the unnecessary complexity of flipping the kernel or the input (convolutions are commutative), we use a very similar operation which is called cross-correlation (2.5), which does not flip either the kernel or the input. This has no real impact on the operation in convolutional neural networks since the kernel is a set of features to be learned and it does not matter if the kernel is learned flipped or not if it is processed in the same manner every time.

(2.5)

An image is the perfect example of a 2D input to a convolution operator as it defined by a grid of pixel values. Convolving the image with a kernel generates a feature map composed of kernel activated features at each block of the image.

In figure 1, it is possible to see an example of the result of a convolution operation.



Figure 1: Image before and after Convolution

Source: [Sobel, 2014]

### 2.5 Convolutional Layer

Convolutional layers take advantage of the spatially relevant information of data with a grid-like topology, unlike Fully Connected layers which treats every element, which can be an element of the input or an element of a previous layer which we call a neuron, in the same manner. Each connection in a network contains a weighted parameter and each neuron a bias parameter. These parameters are very commonly referred as the weights of the network. Both types of parameters can be updated and are optimized during learning.

A Neural Network which contains Convolutional layers is typically called a Convolutional Neural Network (CNN), although they also usually possess other kind of layers such as Pooling and Fully Connected layers. A CNN is created by stacking these layers, which transform the input in a fully differentiable manner.

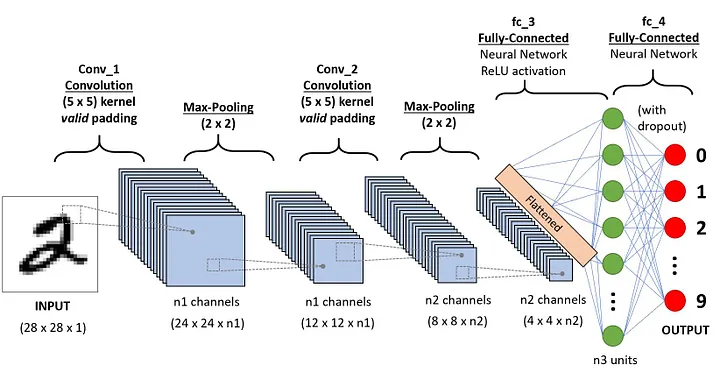
A Fully Connected Neural Network is composed by a series of Fully Connected layers. This network receives a 1D input which is iteratively transformed by the Fully Connected layers or hidden layers. Each neuron of a Fully Connected layer is connected to every hidden neuron of the previous layer which creates a global receptive field. However, when dealing with multi-dimensional data inputs it becomes unfeasible to connect every hidden neuron to all neurons in the previous layer.

Figure 2: Convolutional Neural Network Example

Source of image: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

On the other hand, Convolution Layers connect to a reduced local region, in space, of the input, resulting in a local receptive field with the size of the kernel, illustrated in Figure 3. The Convolution Layer operation can be seen as a sliding window operation, where the kernel slides through the image across its width and height and computes the dot product between the kernel and the image at each location. From this operation results a 2D feature map which shows the activated kernel features at every location in the image. This sliding window operation can be performed multiple times in a single layer if multiple kernels are defined. The stride with which we slide, and the number of kernels is the hyperparameters that contribute to the shape of the output of a Convolutional Layer.

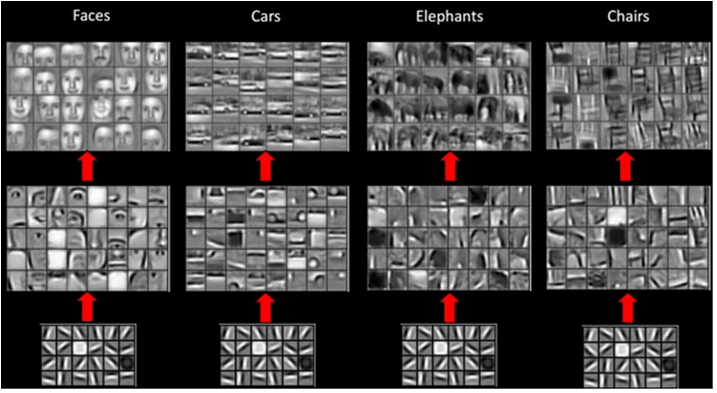


Figure 3: Feature Extraction Examples in CNN

Source: [Charels Siegel, Jeff Daily]

Since the same kernel is used to slide through the input, the weights of the connections are applied multiple times, therefore the elements of the output share kernel weights. This also form of parameter sharing introduces translation equivariance meaning that when the input is translated, the output is translated the same way[Ian Goodfellow, Yoshua Bengio and Aran Courville]**.**

By choosing kernels smaller than the input to we can reduce the number of connections. This occurs because while a Fully Connected contains number of learnable parameters, where N and M are the input and output sizes, in Convolutional Layers there are , where is the kernel size and *W*, *H* and *D* are the width, height and depth of the kernel.

The output of Convolutional Layer operation is described by Equation 6.

(6.6)

**Overview of Standard 2D Convolution**

Suppose a convolution operation transforms an input volume of dimensions to an output volume of dimensions , as shown in Fig. 4(a). Specifically, we require N filters, each of dimension, as shown in Fig. 4 (b). [Gonzales, 2019]

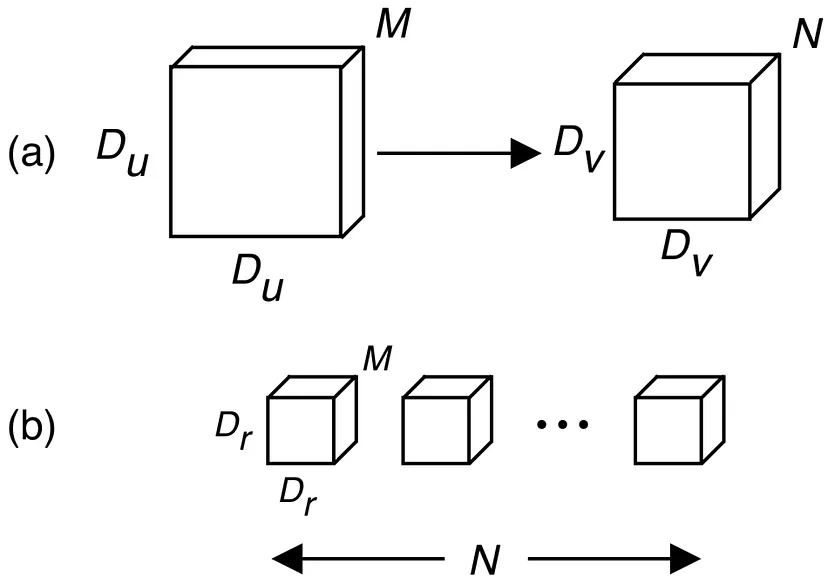


Figure 4: Standard 2D convolution shown in (a) with filters shown in (b)

The perhaps-more-functionally-correct interpretation of 2D convolution is portrayed in Fig. 5. A filter (shown in orange) is stepped along the spatial dimensions of an input volume (shown in blue). An inner product is taken between the overlapping regions of the input volume and filter at every step (shown in light red). In practice, the overlapping portion of both the filter and input volume are vectorized and the dot product is taken between the two resulting vectors. In either case, a single value is computed, as shown by the red element in Fig. 5. Although functionally correct, this interpretation obscures the spatial filtering that occurs within the convolution. [Gonzales, 2019]

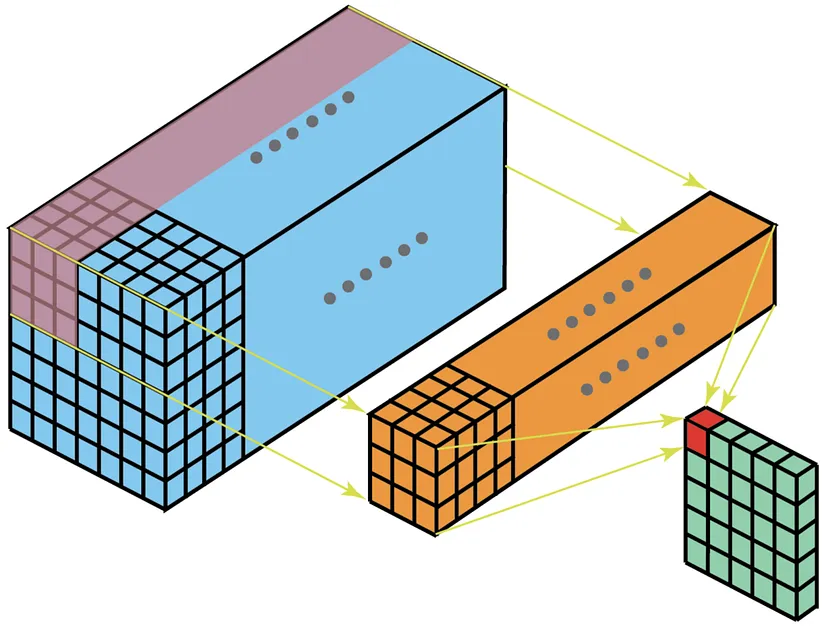


Figure 5: Functional interpretation of 2D convolution

The other interpretation of the standard 2D convolution draws more emphasis to the spatial filtering that takes place. Like the interpretation above, a filter, shown in Fig. 6(b), is stepped along the spatial dimensions of an input volume, shown in Fig. 3(a). But rather than the inner product spanning the depth dimension, the inner product is taken on a per-channel basis. In other words, channel i in the input volume is convolved with channel i in the filter (single channel convolution), where i indexes along the depth dimension. The resulting volume is shown in Fig. 3(c). Finally, all resulting values are summed, leading to a single value, which is not shown. Again, note that this interpretation emphasizes the spatial filtering and that for N filters, each and every channel of the input volume is filtered N times, which begins to seem rather excessive. [Gonzales, 2019]

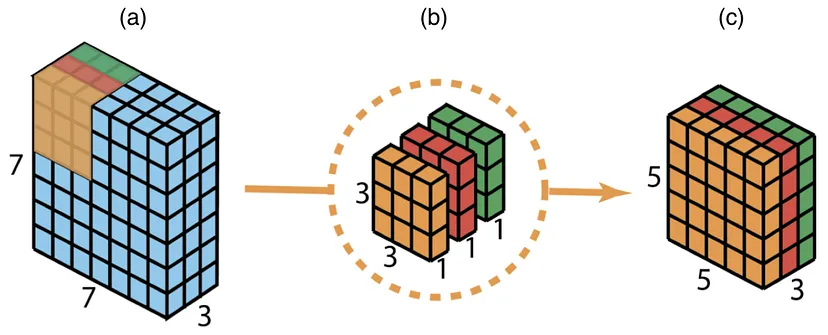


Figure 6: Input volume (a) and filter (b) are convolved on a per-channel basis, resulting in (c)

### 2.6 Pooling Layers

Along with Convolutional Layers, Pooling layers are also an important component of a CNN. Usually, Pooling Layers are inserted after a stack of Convolutional Layers. The Pooling operation creates a smaller feature map by applying a filter to subregions of the input. The operation is then applied, much like the convolution operation, in a sliding window like fashion. An illustration of the operation can be seen in Figure 4.

Pooling allows the network to become approximately invariant to small translations and at the same time, by reducing the resolution of the feature maps, it helps decrease the complexity of the network. In most applications of a CNN, translation invariance is a useful property since it is usually more important to know if a feature is present than to know precisely where it is. Using Pooling can be seen as adding an infinitely strong prior to the function the layer should learn, dictating it must be invariant to small **translations** [Ian Goodfellow, Yoshua Bengio and Aran Courville]**.**

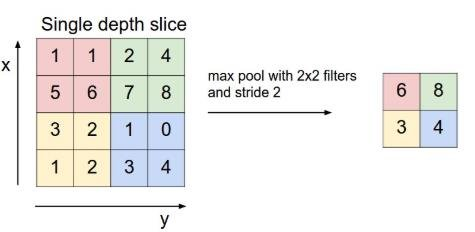
****

Figure 7 : Max Pooling Operation

Source of Fig 5: [Lakkavaram]

There are multiple ways to filter a region of features. Here we describe the two most common variations of a Pooling Operation:

* **MaxPooling**: MaxPooling finds the maximum value for each subregion and creates an out- put where each element is the maximum value of every region.
* **AvgPooling**: AvgPooling averages the feature values of each subregion and creates an out- put where each element is the average value of every region.

### **2.7 Activation**

The Universal Approximation Theorem [Csáji., 2001.]states that a neural network single hidden layer can approximate any continuous function on compact subsets of , given enough hidden neurons and an activation function that is non-constant, bounded, and monotonically-increasing continuous, can approximate any continuous function on compact subsets of .

Up until this point, the network would only apply a linear transformation to the input. Moreover, the network would not satisfy the Universal Approximation Theorem therefore it wouldn’t be able to learn more complex non-linear functions. To do so, we must introduce non- linear activation functions. Activation functions are applied after a linear operation and are an element-wise operation which means it does not affect receptive fields.

Given an activation function *σ* , the Equation 6 is rewritten as:

(2.7)

The chosen functions need to be differentiable to be trained. The most used functions are:

* **Sigmoid:** The Sigmoid function, illustrated in Figure 6, is defined by:

(2.8)

This biologically inspired function squashes the input values to a range of ]0, 1[. However, this function has two properties that pose major disadvantages which are the not zero-centred output and the fact it is more prone to gradient saturation.

* **Tanh:** The Hyperbolic Tangent (tanh), illustrated in Figure 6, function is defined by:

(2.9)

This function squashes the input value to the range] − 1, 1[. Tanh is a rescaled sigmoid function however this version is zero-centred. Nonetheless, it still suffers from the gradient saturation problem.

* **ReLU**  [Vinod Nair and Geoffry Hunton, 2010]**:**

Along with the reduction in computational complexity, ReLU fixes the gradient saturation issue present in the previous functions however it introduces other problems such as dead ReLU units. Subsequent iterations of ReLU such as LeakyReLU or Exponential Linear Units (ELU) address this issue. This function is not differentiable in all its domain with its derivative at x = 0 being undefined. To use back propagation, explained in Section 2.7, every operation in the network must be differentiable. In practice, the derivative of ReLU is defined as:

(2.10)

Chart, line chart

Description automatically generatedChart, line chart

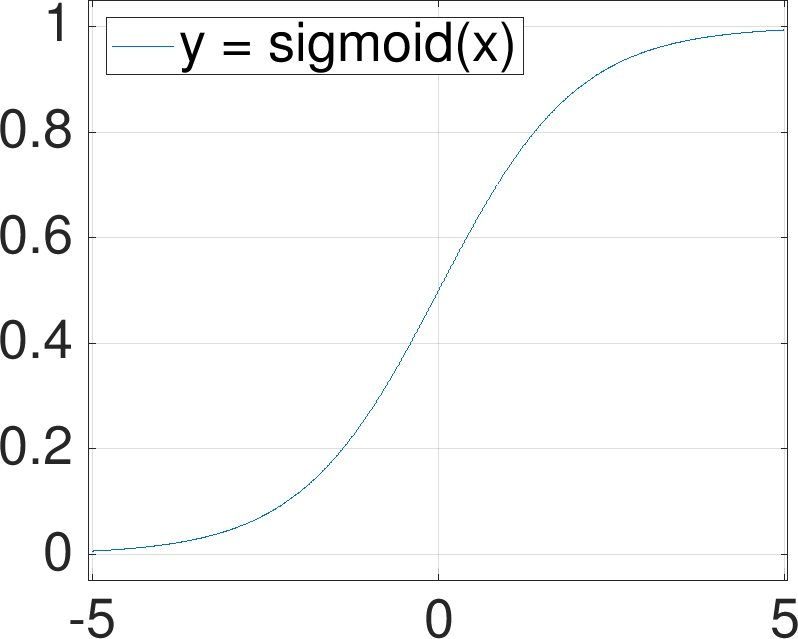
Description automatically generated

Figure 8: Sigmoid, Tanh and ReLU functions

### 2.8 Loss

As mentioned before, CNN’s usually contain Fully Connected Layers which are generally the last layers to be stacked. After high and low-level feature extraction, we can take advantage of the global receptive field of these layers and perform a global transformation of the input.

The last layer is usually one dimensional with as many units as labels in the case of a classification problem or a single unit in case of a regression problem. The values of this last layer either represent the score, in the form of logits, of the input for each class or the estimated regressed value.

To train a CNN, we need to come up with a loss or cost function that can measure the quality of model’s predictions or regressions, given certain parameters. The network will be optimized to minimize this value. The most common loss function for classification problems is the Cross Entropy (Equation 11), applied after a SoftMax (Equation 12) operation which squashes a vector of arbitrary values to a vector of real values, with each entry having a value in the range ]0, 1] and , while for regression problems is commonly used the Mean Squared Error or the Absolute Squared Error (MSE/MAE).

(2.11)

(2.12)

(2.13)

(2.14)

### 2.9 Back Propagation

For the network to output optimal values its parameters must be adjusted. The optimization technique most applied to CNN’s is gradient descent. To update the weights, we need to compute the gradient of the loss with respect to the network’s parameters, which are the weights of the connections and bias. Due to the stacking nature of neural networks, the gradient of each layer can be propagated backwards to an earlier layer for a more efficient computation. This algorithm is named Back Propagation and is powered using consecutive applications of the chain rule:

(2.15)

The gradient descent algorithm is deﬁned by the Equation:

(2.16)

where indicates the model’s parameters on iteration t, in this case an epoch, and λ is a weighting parameter called Learning Rate (LR), which dictates the size of the step taken in the direction of the negative gradient.

Up until this point, each parameter update was computed using the entire training dataset. When dealing with simple loss surfaces and small datasets this approach works, however in practice we deal with extremely complex and non-convex, smooth loss surfaces and huge datasets to fight model overfitting which makes the task of fitting all the data in one iteration unfeasible. Instead, we can divide our training dataset into smaller chunks or batches and iteratively updating the networks parameters using each batch at a time which also takes advantage of adding more variance to the updates since the gradients are noisier. This optimization technique is called mini-batch gradient descent and can be defined by (equation 17). The size of the batch can vary and with batch size of one we call it Stochastic Gradient Descent (SGD), defined by Equation 18, where now stands for the parameters on the batch iteration t.

(2.17)

(2.18)

Convolutional Neural Network explanation and equations are mostly from [Castro, Age Estimation using Deep Learning on 3d Facial Features (Master thesis)]

### 2.10 Transfer Learning:

Transfer learning and fine-tuning have revolutionized the field of deep learning, enabling researchers to achieve state-of-the-art performance on many tasks with smaller datasets and fewer computational resources. Transfer learning involves using a pre-trained deep learning model as a starting point for a new task. The idea is that the features learned by the pre-trained model on a large dataset can be reused for a new, related task, as demonstrated by [Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 2012] in their influential paper "Learning Deep Features for Recognition with Convolutional Neural Networks". This paper showed that pre-training a convolutional neural network (CNN) on a large dataset and fine-tuning it on a smaller dataset can improve performance on image classification tasks.

Fine-tuning is a related technique that involves taking a pre-trained model and training it further on a new task, typically with a smaller dataset. This allows the model to adapt to the specific features and nuances of the new task, improving its generalization performance, as described by Chen et al. (2021) in their recent paper "A Unified Framework for Domain Adaptation and Adversarial Robustness". The authors showed how fine-tuning a pre-trained model can improve adversarial robustness and domain adaptation performance.

Both transfer learning and fine-tuning have become popular techniques in deep learning, with many researchers using them to achieve state-of-the-art performance on a wide range of tasks. Ongoing research in the field is focused on developing new techniques and strategies for improving the performance and reliability of transfer learning and fine-tuning in deep learning [Yosinski, J., Clune, J., Bengio, Y., & Lipson, H.]

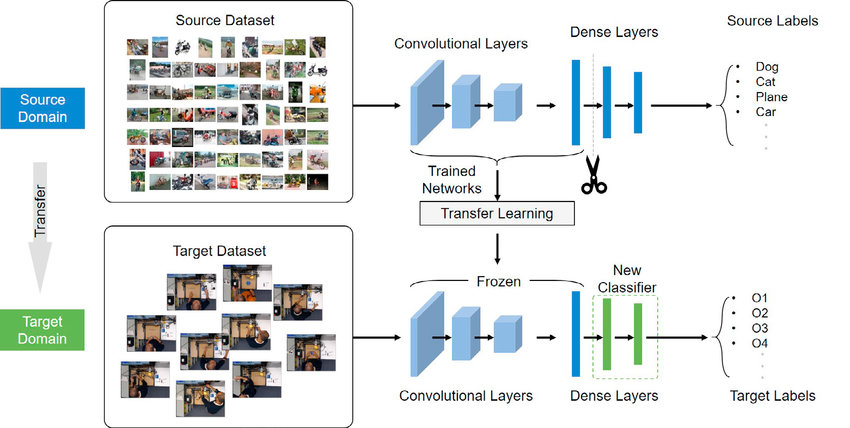


Figure 9: Transfer Learning

Source of image: <https://www.researchgate.net/figure/The-architecture-of-our-transfer-learning-model_fig4_342400905>

## Part 2: Age Estimation Using machine learning.

### 2.1 Introduction

Age detection is a challenging task in computer vision that has gained increasing attention in recent years. The goal of age detection is to predict the age of an individual based on an input image or video. This task has numerous real-world applications, including age-based advertising, age-based security systems, and age-based content filtering. Various deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants, have been proposed for age detection. These models use different approaches to learn features that are relevant to age prediction.

### 2.2 Facial Age Estimation Using Convolutional Neural Networks

One study, titled "**Facial Age Estimation Using Convolutional Neural Networks**" by [Adrian Kjærran, Erling Stray Bugge, Christian Bakke Vennerød, 2021], presents a deep CNN model with five convolutional layers and three fully connected layers. In this paper, a deep convolutional neural network with five convolutional layers and three fully connected layers are presented to estimate individuals’ ages from images. The model is in its entirety trained from scratch, where a combination of three different datasets is used as training data. These datasets are the APPA dataset, UTK dataset, and IMDB dataset. On the test set, the model achieves a categorical accuracy of 0.52.

Overall, the study provides valuable insights into the development and evaluation of a CNN-based model for facial age estimation, highlighting its strengths and areas for improvement. The combination of multiple datasets and the use of a scratch-trained model contribute to the advancement of age estimation techniques in the field of computer vision.

### 2.3 DEX: Deep Expectation of Apparent Age from a Single Image

Another paper, "**DEX: Deep Expectation of Apparent Age from a Single Image**" focuses on estimating apparent age from still face images [Rasmus Rothe; Radu Timofte; Luc Van Gool]. This paper addresses the task of estimating the apparent age in still face images by utilizing deep learning techniques. To accomplish this, convolutional neural networks (CNNs) based on the VGG-16 architecture are employed, which have been pretrained on the ImageNet dataset for image classification. Due to the limited availability of annotated images for apparent age, potential benefits are explored by finetuning the CNNs on crawled Internet face images that contain age information. For this purpose, a dataset of 0.5 million images of celebrities from IMDB and Wikipedia has been crawled, which has been made publicly available and is currently the largest public dataset for age prediction.

The age regression problem is approached by posing it as a deep classification problem, followed by a softmax expected value refinement. It is shown that our proposed method, Deep Expectation (DEX) of apparent age, improves upon direct regression training of CNNs. The DEX method first detects the face in the test image and then extracts the CNN predictions from an ensemble of 20 networks on the cropped face. The CNNs of DEX are finetuned on both the crawled images and the provided images with apparent age annotations. Importantly, DEX does not utilize explicit facial landmarks.

The DEX method achieved the first place in the ChaLearn LAP 2015 challenge on apparent age estimation, outperforming 115 registered teams, including the human reference.

### **2.4 SSR-Net: Soft Stagewise Regression Network for Age Estimation**

Additionally, the paper "SSR-Net: Soft Stagewise Regression Network for Age Estimation" introduces a novel approach to age estimation by combining multi-class classification with regression.

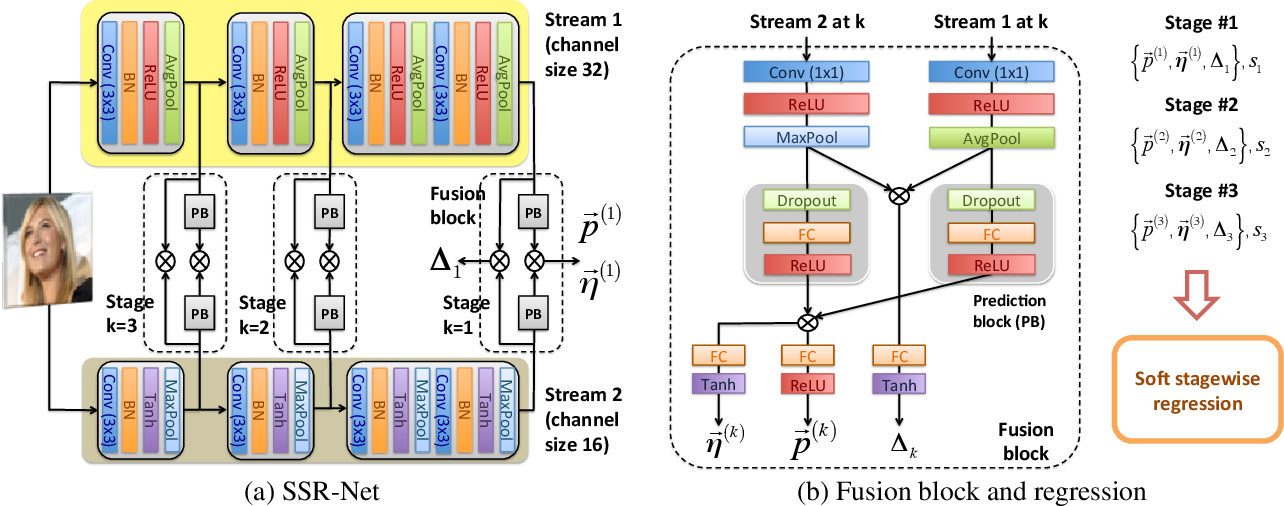
SSR-Net address age estimation by performing multi-class classiﬁcation and then turning classiﬁcation results into regression by calculating the expected values. SSR-Net takes a coarse-to-ﬁne strategy and performs multi-class classiﬁcation with multiple stages. Each stage is only responsible for reﬁning the decision of its previous stage for more accurate age estimation. Thus, each stage performs a task with few classes and requires few neurons, greatly reducing the model size.

Figure 10: SSR-Net architecture

Source of image: <https://www.semanticscholar.org/paper/SSR-Net%3A-A-Compact-Soft-Stagewise-Regression-for-Yang-Huang/d26d07e6d8f1d415c144de51f6d4f9e08efd89c6/figure/0>

For addressing the quantization issue introduced by grouping ages into classes, SSR-Net assigns a dynamic range to each age class by allowing it to be shifted and scaled according to the input face image. Both the multi-stage strategy and the dynamic range are incorporated into the formulation of soft stagewise regression. A novel network architecture is proposed for carrying out soft stage- wise regression. The resultant SSR-Net model is very compact and takes only 0.32 MB. Despite its compact size, SSR-Net’s performance approaches those of the state-of-the-art methods whose model sizes are often more than 1500× larger[Tsun-Yi Yang1, Yi-Hsuan Huang1 , Yen-Yu Lin , Pi-Cheng Hsiu , Yung-Yu Chuang].

Motivated by the complementary 2-stream structure proposed by Yang et al. **[Yang et al., 2017],** SSR-Net adopt a 2-stream model where there are two heterogeneous streams. For both streams, the basic building block is composed of 3 × 3 convolution, batch normalization, non-linear activation and 2 × 2 pooling. However, different types of activation functions (ReLU versus Tanh) and pooling (average versus maximum) are adopted for each stream to make them heterogeneous. This way, they could explore different features and their fusion could improve the performance. Features from different levels are adopted for different stages. For each stage, features from both streams at some levels are fed into a fusion block which is illustrated in Figure1(b).The fusion block is responsible for generating stage-wise outputs, the distribution the offset vector , and the scale factor  , for the kth stage.In the fusion block, features from both streams ﬁrst go through 1 × 1 convolution, activation and pooling for having more compact features. For obtaining , the two obtained feature maps are fused by element-wise multiplication . The product then goes through a fully connected layer and then a Tanh function for obtaining a value in [−1, 1] as . Both and are vectors and more complex. Thus, the features go through and additional prediction block before taking element-wise multiplication, FC layer and activation. Since represents a distribution, ReLU is used as its activation for obtaining positive values. On the other hand, Tanh is used for to allow shift on both positive and negative sides.[Tsun-Yi Yang1, Yi-Hsuan Huang1 , Yen-Yu Lin , Pi-Cheng Hsiu , Yung-Yu Chuang]

### Age Estimation Using Active Appearance Model and Support Vector Machine

Furthermore, "**Age Estimation Using Active Appearance Model and Support Vector Machine**" by [Luu, Khoa- Ricanek, Karl- Bui, T.D. Suen, Ching] proposes a hybrid approach combining the Active Appearance Model (AAM) and Support Vector Machine (SVM) for age estimation. The AAM is used for facial feature extraction, while the SVM is employed for age classification. The study demonstrates promising results on various age datasets, highlighting the effectiveness of this hybrid approach.

### 2.6 Conclusion

Most of the cited models to detect age are based on datasets of facial images, and ultimately give the age estimation of a face image based on face images. Added to that most of the public datasets available are for faces only, and that is not suitable for our case. Inria Person dataset is the only dataset (in our knowledge) that has full body images, but it consists of 800 images only in different positions, and the costume of the people is different than our case, so it is not suitable also.

In our use case, the cameras catch whole body images, and sometimes the faces are not well shown, so these techniques, although are very powerful and precise, will not be very useful. Our technique is based on transfer learning and fine tuning of trained models, according to our dataset of images of the whole body and tries to use the fact that the proportion between the body parts of children and adults are different to make an image classification of an image.

# Research Design

## 3.1 Methodology

The methodology employed in this study is like that of previous works that have used deep learning techniques to classify images based on age categories. In addition, we incorporated pose estimation key points into some of the models to see if it would improve the model's ability to accurately classify age categories.

Google TensorFlow Movenet thunder Pose Estimation is a famous tool for detecting and tracking human body movements. It works by identifying key points on the human body and using them to create a pose estimation model. These key points include the head, neck, shoulders, elbows, wrists, hips, knees, and ankles. The model can be trained using a variety of techniques, including deep learning and computer vision. Once trained, it can accurately detect and track human body movements in real-time video or image data.

We used three methods for detecting the influence of the added points. The first was to use normal transfer learning and fine tuning, the second was to train the model on a more precise dataset for less epochs, and the third was to train the models on precise dataset enough to fully converge. After every train, we compare the metrics. Why these three approaches? Training a model with transfer learning on top of a model like Vgg19, Resnet-50 or MobileNet will lead to a certain convergence. But our goal in this case is not to have a high accuracy, our goal is to test the effectiveness of adding the pose estimation key points on the images as a method to help the age category detection. So, we created these three approaches to spot the light on the new technique tested.

We used an Image Data Generator, and the pose estimation key points were added to the images on the fly, while the images were imported and fed to the model while training.

In brief: “You cannot compare two lights in the daytime”. That was our methodology in brief. Since the models we are using are converging easily, ang getting very high accuracy, comparing the models with or without the pose estimation key points will not help detect the effect of this technique. So, to make it easier to detect the effect of this added technique, we removed all the nearly similar images from the dataset, we used a new partition of 40% of data for training, 30% for validation and 30% for testing. After this we trained the model for 8 epochs only, to have a glance on the results, and then trained for final 30 epochs.

The tables in the chapter Results contain the results of the training.

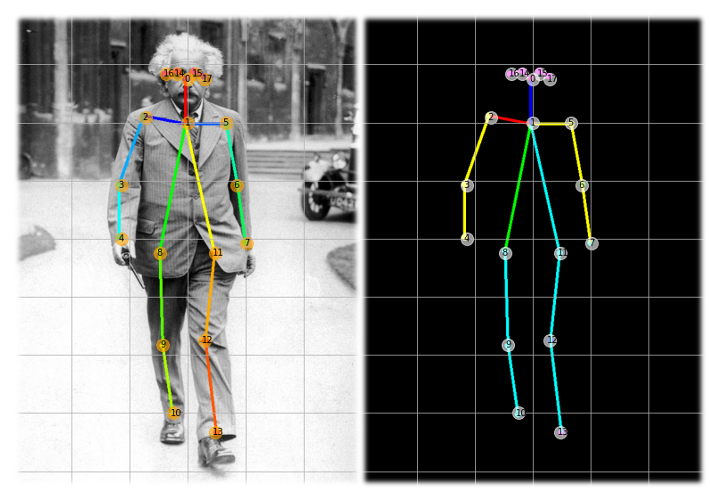
## 3.2 Movenet Thunder: TensorFlow Single Pose Estimation Key Points

TensorFlow Pose Estimation is a deep learning-based technique that estimates human body poses from video and image data. This technique is widely used in various applications, including human-computer interaction, sports analytics, virtual reality, and surveillance. The TensorFlow Pose Estimation model uses the Movenet architecture, which is a family of lightweight, low-latency neural networks developed by Google. Movenet models are optimized for on-device inference and can run in real-time on a wide range of devices, including smartphones, laptops, and embedded systems.

One of the Movenet models that has gained a lot of attention in recent years is Movenet Thunder. Movenet Thunder is a real-time pose estimation model that can run at over 200 frames per second on a single CPU core. It achieves this high performance by using a multi-stage architecture that consists of several lightweight neural networks. The first stage of Movenet Thunder is a person detector that identifies the presence of human bodies in the input image or video. The second stage is a body pose regressor that estimates the 2D or 3D poses of the detected humans. Movenet Thunder can estimate both single-person and multi-person poses and can handle a wide range of body poses and activities, including standing, sitting, walking, running, jumping, and more.

Movenet Thunder has been trained on large-scale datasets, including the COCO dataset and the MPII Human Pose dataset, and has achieved state-of-the-art performance on various benchmark datasets. It is also available as a pre-trained TensorFlow model that can be easily integrated into other applications. The pre-trained model is available in different formats, including TensorFlow Lite, TensorFlow.js, and TensorFlow Hub, making it easy to use on various platforms and devices.

The Movenet Thunder model has many potential applications in various domains, including fitness and wellness, sports analytics, gaming, and entertainment. For example, it can be used to track body movements and provide feedback on exercise form and performance. It can also be used in sports analytics to analyse and compare the movements of athletes, and in gaming and entertainment to enable immersive experiences that respond to the user's body movements. With its high performance, accuracy, and versatility, Movenet Thunder is a promising technique for a wide range of applications that require real-time human pose estimation.

Example: 

Source of image: <https://morioh.com/p/b4b47a464206>

## 3.3 The datasets

In this thesis we worked on two datasets: First we tried all the models training and testing on the dataset given by AIDirections, the company responsible for this internship, and then we created a second dataset to have a more detailed and deeper understanding of the new tested method.

### 3.3.1 First Dataset

The first dataset is given by AIDirections company, and it contains images of adults and children from CCTV cameras in a mall in an Arabic country. The dataset contains a wide variety of images for people in different positions. This is an example of the images with and without the pose estimation key points.

|  |  |
| --- | --- |
| **Original Image** | **Image with PEK** |
|  |  |
|  |  |
|  |  |
|  |  |

Figure 11: Example of images with/without Pose Estimation Key Points

### **3.3.1 Expanding the dataset:**

The original dataset contains images of adults and children, but the dataset was imbalanced. The dataset contains 95% of adults’ images, and only 5% of child images. This imbalance could lead to a hidden biased model. The model will “cleverly” assign all the images labels of adults. This will lead to high accuracy fast, but it will be a “false victory”. This was a very tricky part. After implementing and testing many models, the accuracy of all the models was more than 95%, but that was since more than 90% of the dataset belongs to one class. This error appeared when checking the precision, recall and F-score of the models. A good solution was to expand the dataset and give higher weights to the class of lower occurrences.

### **3.3.2 Bing Image Search:**

The Bing image search feature provided a modest contribution in searching for images similar to those present in the dataset. While its effectiveness was limited, it offered a means to augment the dataset by including images that closely resembled the originals, thereby avoiding the inclusion of significantly dissimilar images.

By utilizing the Bing image search API (Microsoft Corporation, n.d.), relevant images could be retrieved based on similarity criteria, such as visual features. This enabled the expansion of the dataset by incorporating additional examples that maintained consistency with the original images.

Although the Bing image search option proved helpful to some extent, it is important to note that its usefulness might vary depending on the specific requirements and characteristics of the dataset. Consequently, other data augmentation techniques and image sources should also be explored to ensure a diverse and representative dataset for training machine learning models. [Corporation., Microsoft, n.d.]

### 3.3.3 Second Dataset:

In the second phase, we proceeded to generate a fresh dataset to address a limitation observed in the initial set. This limitation stemmed from the fact that a significant portion of the images primarily focused on specific clothing traditions. To broaden the scope and diversity of our dataset, we embarked on an online search for additional images. To facilitate this process and streamline the acquisition of suitable visuals, we leveraged the capabilities of NeoDownloader.

With the intention of enriching our dataset, a crucial step involved the creation of a new collection of images. We recognized that the original dataset had a notable bias towards specific clothing traditions, which prompted us to undertake an online exploration for additional visual resources. This approach enabled us to overcome the limitations of the initial dataset and introduce a wider array of images into our research repository.

We created a new dataset by searching images that show the whole body of a person or a child in different positions. We cleaned all the images and resized them to (224,224,3) and saved as .jpg images. The new dataset was split to 10/10/80 for testing/validation and training.

After we did not find that the new method did not give any improvement to the pre trained models, we tried to build our own models to compare the results of the model before and after the pose estimation key points adding.

In this experiment we created our own dataset that consists of 5349 images for adults, and 6320 images for child, divided into 80% for training, 10% for validation and 10% for testing.

## 3.4 Adjusting Class Weights:

Class imbalance is a prevalent issue in classification problems, where the distribution of classes is skewed. One approach to tackle this problem is by utilizing class weights in Keras, a popular deep learning framework [Colhet, 2015]. By assigning appropriate weights to each class, the classifier can assign higher importance to the under-represented class instances, thereby addressing the imbalance.

To implement this technique, Keras provides a parameter that accepts class weights [Chollet, F. et al., 2018]. By passing these weights during model training, the classifier can allocate more attention to examples from the minority class. This adjustment helps the model to better capture the patterns and nuances associated with the under-represented class, ultimately improving its predictive performance.

By using class weights in Keras, researchers and practitioners can effectively handle class imbalance scenarios, ensuring that the model is not biased towards the majority class but instead pays appropriate attention to all classes in the dataset [Brownlee, 2020]

## 3.5 Convolutional Neural Networks Used in this work:

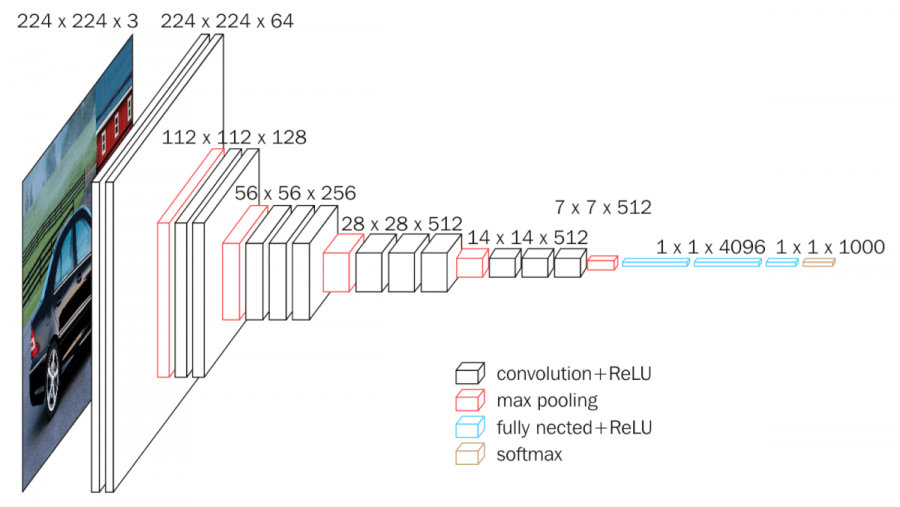
Convolutional neural networks (CNNs) have been widely used in image classification tasks due to their ability to learn hierarchical features from raw pixel data. In the field of computer vision, CNNs have shown state-of-the-art performance in various tasks such as object detection, image segmentation, and facial recognition. According to a review article by [Yann LeCun, Yoshua Bengio, and Geoffrey Hinton] CNNs have been successful in many computer vision tasks due to their ability to automatically learn useful representations from raw data, as opposed to using handcrafted features. Furthermore, recent advancements in CNN architectures, such as the VGG, ResNet, and MobileNet families, have shown even better performance on image classification tasks, even on very large and complex datasets like ImageNet. Therefore, CNNs are a suitable choice for our experiments on the apparel classification task, which involves analyzing images and classifying them into different categories based on their visual features.

### 3.5.1 VGG19:

VGG19 is a deep convolutional neural network architecture proposed by researchers at the University of Oxford in 2014 [Simonyan, K., & Zisserman, A.]. The architecture consists of 19 layers, including 16 convolutional layers and 3 fully connected layers. The VGG19 architecture is known for its simplicity and effectiveness in image classification tasks, achieving state-of-the-art results on the ImageNet dataset [Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. ]

One of the key features of the VGG19 architecture is the use of small, 3x3 convolutional filters, which allow the network to learn complex patterns in the input image at multiple scales while keeping the number of parameters relatively small [Simonyan, K., & Zisserman, A.]. This approach has been shown to improve performance on a variety of image classification tasks [Krizhevsky, A., Sutskever, I., & Hinton, G. E. ]

The VGG19 architecture has been shown to achieve state-of-the-art results on the ImageNet dataset, which contains over 1.2 million images belonging to 1000 different classes. In fact, the VGG19 model achieved a top-5 error rate of just 3.34% on this dataset, which was the best performance at the time of its publication [Simonyan, K., & Zisserman, A.]

One of the reasons why VGG19 is a good choice for image classification tasks is its simplicity and ease of use. The architecture is easy to implement and train and can achieve excellent results with relatively few modifications. Additionally, the use of small convolutional filters and max pooling layers makes the network relatively robust to variations in the input image, such as changes in scale and rotation [Simonyan, K., & Zisserman, A.]

*Figure 12: VGG-19 architecture*

Image source: <https://wikidocs.net/165427>

In conclusion, VGG19 is a powerful convolutional neural network architecture that has achieved state-of-the-art results on the ImageNet dataset and is a good choice for image classification tasks. Its use of small convolutional filters and max pooling layers, combined with its simplicity and ease of use, make it a popular choice among researchers and practitioners in the field of deep learning.

### 3.5.2 ResNet-50:

ResNet-50 is a deep residual neural network architecture proposed by researchers at Microsoft Research [He, K., Zhang, X., Ren, S., & Sun, J.]. The architecture consists of 50 layers, including 49 convolutional layers and 1 fully connected layer. The ResNet-50 architecture is known for its deep structure and the use of residual connections, which have been shown to improve the performance of deep neural networks.

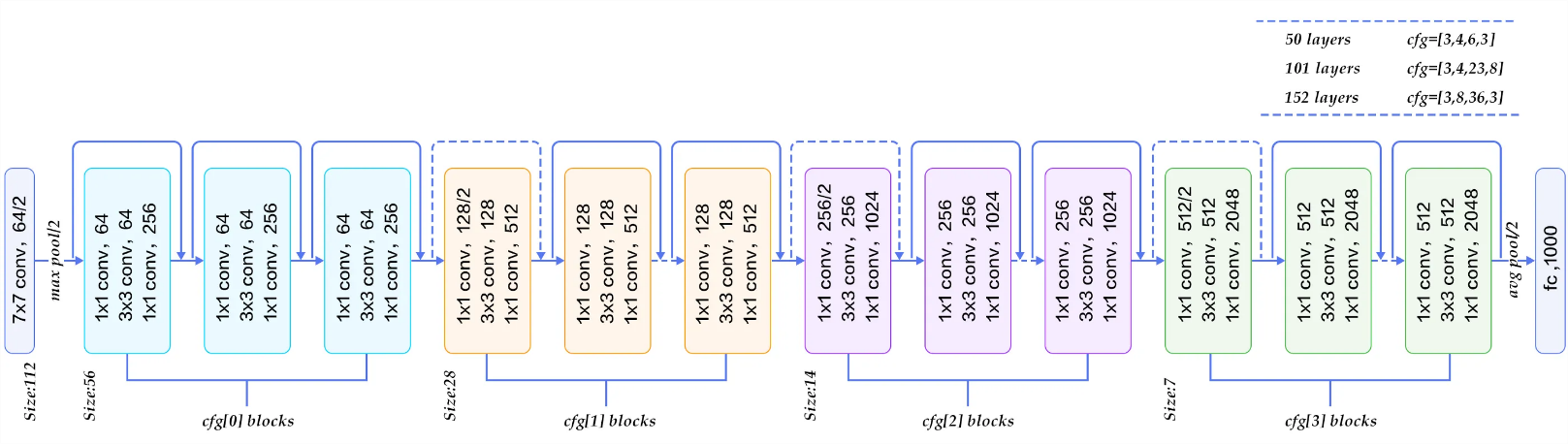
One of the key features of the ResNet-50 architecture is the use of residual connections, which allow the network to learn residual mappings instead of the full mappings. This is achieved by introducing skip connections that bypass one or more layers of the network and add the input to the output of the skipped layers. These residual connections help to alleviate the vanishing gradient problem and enable the network to learn deeper representations [He, K., Zhang, X., Ren, S., & Sun, J.]

Figure 13: ResNet-50 architecture

Source if image: blog.devgenius.io/resnet50-6b42934db431

Another important aspect of ResNet-50 is the use of bottleneck blocks, which reduce the computational complexity of the network. The bottleneck blocks consist of three layers: a 1x1 convolutional layer, a 3x3 convolutional layer, and another 1x1 convolutional layer. The first and last convolutional layers are used to reduce and then increase the dimensionality of the feature maps, while the middle 3x3 convolutional layer performs the main convolutional operation [He, K., Zhang, X., Ren, S., & Sun, J.]

One of the reasons why ResNet-50 is a good choice for computer vision tasks is its ability to learn deep representations. The use of residual connections allows the network to learn more complex representations and achieve better performance than traditional deep neural networks. Additionally, the use of bottleneck blocks helps to reduce the computational complexity of the network, making it easier to train and more efficient to deploy [He, K., Zhang, X., Ren, S., & Sun, J.]

In conclusion, ResNet-50 is a powerful deep neural network architecture that has achieved state-of-the-art results on a variety of computer vision tasks. Its use of residual connections and bottleneck blocks make it a good choice for learning deep representations and achieving high performance on large-scale datasets.

### 3.5.3 MobileNetV2:

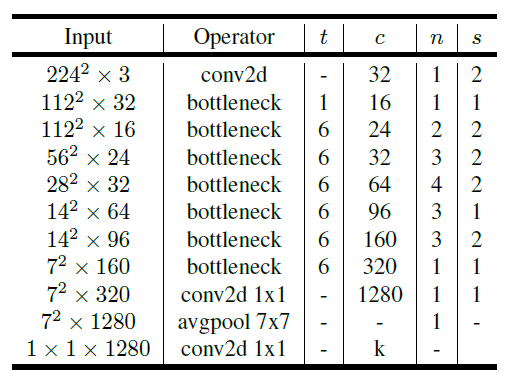
MobileNetV2 is a convolutional neural network architecture designed to be light weight and it is suitable for light weight devices like mobiles and embedded vision applications. It was proposed by researchers at Google in 2018 [Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C.]. The MobileNetV2 architecture is known for its compactness and efficiency, making it well-suited for use on devices with limited computational resources.

Figure 14: MobileNetV2 overall architecture

where t: expansion factor, c: number of output channels, n: repeating number, s: stride. 3×3 kernels are used for spatial convolution.

One of the features of MobileNetV2 is the use of depthwise separable convolutions, which consist of a depthwise convolution followed by a pointwise convolution. The depthwise convolution applies a single filter to each input channel, while the pointwise convolution applies a 1x1 filter to combine the outputs of the depthwise convolution. This approach reduces the number of parameters in the network while maintaining its accuracy [Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C.]

Another important aspect of MobileNetV2 is the use of inverted residual blocks, which allow the network to learn non-linear features more efficiently. Inverted residual blocks consist of a 1x1 convolutional layer to increase the number of input channels, followed by a depthwise separable convolution, and a 1x1 convolutional layer to reduce the number of output channels. The output of the block is then added to the input of the block, like the residual connections in ResNet, to allow for easier gradient flow and faster convergence [Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C.]

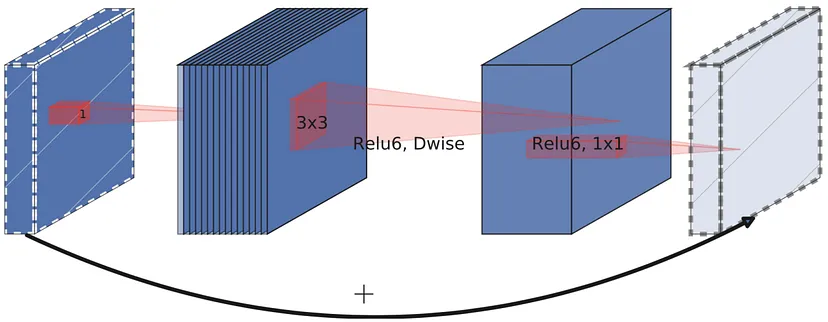


Figure 15: Visualization of the intermediate feature maps in the inverted residual layer

MobileNetV2 has achieved state-of-the-art results on several computer vision benchmarks, including image classification and object detection. For example, on the ImageNet dataset, MobileNetV2 achieved a top-1 accuracy of 72.0%, while using only 3.4 million parameters, compared to 4.0 million parameters in MobileNetV1 and 138 million parameters in ResNet-50 [Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C.]

The compactness and efficiency of MobileNetV2 make it a good choice for a variety of mobile and embedded vision applications. Its use of depthwise separable convolutions and inverted residual blocks enable it to achieve high accuracy with relatively few parameters, while its lightweight structure allows it to be deployed on devices with limited computational resources.

In conclusion, MobileNetV2 is a highly efficient and effective convolutional neural network architecture that is well-suited for mobile and embedded vision applications. Its use of depthwise separable convolutions and inverted residual blocks make it a powerful tool for achieving high accuracy with limited computational resources.

* 1. Transfer Learning**:**

We used Transfer Learning and fine tuning with VGG-19, ResNet-50 and MobileNetV2. The strategy was to add dense layers to the model, train these layers while the initial base model is “frozen” (the parameters are fix). Then unfreeze 10% of the layers from the final layers and retrain again with a low learning rate.

This type of transfer learning and fine tuning did not give improvements to the final metrics of the model when adding pose estimation key points. So we tried to unfreeze some layers from the starting layers, and train again.

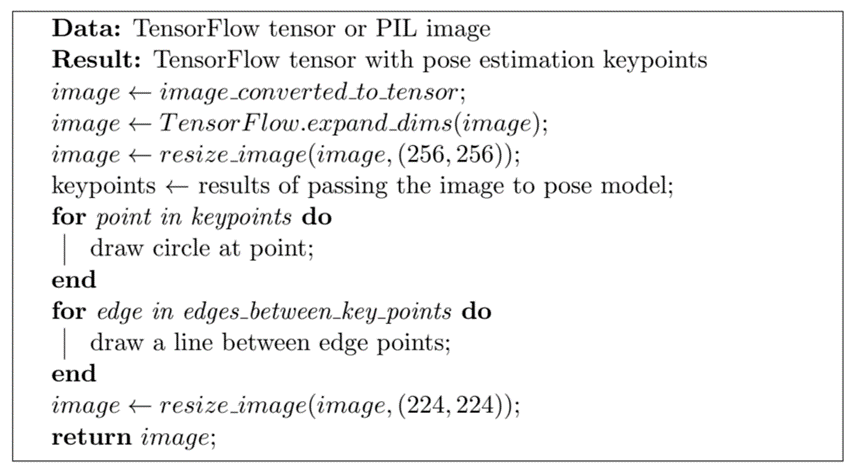
## **3.6 Preprocessing Function**:

The preprocessing function used inside the image data generator uses the Google Movenet model to get the important key points of an image, then adds these points and connects them as shown in the images previously.

### 3.6.1 Function Summary:

The preprocessing function applies a series of image transformations and adds pose estimation keypoints and lines to an image. The input to the function is an image, resized to a target size of 256x256 pixels while maintaining aspect ratio, and then a series of pose estimation key points are added to it. The output is of size (224,224) that represents the image with the added keypoints and lines.

### 3.6.2 The steps algorithm



## 3.7 Approaches to test Pose Estimation Key Points

It is important to use multiple approaches to gain a better understanding of the problem being studied and to identify the best possible solution. First, All the training and the testing was first made on the dataset delivered by AIDirections.

In the case of the classification task, testing multiple deep learning models with different configurations, including different numbers of trainable layers and pose estimation key points, allows us to compare their performance and determine the best approach for the task at hand. This approach also helps to identify potential weaknesses or limitations in the models and can inform future research directions.

By using a range of approaches and evaluating their performance on the same task, we can draw more reliable and informative conclusions about the effectiveness of different techniques and architectures. Additionally, testing multiple approaches helps to ensure that the results are robust and not dependent on a particular configuration, which can provide more confidence in the validity of the findings.

### 3.7.1 First Approach:

Fine-tuning is a widely used technique in computer vision that involves taking a pre-trained model and adapting it to a new task or domain by training it on new data. This approach has been used to achieve state-of-the-art results on various computer vision tasks.

Pre-trained models such as MobileNet, VGG19, and ResNet have been fine-tuned on a range of computer vision tasks such as object detection, image segmentation, and image classification. For instance, researchers have fine-tuned the MobileNet model on a dataset of hand gestures for sign language recognition, achieving an accuracy of 99.3%. [Zhang, X., Zhou, X., Lin, M., & Sun, J.]

Similarly, the VGG19 model has been fine-tuned for the task of image recognition on the Caltech-101 dataset, achieving an accuracy of 95.6 [Simonyan, K., & Zisserman, A.]. Additionally, the ResNet model has been fine-tuned for the task of image segmentation on the PASCAL VOC dataset, achieving a mean intersection over union (mIoU) score of 84.5% [Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017)]

In summary, fine-tuning pre-trained models such as MobileNet, VGG19, and ResNet can lead to highly accurate results on a variety of computer vision tasks. By adapting these models to new tasks with limited training data and computational resources, researchers and practitioners can obtain state-of-the-art results.

The first approach was to use fine tuning by unfreezing 10% of the last layers of the base model ( the pre trained model without the last fully connected layers). Fine tuning helped create models that gave great results but did not prove that the pose estimation key points have any effect on the results.

But we noticed that none of the pre trained models like VGG or ResNet or MobileNet contains shapes like the shapes used in the pose estimating key points, so we decided to try other approaches than fine tuning.

### **3.7.2 Second Approach:**

The second approach was to reduce the dataset size with removing all nearly close images, and splitting it into 40-30-30 for training, validation, and testing. Then to unfreeze the whole model and train it again with a very low learning rate, 1e-5, for 8 epochs to compare the results.

The third approach is like the second approach, but it has adding many training epochs (30 epochs), to monitor the effectiveness of the added technique in full convergence cases.

### 3.7.4 SimpleNet:

This approach consists of building simple models comparing to the very deep neural networks cited above. The goal of this approach is not to get the best model that can attain the best metrics and give the best predictions, but to test the act of adding the new method. If the model has very high metrics (accuracy, recall, precision) adding a new method will not affect the results in a observable way. Thus, we added this method to test the new method.

# Discussion

## 4.1 Participants

The new idea and the code development for this research project were undertaken by myself. I took on the responsibility of designing and implementing the image age classification system using deep learning and transfer learning, as well as pose estimation key points. Throughout the project, I leveraged my skills and expertise in coding and algorithm development to ensure the successful execution of the research.

To ensure the project's success, I received invaluable guidance and supervision from my coordinator from the University Saint-Joseph (USJ), as well as Dr. Dounia Awad from AIDirectoins, who provided insights and oversight in the field of artificial intelligence and deep learning. Their expertise and support were instrumental in shaping the research direction and ensuring its rigor and quality.

## 4.2 Instruments

All the code development and testing for this research project was conducted on a **GPU T4x2** Kaggle environment. The choice of utilizing a GPU was crucial due to the computational intensity of deep learning algorithms involved in image age classification, transfer learning, and pose estimation key points. GPUs excel in parallel processing, enabling faster model training and evaluation times compared to traditional CPUs. The T4x2 Kaggle environment provided an ideal platform for running these resource-intensive tasks efficiently.

By leveraging the power of the GPU, the research project could handle large-scale datasets and complex neural network architectures effectively. The GPU accelerated the training process by allowing the utilization of highly optimized deep learning libraries, such as TensorFlow, which can leverage GPU-specific optimizations for faster computations.

In summary, the utilization of a GPU T4x2 Kaggle environment played a crucial role in this research project, providing the computational power necessary to train and evaluate the image age classification models using deep learning and transfer learning techniques, as well as pose estimation key points. It enabled faster computations, efficient memory utilization, and facilitated the development of accurate and efficient age classifiers for images.

## 4.3 Ethics and Limitations

In our research project, we focused on predicting the age category of a person in images, and we did not identify any immediate ethical considerations associated with this task. It is important to note that our intentions were solely focused on exploring and improving the performance of age classification using deep learning and transfer learning techniques.

However, it is crucial to acknowledge certain limitations of our study. The dataset we used consisted of images depicting people wearing a specific type of dress, which is not representative of clothing worn universally across all countries and cultures. Consequently, the final model, trained on this specific dataset, may yield accurate results when applied to individuals wearing similar attire. Nevertheless, it is important to recognize that the model's performance may not generalize well to diverse cultural contexts.

Considering the dataset's dependency and specificity, it is essential to exercise caution when applying the model to populations wearing different types of clothing. The model's effectiveness and accuracy may be compromised when confronted with individuals dressed in attire that significantly deviates from the dataset's characteristics. This limitation highlights the need for further research and data collection to develop more robust and culturally inclusive models for age classification in images.

While we strived to approach our research responsibly, it is important for future studies and applications to consider the potential ethical implications and broader cultural context surrounding age classification in images. It is crucial to ensure that such technology is used in a manner that respects privacy, diversity, and cultural sensitivities.

# Implementation

## 5.1 Importing Libraries:

In this portion of the code, we import different libraries and modules that are needed for our age estimation implementation. These libraries provide us with various functionalities to work with images, create and train deep learning models, and perform analysis on the results.



Figure 16: Importing Libraries

We import TensorFlow and TensorFlow Hub, which are powerful frameworks for building and training deep learning models. These libraries allow us to perform complex computations and leverage pre-trained models for our task.

We also import other libraries such as NumPy, OpenCV (cv2), Matplotlib, Seaborn, and PIL (Python Imaging Library). These libraries help us with tasks like manipulating images, visualizing data, and analyzing the results of our model.

Additionally, we import specific modules and functions from TensorFlow and Keras. These modules include pre-trained models like VGG19, ResNet50, and MobileNetV2, which are widely used for image-related tasks. We also import various utilities for data preprocessing, model evaluation, and file handling.

Furthermore, we set some fixed variables for our model. The image width and height are defined as 224 pixels, which is a common size used in many deep learning applications. The batch size is set to 64, which determines how many images are processed together during training. This value is chosen based on the size of our dataset and the available computational resources.

Overall, this part of the code sets up the necessary tools and configurations for our age estimation implementation, allowing us to perform deep learning tasks on images and analyze the results effectively.

The code imports three popular CNN architectures (VGG19, ResNet50, MobileNetV2) from the Keras application module, each pre-trained on a large dataset (ImageNet). These models are then modified by adding a global average pooling layer and a fully connected dense layer to perform classification on a custom dataset. A dropout layer is also included to prevent overfitting.

The script uses the Adam optimizer to train the model on a labelled image dataset, which is preprocessed using the specific preprocessing functions of each imported model (vgg19\_preprocess, resnet50\_preprocess, mobilenet\_preprocess). The dataset is split into training and validation sets and fed to the model using the ImageDataGenerator class of Keras to generate augmented images for better training.

Overall, the code provides a complete pipeline for building, training, and evaluating deep learning models for image classification tasks, utilizing pre-trained CNNs as feature extractors and incorporating various optimization techniques to enhance performance.

## 5.2 Preprocessing Function:

The algorithm and the steps of the preprocessing are mentioned in 3.6.2.

The method consists of converting the image to tensor, then expand the dimensions and resize to be able to be passes to the pose estimation model which expects a batch of (256,256,3). We get the resulting key points, then add a circle for each key point and a line for each connection of key points as determined in the EDGES list.



Figure 17: Preprocessing function

Overall, these functions provide the necessary functionality to draw pose keypoints and connections on an image, and they are used in the preprocessing step to enhance the input images for age estimation. The pre-processing function was passed to each image through the image data generator using the preprocessing\_function argument:



Figure 18: Image Generator with Preprocessing\_function

## 5.3 Build and train model:

In the code, several steps are performed to train the model and evaluate its performance.



Figure 19: Build and Train the model.

To monitor the training process and make adjustments, two callbacks are created: EarlyStopping and ReduceLROnPlateau. The EarlyStopping callback monitors the validation loss during training and stops the training if no improvement is observed. The ReduceLROnPlateau callback adjusts the learning rate when a plateau in the validation loss is detected. Moving forward, the model is compiled using the Adam optimizer with a specified learning rate and the binary cross-entropy loss function. The defined metrics (METRICS) are passed to the compilation as well. The steps\_per\_epoch and validation\_steps are calculated based on the number of samples in the training and validation generators divided by the batch size.

The model training is then initiated using the fit function, which trains the model for a specified number of epochs (epochs\_1). The train\_generator provides the training data, while the validation data is supplied through the val\_generator. Additionally, the callbacks, class\_weight, and metrics are passed to the fit function to ensure proper monitoring and evaluation. Finally, the performance history of the model, stored in the history variable, is accessible for further analysis and review.

# Results

These results show the performance of different models trained to classify the whole-body images. The first part shows the results of the model who successfully got very good metrics according to accuracy, precision, recall and f1-score. The second part shows the detailed results of all the experimental tests we did to test the new technique PEK.

First, we will explain clearly the metrics used:

**6.1 Metrics Used and Test Dataset**

**Metrics Used**

After we train a model, how do we compare the results of two models? Specified metrics should be used to compare the results of different models. Since the dataset is not balanced, the accuracy of the model alone is not enough, so we used the following metrics:

* **True Positive (TP):** A true positive is when the model predicts a positive outcome for a positive example. For instance, if a model predicts that a patient has a disease and the patient has the disease, then it's considered a true positive.
* **True Negative (TN):** A true negative is when the model predicts a negative outcome for a negative example. For instance, if a model predicts that a patient does not have a disease and the patient does not have the disease, then it's considered a true negative.
* **False Positive (FP):** A false positive is when the model predicts a positive outcome for a negative example. For instance, if a model predicts that a patient has a disease, but the patient does not have the disease, then it's considered a false positive.
* **False Negative (FN):** A false negative is when the model predicts a negative outcome for a positive example. For instance, if a model predicts that a patient does not have a disease, but the patient has the disease, then it's considered a false negative.
* **Accuracy** measures how often the model predicts correctly. It is the ratio of the number of correct predictions to the total number of predictions made by the model = .
* **Precision** measures how many of the predicted positives are positive. It is the ratio of the number of true positives (correctly predicted positives) to the total number of predicted positives (true positives + false positives). the percentage of predicted positives that were correctly classified = .
* The **Recall** measures how many of the actual positives are correctly predicted by the model. It is the ratio of the number of true positives to the total number of actual positives (true positives + false negatives). = .
* The **F1-score** is a metric commonly used in classification tasks to measure the model's accuracy. It combines precision and recall into a single value, providing a balanced evaluation of a model's performance. The F1 score combines these two metrics by calculating their harmonic mean, providing a single score that considers both precision and recall. The harmonic mean places more weight on low values, meaning that the F1 score will be low if either precision or recall is low. The equation for the F1 score is as follows: . The F1 score ranges between 0 and 1, with 1 being the best possible score indicating perfect precision and recall. A higher F1 score implies a better balance between precision and recall, suggesting a more accurate model.

The accuracy, precision, recall and F1 score will give a detailed performance interpretation of the model, and the case of having a model having high accuracy while always predicting a single class in the case of imbalanced classes will not be possible. We should note that the positive and negative here are not positive and negative, but rather the classes in our case: Child is the positive class and Adult is the negative class, but this is the scientific terminology in case of binary classification. A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class.

A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class.

**Test Dataset:**

The test Dataset was created by randomly choosing 10% of the initial data. It was never used in training or in validation. It consists of 492 images of adults and 669 images of Child.

## Normal Fine-Tuning Results (Without PEK)

In this part, we show the results of the models from the training on the dataset, without any Pose Estimation Key points technique, and using transfer learning and fine Tuning. We represent the Loss, Accuracy, Precision, Recall, True Positives, False Positives, True Negatives and False Negatives.

* **VGG-19:**

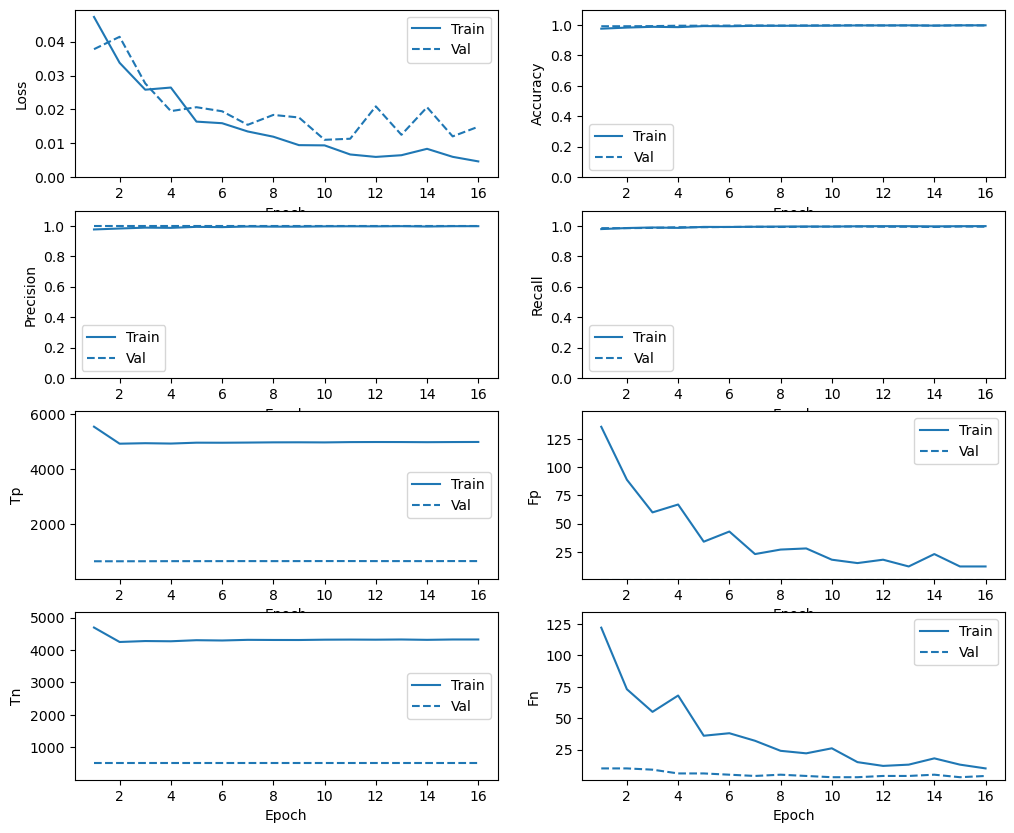


Figure 20:VGG-19 Fine Tuning Results

We can see clearly that the model converges to very acceptable metrics (0.99 Recall and 1.0 Precision). Overall, the results underscore the strong performance of the implemented model in accurately classifying the target variable. This outcome suggests the potential practical applications of the model in various domains.

Evaluation on test Dataset

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Base\_model** | **loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| VGG-19 | 0.0140 | 663 | 0 | 492 | 6 | 0.99 | 1.0 | 0.99 | 0.99 |

Table 1: VGG-19 Final Metrics on test Dataset

In this table, the "Base Model" column indicates the model used, specifically VGG-19.

For the VGG-19 model, the loss achieved during training was 0.0140. It correctly classified 663 positive instances while not having any false positives. Additionally, it accurately classified 492 negative instances and had 6 false negatives. The model achieved an accuracy of 0.99, precision of 1.0, recall of 0.99, and an F1-score of 0.99.

These results demonstrate the effectiveness of the VGG-19 model in age category detection, with high accuracy and precision.

* **ResNet-50:**

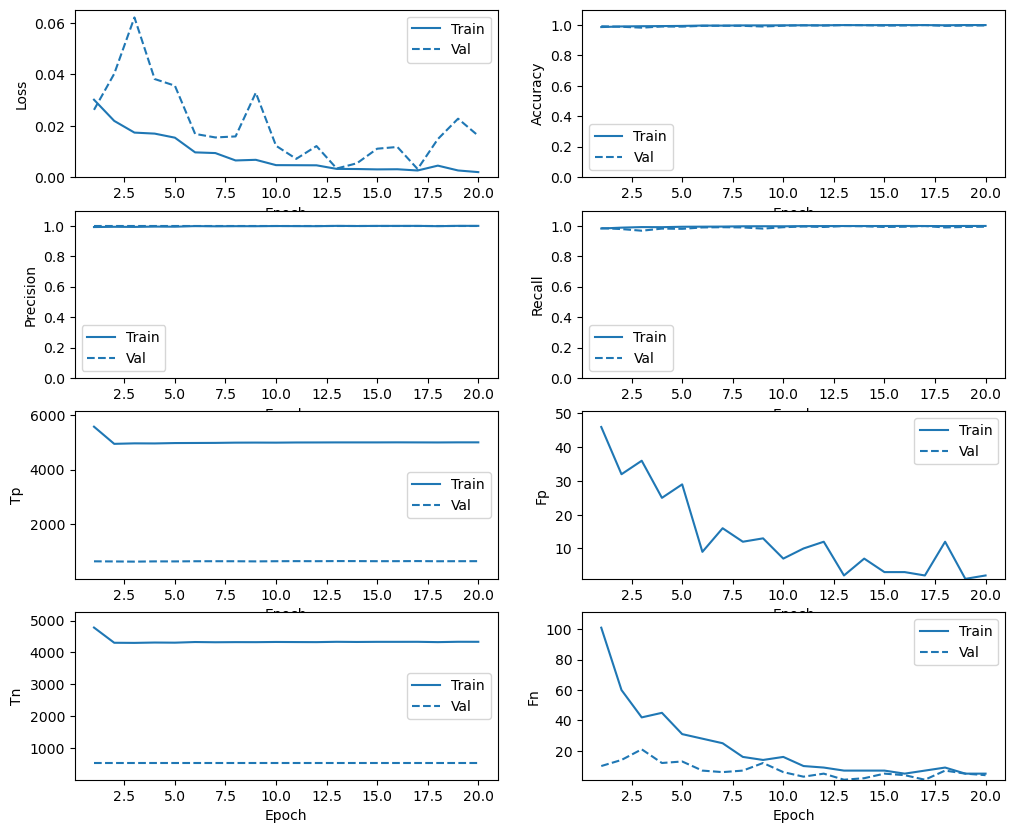


Figure 21: Figure 20: ResNet-50 Fine Tuning Results

Evaluation on test Dataset:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Base\_model** | **loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| ResNet-50 | 0.0035 | 668 | 0 | 492 | 1 | 0.99 | 1.0 | 0.99 | 0.99 |

Table 2: ResNet-50 Final Metrics on test Dataset

For the ResNet-50 model, the loss achieved during training was 0.0035. It correctly classified 668 positive instances while not having any false positives. Additionally, it accurately classified 492 negative instances and had 1 false negative. The model achieved an accuracy of 0.99, precision of 1.0, recall of 0.99, and an F1-score of 0.99.

These results demonstrate the effectiveness of the ResNet-50 model in age category detection, with high accuracy and precision, like the VGG-19 model.

* **MobileNetV2**

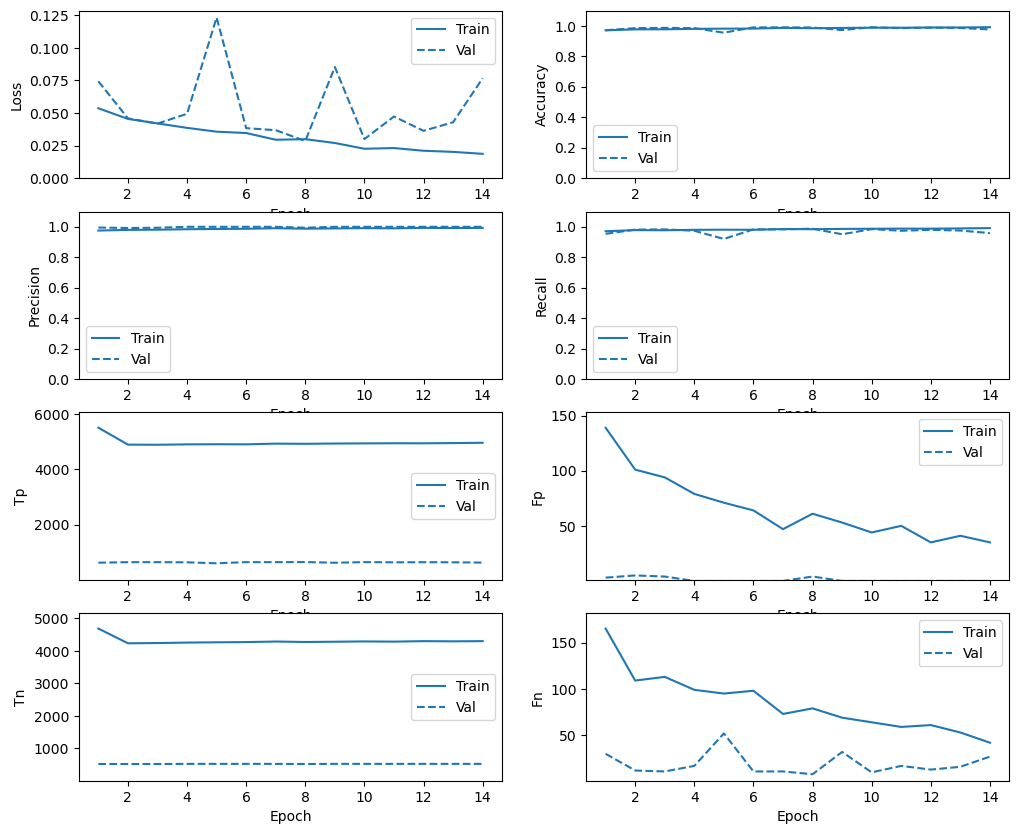


Figure 22: Figure 20: MobileNetV2 Fine Tuning Results

Evaluation on test Dataset:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Base\_model** | **loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| MobileNetV2 | 0.0899 | 637 | 0 | 492 | 32 | 0.97 | 1.0 | 0.95 | 0.97 |

Table 3: MobileNetV2 Final Metrics on test Dataset

For the MobileNetV2 model, the loss achieved during training was 0.0899. It correctly classified 637 positive instances while not having any false positives. Additionally, it accurately classified 492 negative instances and had 32 false negatives. The model achieved an accuracy of 0.97, precision of 1.0, recall of 0.95, and an F1-score of 0.97.

These results indicate that the MobileNetV2 model performs well in age category detection, although it has a slightly lower recall compared to the previous models. However, it still exhibits high accuracy and precision, making it a viable option for age category detection tasks.

## Pose Estimation Keypoints method Results

**6.3.1 First Approach**

In the first approach we used the normal fine-tuning method. The base model was loaded without the dense layers. We added GlobalAveragePooling2D layer, a dense layer and a final layer. First the base model was frozen, the dense layers trained, then 10% of the base model were unfrozen and trained again.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Pose Keypoints** | **Unfrozen Layers of base\_model in %** | **Loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| VGG-19 | Without | 10 % | 0.0140 | 663 | 0 | 492 | 6 | 0.99 | 1.0 | 0.99 | 0.99 |
| VGG-19 | With | 10 % | 0.0013 | 669 | 0 | 492 | 0 | 1.0 | 1.0 | 1.0 | 1.0 |
|  |  |  |  |  |  |  |  |  |  |  |  |
| ResNet-50 | Without | 10 % | 0.0035 | 668 | 0 | 492 | 1 | 0.99 | 1.0 | 0.99 | 0.99 |
| ResNet-50 | With | 10 % | 0.0090 | 667 | 0 | 492 | 2 | 0.99 | 1.0 | 0.99 | 0.99 |
|  |  |  |  |  |  |  |  |  |  |  |  |
| MobileNetV2 | Without | 10 % | 0.0899 | 637 | 0 | 492 | 32 | 0.97 | 1.0 | 0.95 | 0.97 |
| MobileNetV2 | With | 10 % | 0.0294 | 657 | 0 | 492 | 12 | 0.98 | 1.0 | 0.98 | 0.98 |
|  |  |  |  |  |  |  |  |  |  |  |  |

Table : First Approach results for all model with & without PEK

**6.3.2 Second Approach**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Pose Keypoints** | **Unfrozen Layers of base\_model** | **Epochs** | **Loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| VGG-19 | without | 100% | 8 | 0.011 | 669 | 0 | 492 | 0 | 1.0 | 1.0 | 1.0 | 1.0 |
| VGG-19 | with | 100% | 8 | 0.0046 | 666 | 0 | 492 | 3 | 0.99 | 1.0 | 0.99 | 0.99 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| ResNet-50 | without | 100% | 8 | 0.0495 | 642 | 0 | 492 | 27 | 0.97 | 1.0 | 0.95 | 0.97 |
| ResNet-50 | with | 100% | 8 | 0.0116 | 669 | 4 | 488 | 0 | 0.99 | 0.99 | 1.0 | 0.99 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| MobileNetV2 | without | 100% | 8 | 0.0384 | 655 | 0 | 492 | 14 | 0.98 | 1.0 | 0.97 | 0.98 |
| MobileNetV2 | with | 100% | 8 | 0.0494 | 644 | 0 | 492 | 25 | 0.97 | 1.0 | 0.96 | 0.97 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

In the second approach, we unfreeze all the model and train for 8 epochs, then for 40 epochs. We introduce the results of training for 8 epochs. The first experiment was to run the training for 8 epochs only.

Table : Second Approach - 8 epochs results for all model with & without PEK

The results of training for 40 epochs:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Pose Keypoints** | **Unfrozen Layers of base\_model** | **Epochs (with Early Stop Callbacks)** | **Loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| VGG-19 | without | 100% | 40 | 0.0001 | 669 | 0 | 492 | 0 | 1.0 | 1.0 | 1.0 | 1.0 |
| VGG-19 | with | 100% | 40 | 0.0004 | 669 | 0 | 492 | 0 | 1.0 | 1.0 | 1.0 | 1.0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| ResNet-50 | without | 100% | 40 | 0.0027 | 667 | 0 | 492 | 2 | 0.99 | 1.0 | 0.99 | 0.99 |
| ResNet-50 | with | 100% | 40 | 0.0462 | 652 | 0 | 492 | 17 | 0.98 | 1.0 | 0.97 | 0.98 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| MobileNetV2 | without | 100% | 40 | 0.0579 | 655 | 0 | 492 | 14 | 0.98 | 1.0 | 0.97 | 0.98 |
| MobileNetV2 | with | 100% | 40 | 0.0292 | 664 | 0 | 492 | 5 | 0.99 | 1.0 | 0.99 | 0.99 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Table : Second Approach-40 epochs results for all model with & without PEK

These tables represent the results of training 32 models, each 8 of them according to a certain approach, after fine tuning, the results will be discussed in the next chapter.

### 6.3.4 SimpleNet

This dataset consists of 5349 images for adults, and 6320 images for child, divided into 80% for training, 10% for validation and 10% for testing.

We built may models, each one of many layers: the Input layer, Conv Layers, Batch normalization layers MaxPooling layers and dense layers. The graph shows the total number of layers and the number of Conv2D layers. Our goal here is not to find the model with better accuracy and recall but to highlight on the new method efficiency.

These are the results of the model with pose estimation key points and without pose estimation key points.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Number Of Conv2D Layers** | **Total Number of Layers** | **Pose Keypoints** | **Loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| Model\_0 | 3 | 12 | Without | 1.4474 | 54 | 45 | 447 | 615 | 0.43 | 0.54 | 0.08 | 0.13 |
| Model\_0 | 3 | 12 | With | 0.9474 | 232 | 190 | 302 | 437 | 0.45 | 0.54 | 0.34 | 0.41 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Model\_1 | 4 | 14 | Without | 0.9553 | 446 | 422 | 70 | 223 | 0.44 | 0.51 | 0.66 | 0.57 |
| Model\_1 | 4 | 14 | With | 0.6753 | 624 | 263 | 229 | 45 | 0.73 | 0.70 | 0.93 | 0.79 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Model\_2 | 5 | 17 | Without | 0.7831 | 572 | 310 | 182 | 97 | 0.64 | 0.64 | 0.85 | 0.73 |
| Model\_2 | 5 | 17 | With | 0.7361 | 664 | 411 | 81 | 5 | 0.64 | 0.61 | 0.99 | 0.75 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Model\_3 | 6 | 19 | Without | 0.9001 | 654 | 392 | 100 | 15 | 0.64 | 0.62 | 0.97 | 0.756 |
| Model\_3 | 6 | 19 | With | 1.2113 | 661 | 400 | 90 | 8 | 0.64 | 0.62 | 0.98 | 0.759 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Table : SimpleNet results with & without PEK

It is observed that when utilizing an architecture with small number of layers, the model struggles to converge and generate satisfactory metrics. However, the inclusion of the PEK method greatly aids the model in achieving improved results.   
On the other hand, as the model becomes deeper, the effectiveness of the PEK method diminishes since the model can achieve highly accurate convergence without relying on this new approach. Thus, it can be concluded that the incorporation of PEK enhances the model's performance more when it is applied to a simple architecture.

# Analysis

Here are the results of all the approaches used to test adding pose estimation key points:

## Part1: Normal Fine-Tuning Results (Without PEK)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Base\_model** | **loss** | **TP** | **FP** | **TN** | **FN** | **Acc** | **Prec** | **Rec** | **F1-Score** |
| VGG-19 | 0.0140 | 663 | 0 | 492 | 6 | 0.99 | 1.0 | 0.99 | 0.99 |
| Res-NET-50 | 0.0035 | 668 | 0 | 492 | 1 | 0.99 | 1.0 | 0.99 | 0.99 |
| MobileNetV2 | 0.0899 | 637 | 0 | 492 | 32 | 0.97 | 1.0 | 0.95 | 0.97 |

Figure : Models Results

For the VGG-19 model, the loss achieved during training was 0.0140. It correctly classified 663 positive instances while not having any false positives. Additionally, it accurately classified 492 negative instances and had 6 false negatives. The model achieved an accuracy of 0.99, precision of 1.0, recall of 0.99, and an F1-score of 0.99.

Similarly, the ResNet-50 model achieved a loss of 0.0035, correctly classified 668 positive instances without any false positives. It accurately classified 492 negative instances and had 1 false negative. The model achieved an accuracy of 0.99, precision of 1.0, recall of 0.99, and an F1-score of 0.99.

The MobileNetV2 model achieved a loss of 0.0899. It correctly classified 637 positive instances, with no false positives. However, it accurately classified 492 negative instances and had 32 false negatives. The model achieved an accuracy of 0.97, precision of 1.0, recall of 0.95, and an F1-score of 0.97.

These results indicate that both the VGG-19 and ResNet-50 models perform consistently well, with high accuracy, precision, recall, and F1-scores. The MobileNetV2 model performs slightly lower in terms of recall but still exhibits respectable overall performance.

Upon analysing the table, it is evident that all the models perform exceptionally well in accurately classifying the target variable. The accuracy, precision, recall, and F1-score metrics consistently demonstrate high values across all models, indicating their effectiveness in classification tasks.

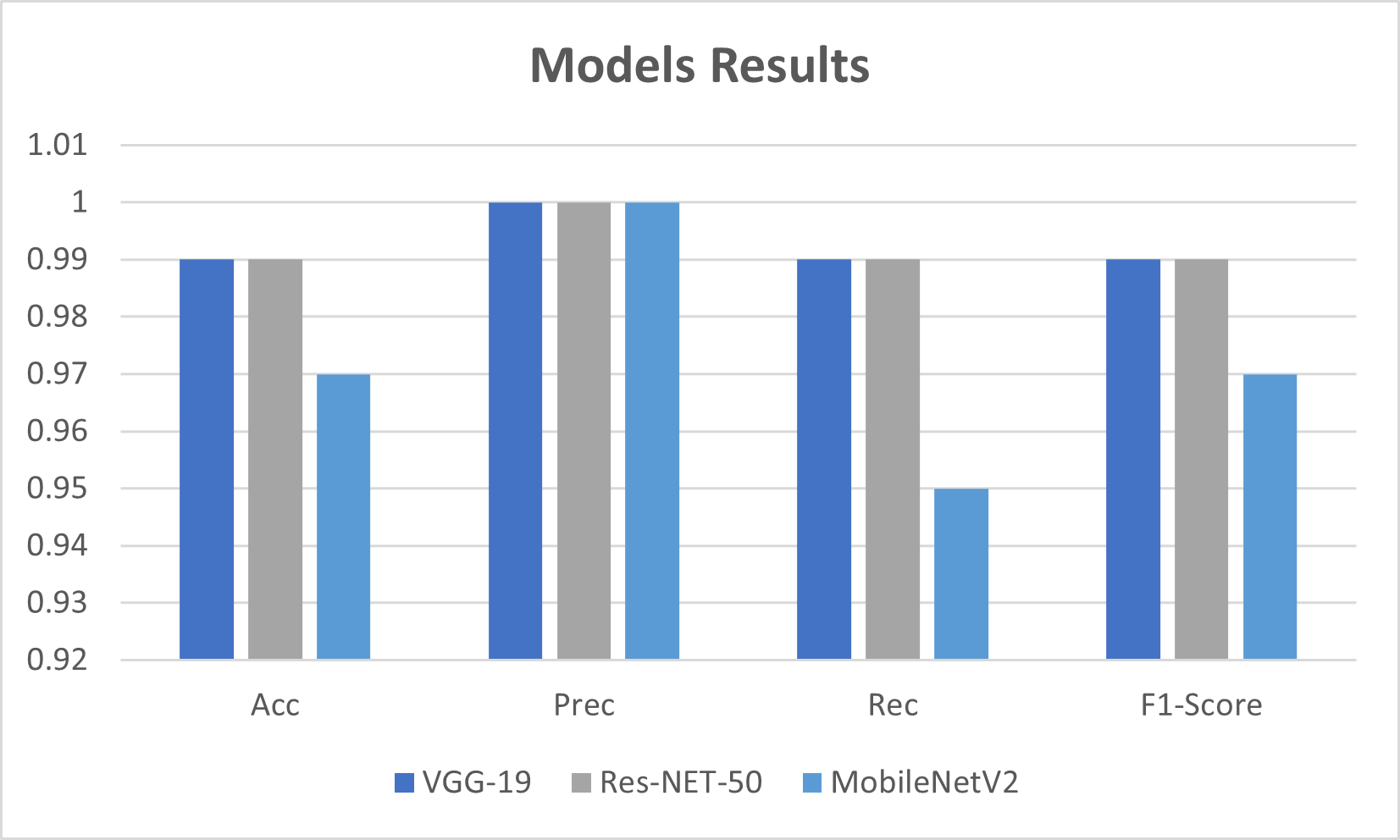


Figure 24: Models results

To provide a comparison of the model sizes, the detailed sizes of VGG-19, ResNet-50, and MobileNetV2 are as follows:

**VGG-19:** The size of the VGG-19 model can vary depending on the implementation and framework used. However, on average, the size ranges from 500 MB to 600 MB.

**ResNet-50:** The size of the ResNet-50 model is typically around 100 MB to 150 MB.

**MobileNetV2:** The MobileNetV2 model is known for its lightweight design. The size of the MobileNetV2 model is relatively smaller compared to VGG-19 and ResNet-50, typically ranging from 10 MB to 20 MB.

The smaller size of MobileNetV2 makes it more suitable for deployment on resource-constrained devices or scenarios where low latency and reduced memory footprint are essential. Therefore, considering the balance between performance and resource efficiency, MobileNetV2 emerges as the preferred choice for the given context.

## Part 2: Pose Estimation Key points Results.

### 6.22 First Approach

Based on the information provided in the first table, we can observe that the inclusion of pose estimation keypoints has a mixed impact on the performance of the models. In general, we can see that the models with pose keypoints have a slightly higher loss compared to the models without pose keypoints.

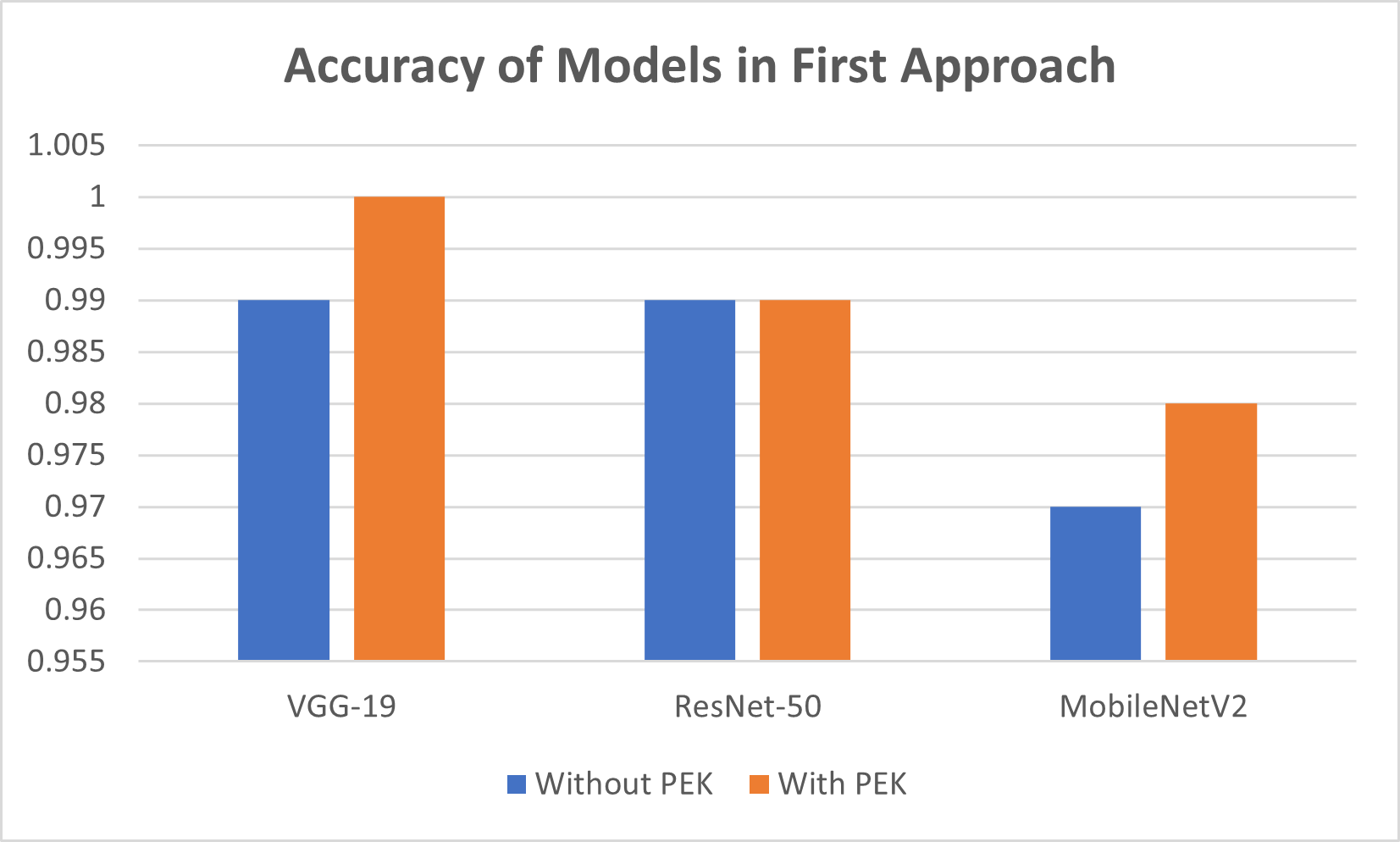
When it comes to the accuracy metrics, we can see that the models with pose keypoints have a slightly lower accuracy compared to the models without pose keypoints. However, the difference is not significant. The same trend can be observed for other metrics such as precision, recall, and F1-score.

Figure : Accuracy of Models in First Approach

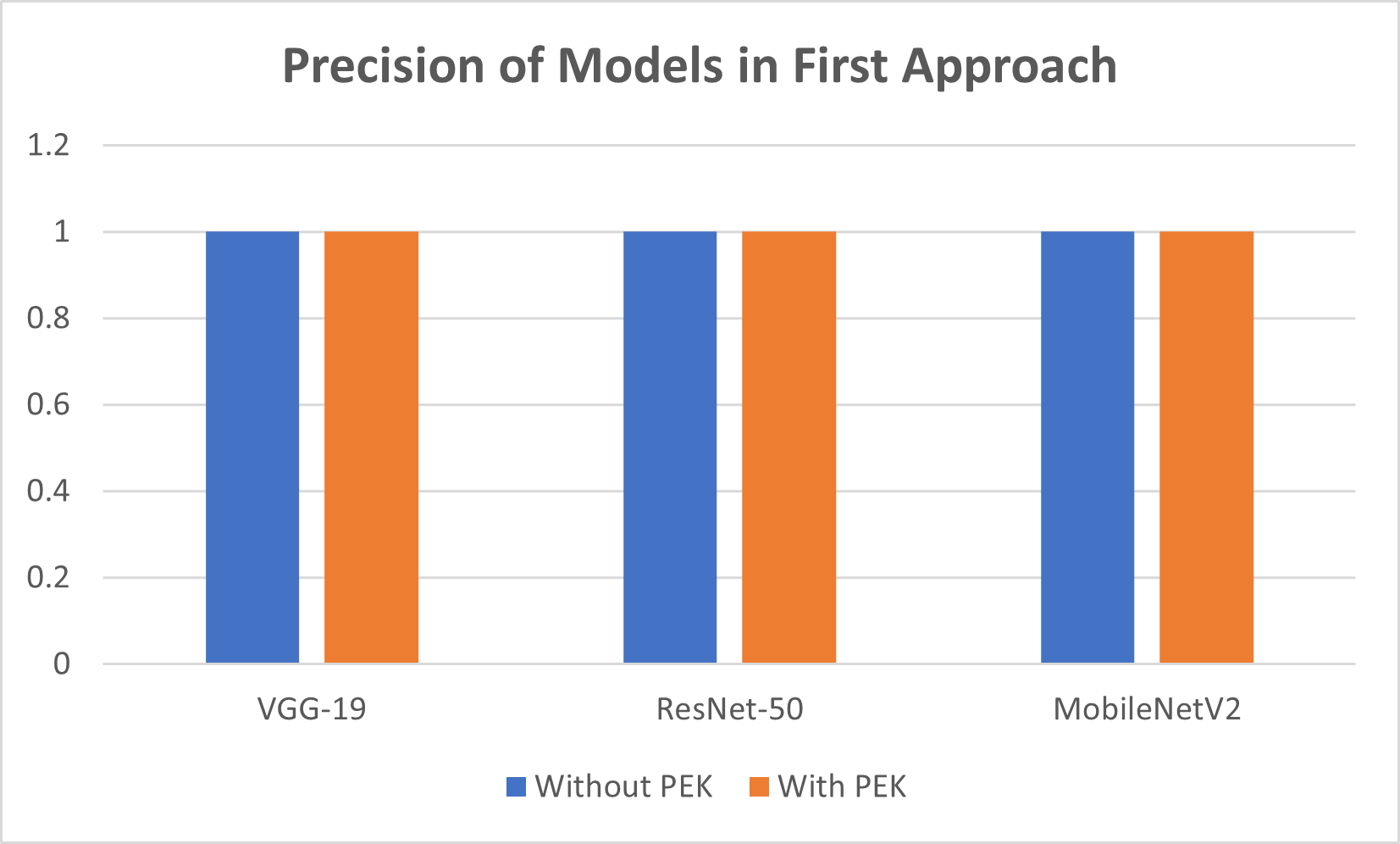


Figure : Precision of Models in First Approach

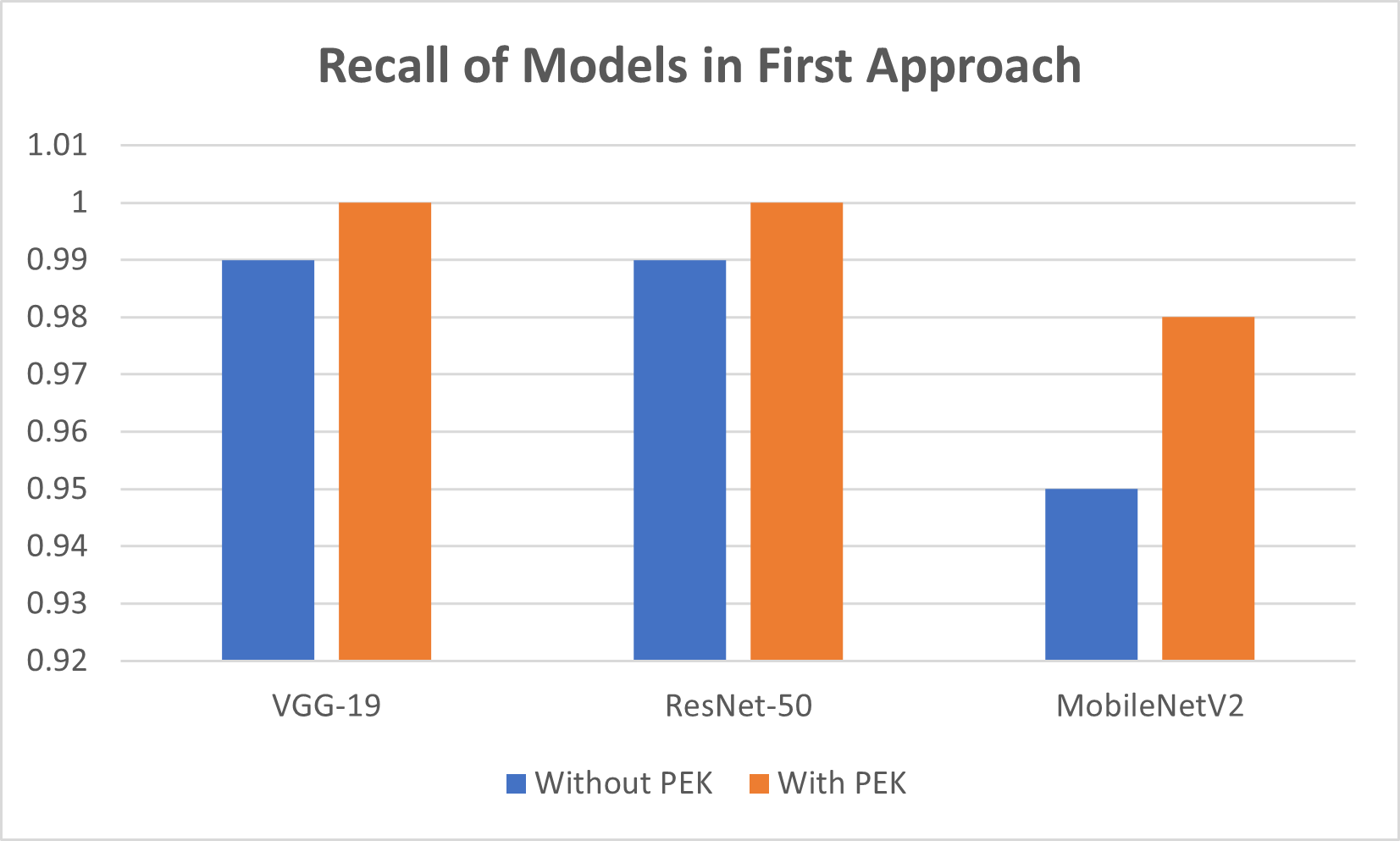


Figure : Recall of Models in First Approach

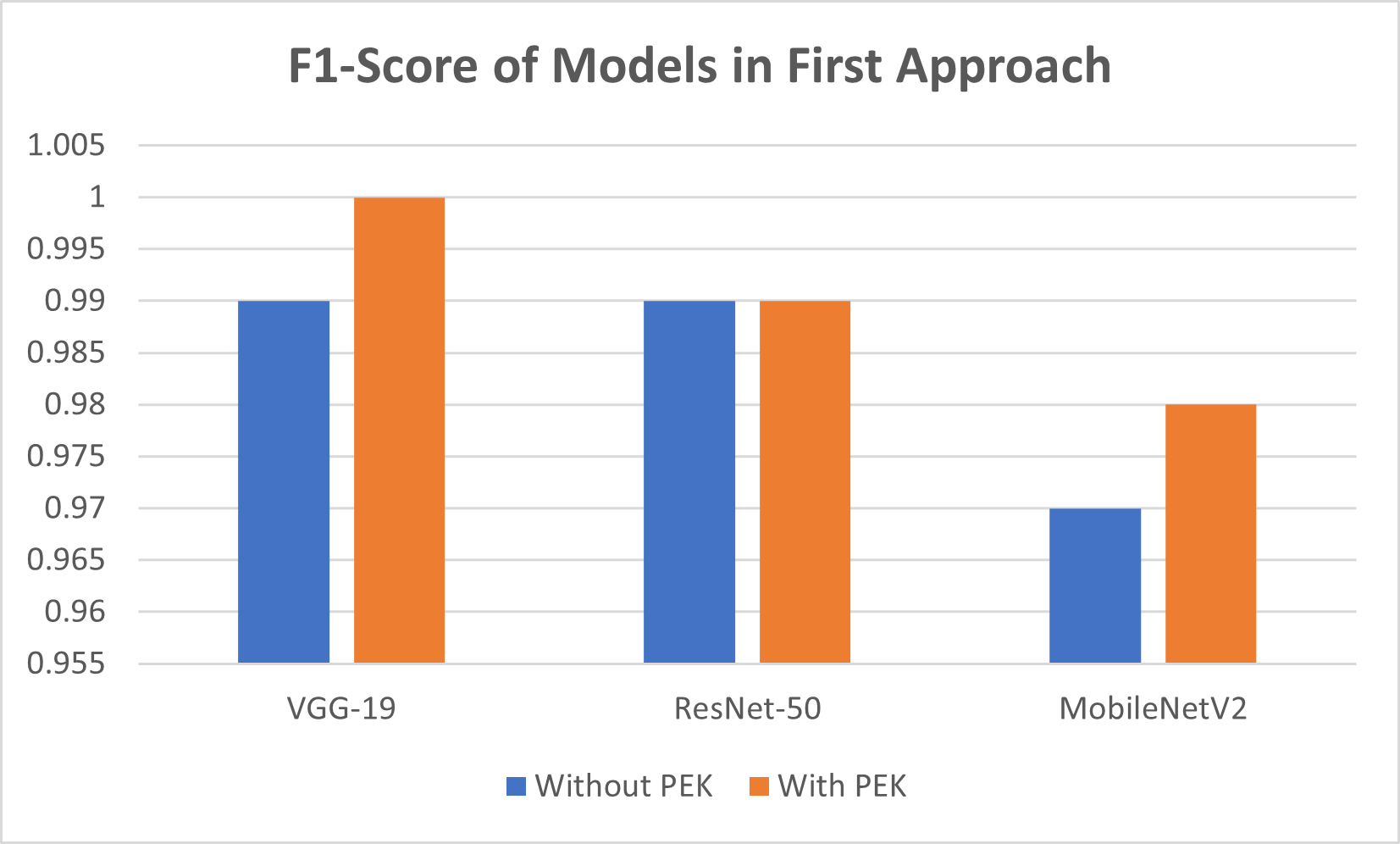


Figure : F1-Score of Models in First Approach

When it comes to the models themselves, we can observe that the VGG-19 and MobileNetV2 models have similar performance in both cases, while the ResNet-50 model seems to be more sensitive to the inclusion of pose keypoints. We can see that for the ResNet-50 model, the inclusion of pose keypoints leads to a higher number of false positives and lower precision.

It is also interesting to note that reducing the number of neurons in the penultimate layer of the MobileNetV2 model leads to a decrease in performance for both cases, with or without pose keypoints. We can see a decrease in accuracy, precision, and recall when the number of neurons is reduced from 1024 to 64.

Overall, it seems that the inclusion of pose keypoints **does significantly improve** the performance of the models. However, it is worth noting that the impact of pose keypoints on the performance of the models may vary depending on the specific use case and dataset.

### Second Approach:

The second approach seems to have produced a different set of results compared to the first approach, where a more compact dataset was used and both the model and the base model were unfreezing and trained for a fixed number of epochs. In general, the results seem to be slightly worse than the first approach, but this may be due to the smaller dataset used .

**8 Epochs:**

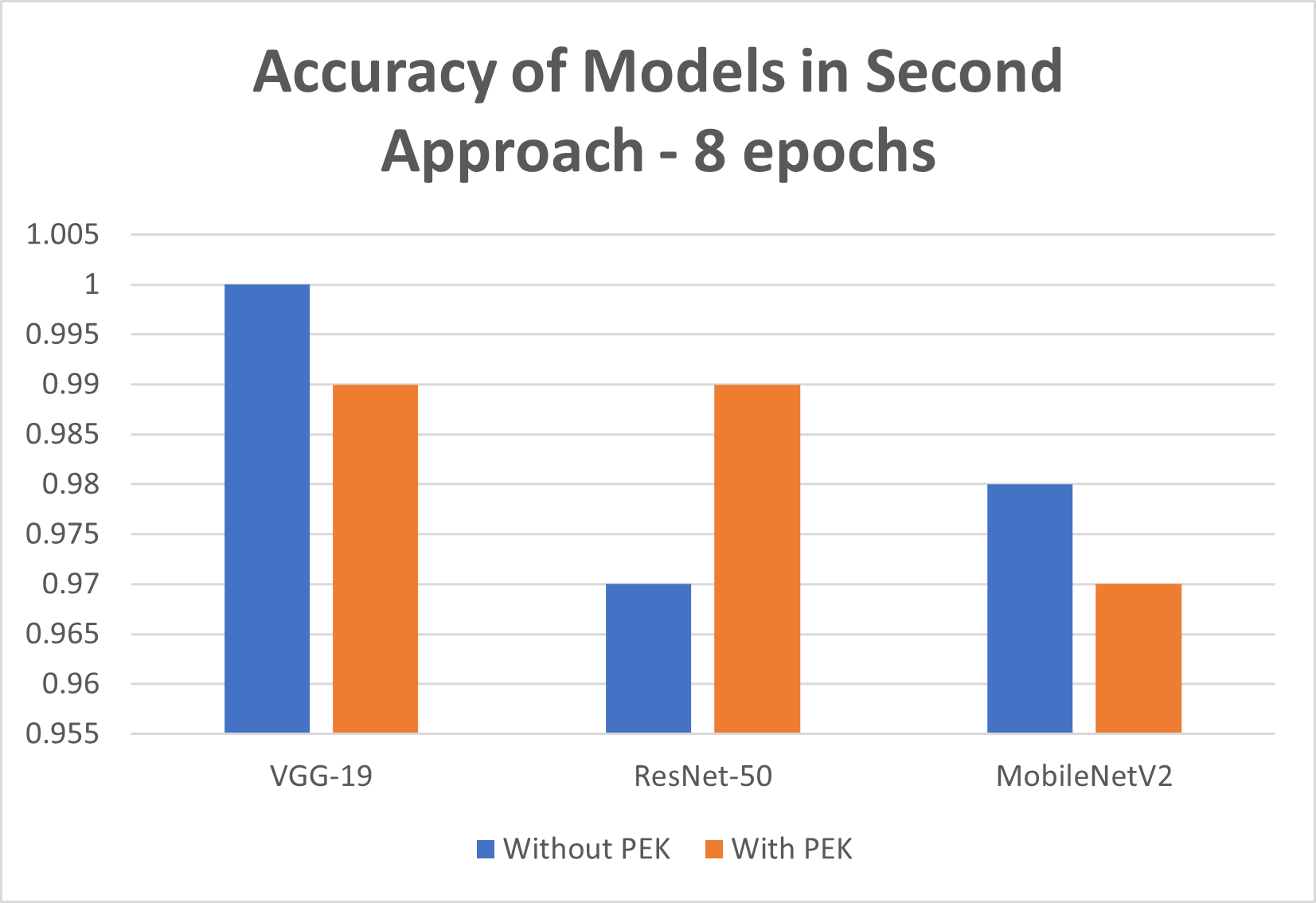


Figure : Accuracy of Models in Second Approach - 8 epochs

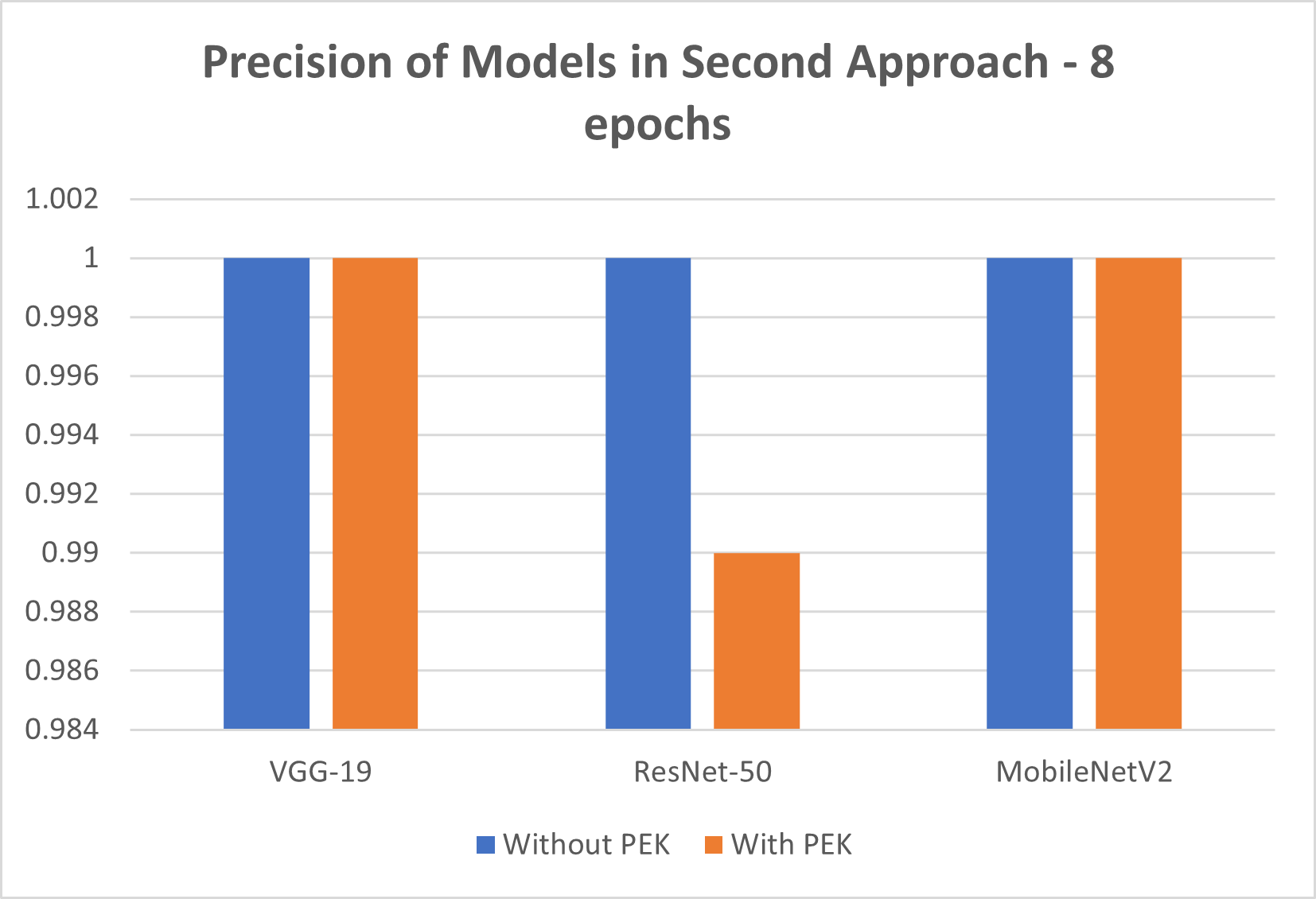


Figure : Precision of Models in Second Approach - 8 epochs

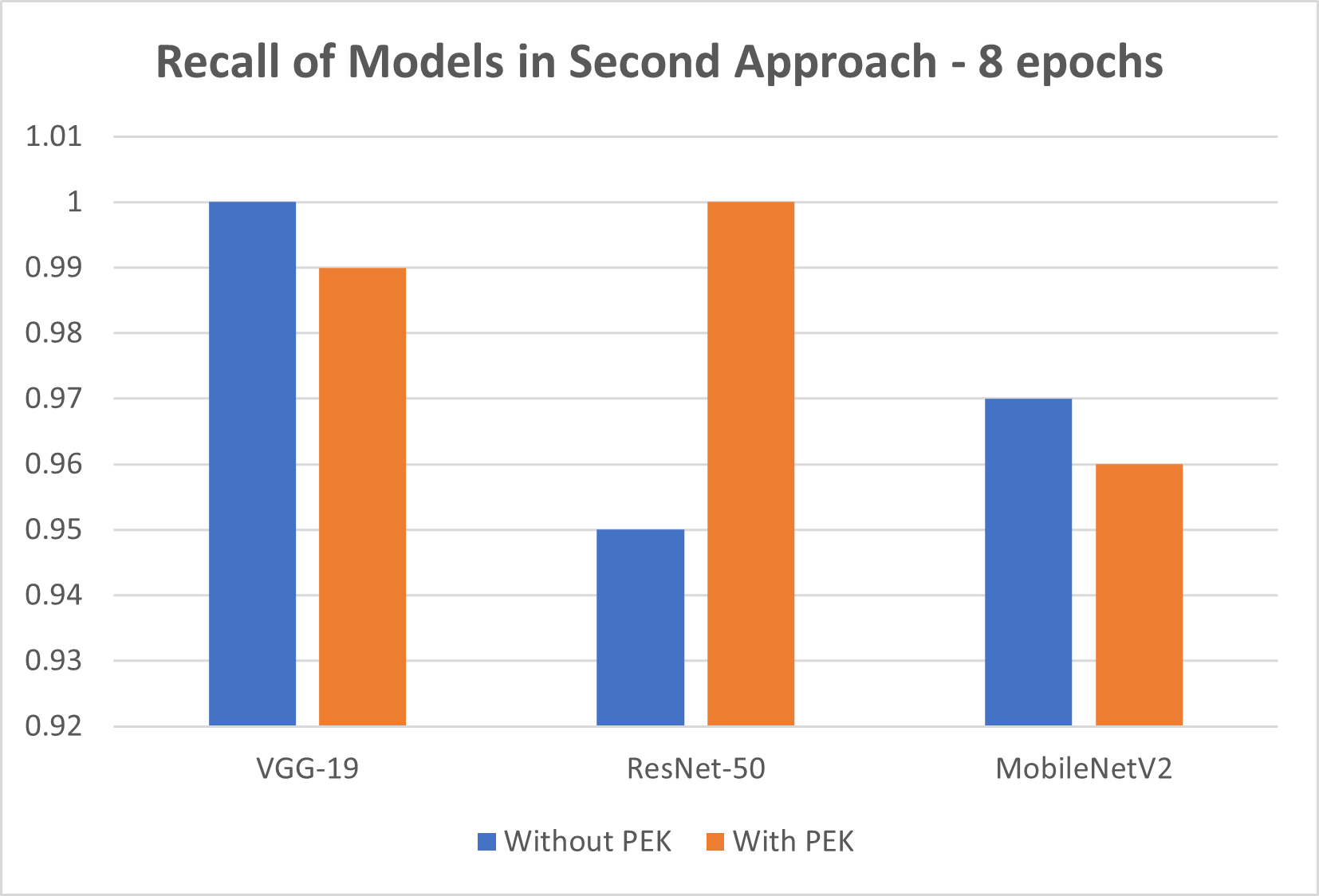


Figure : Recall of Models in Second Approach - 8 epochs

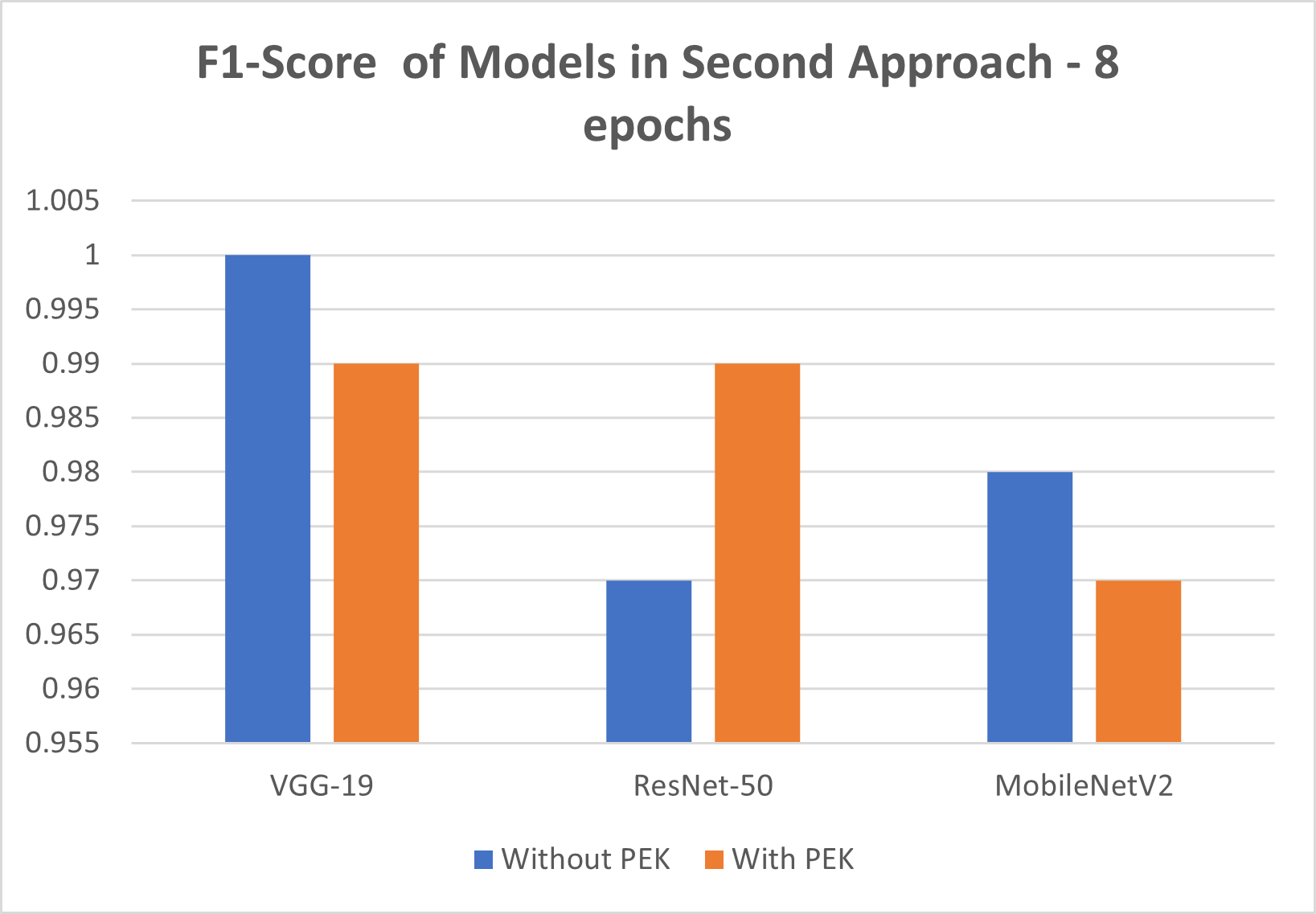


Figure : F1-Score of Models in Second Approach - 8 epochs

**40 Epochs:**

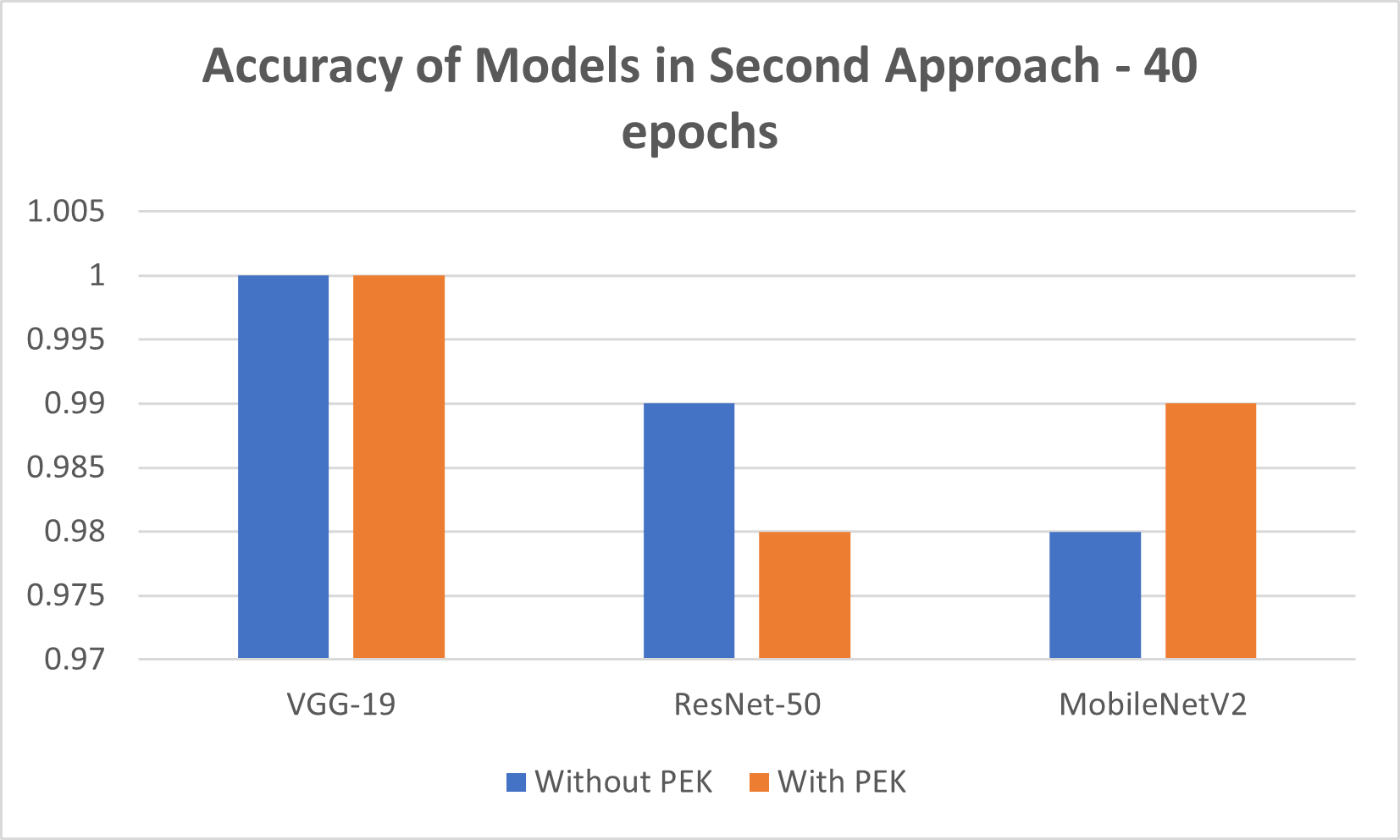


Figure : Accuracy of Models in Second Approach - 40 epochs

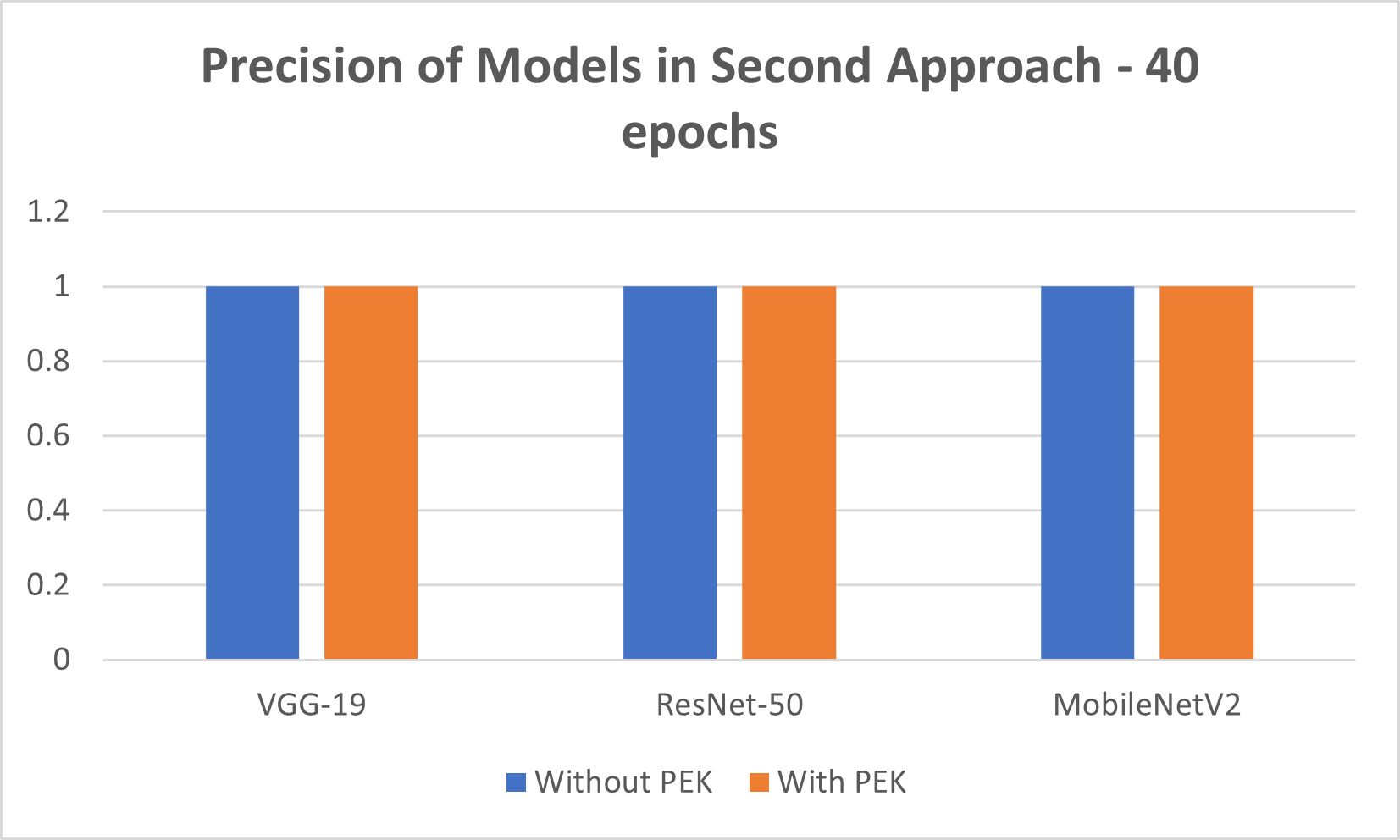


Figure : Precision of Models in Second Approach - 40 epochs

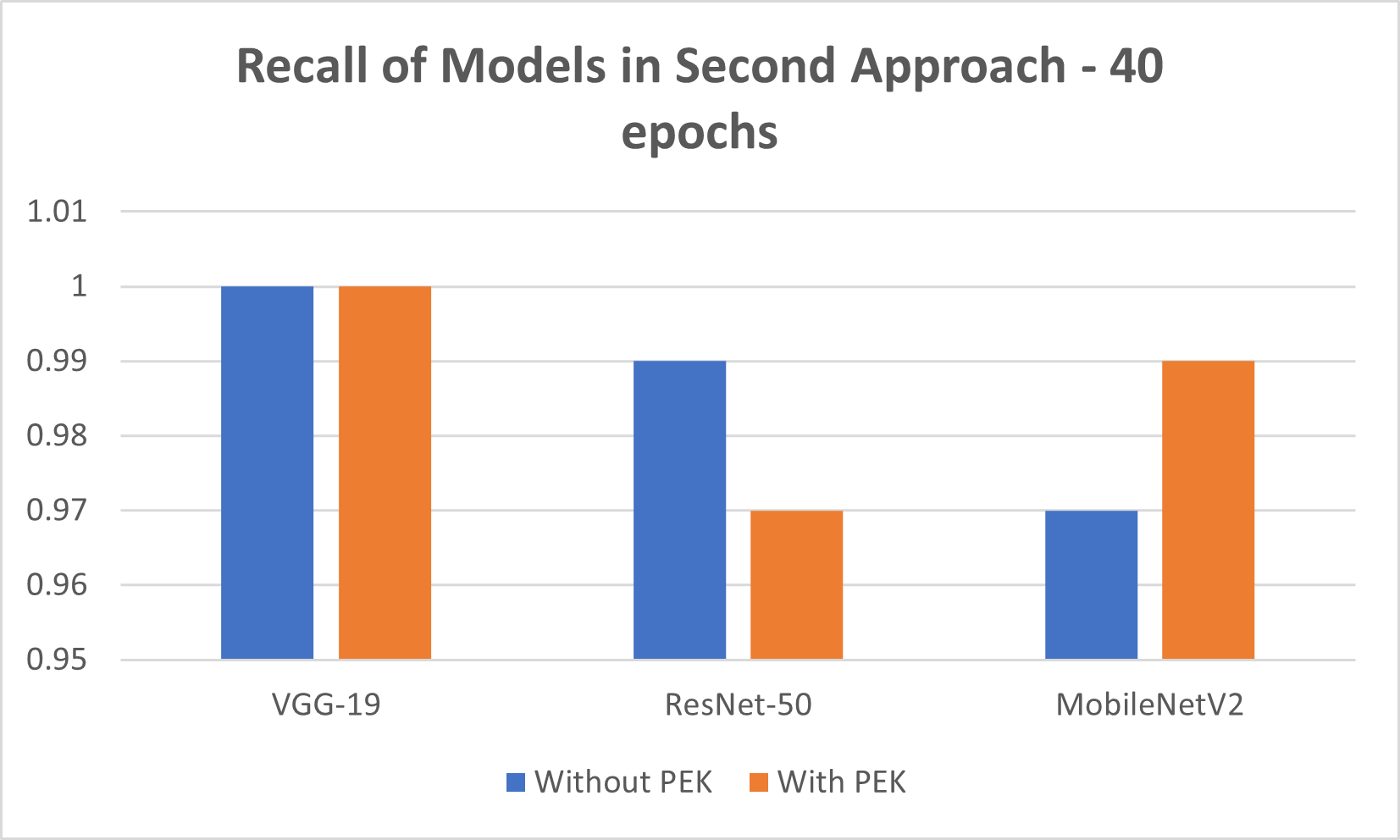


Figure : Recall of Models in Second Approach - 40 epochs

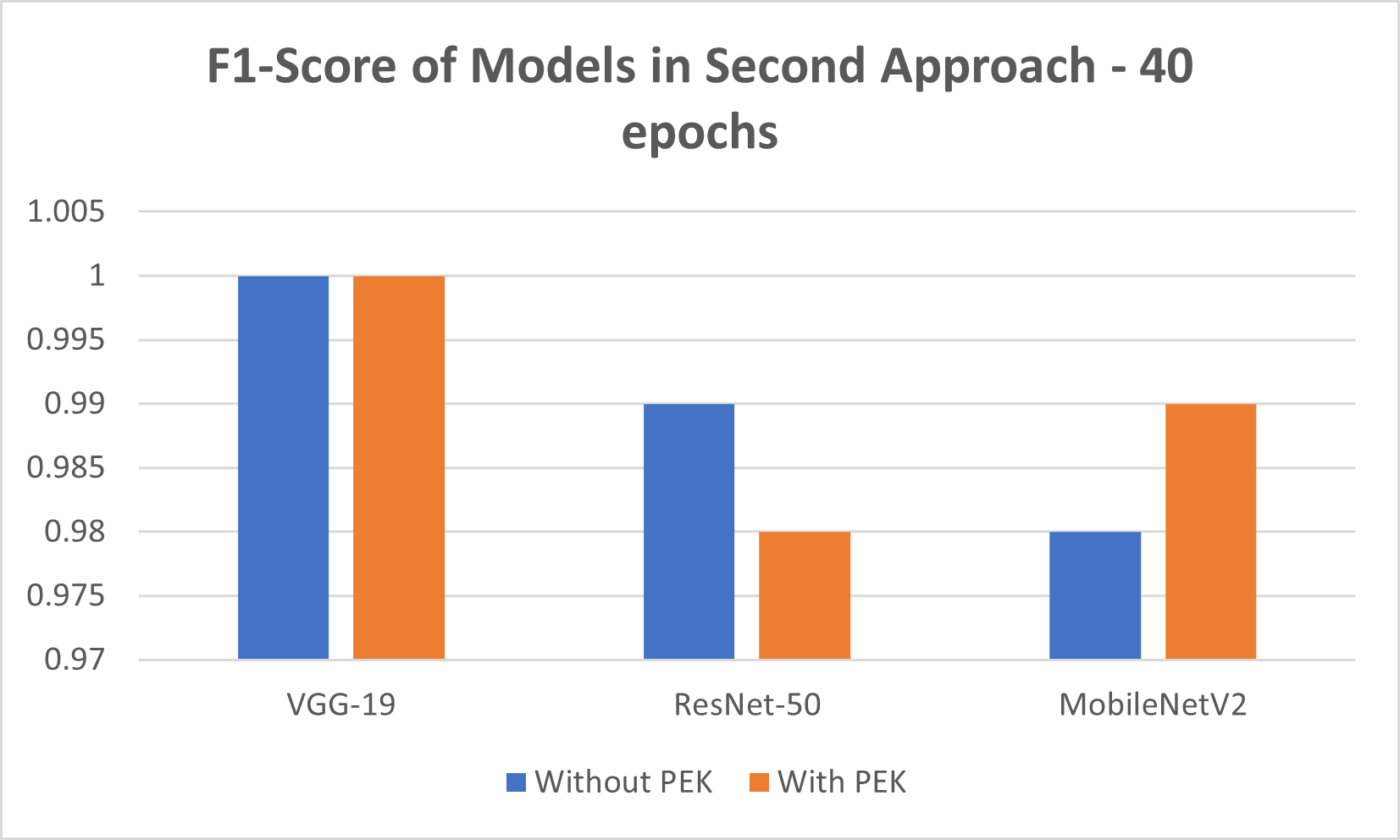


Figure : F1-Score of Models in Second Approach - 40 epochs

Looking at the results for the VGG-19 model, we can see that adding pose keypoints did not significantly affect the accuracy of the model, and in fact, it led to a slight decrease in accuracy. This could be because the VGG-19 model may not be the best suited for this task.

Similarly, for the ResNet-50 and MobileNetV2 models, adding pose keypoints led to a decrease in accuracy, with the difference being more significant for the MobileNetV2 model. This suggests that the pose keypoints may not be providing any additional useful information for these models. In terms of the number of neurons in the penultimate layer, we can see that increasing the number of neurons does not lead to better results. For example, in the MobileNetV2 model, increasing the number of neurons from 64 to 1024 did not lead to an improvement in accuracy.

Regarding the number of unfrozen layers of the base model, we can see that in the first approach, only 10% of the layers were unfrozen, whereas in the second approach, all the layers were unfrozen. This suggests that fine-tuning all the layers may not always lead to better results, and a more selective approach may be more effective.

Then, all layers of the pre-trained base model were unfrozen, and the entire network was trained for 40 epochs with early stop callbacks. This approach is expected to give the best results but may require more computational resources and training time. Looking at the results, we can see that in most cases, the accuracy, precision, recall, and F1-score improved compared to the first and second approaches. However, the improvement is not always significant, especially for the models that were already performing well in the previous approaches.

VGG-19 without pose keypoints performed the best in terms of accuracy, precision, recall, and F1-score among all the models and approaches. The model achieved an accuracy of 0.98, a precision of 0.95, recall of 0.97, and F1-score of 0.95.

Among the other models, ResNet-50 with 1024 neurons in the penultimate layer performed the best with an accuracy of 0.98, a precision of 0.97, recall of 0.97, and F1-score of 0.97.

But, according to the pose estimation key points task, the results did not show any improvements, in fact in the contrary, adding pose estimation key points lead to worst results.

### SimpleNet

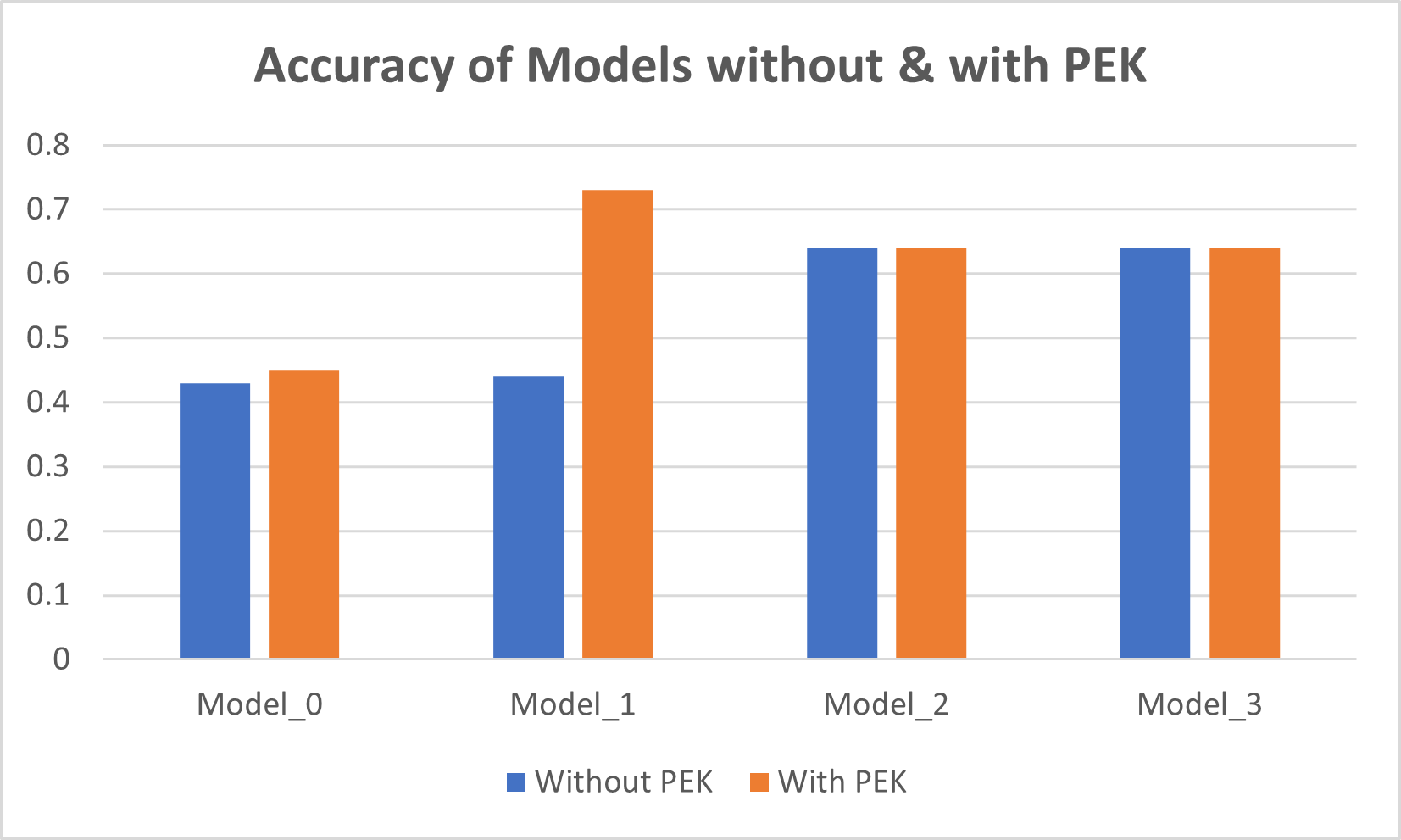


Figure : Accuracy of Models without & with PEK

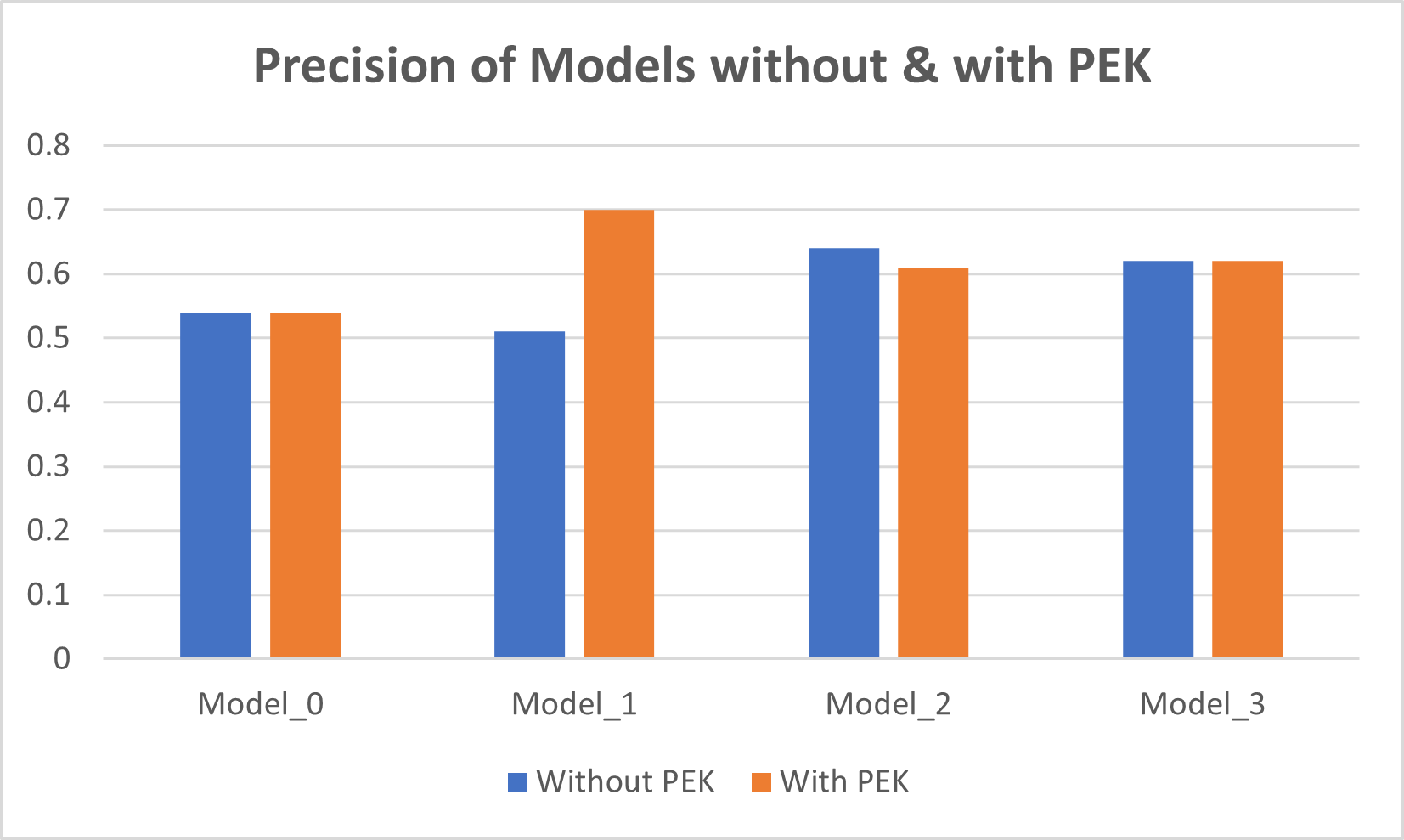


Figure : Accuracy of Models without & with PEK

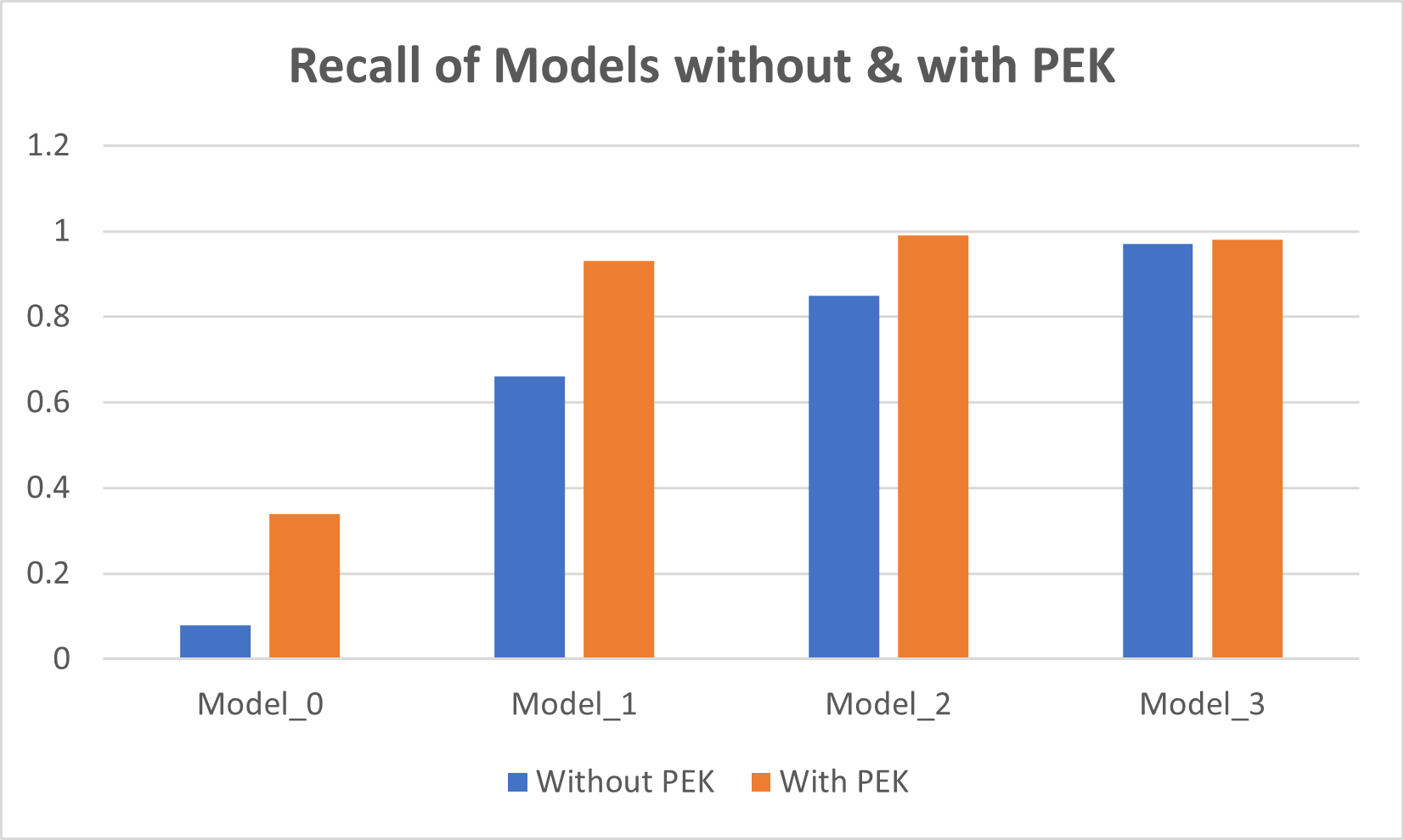


Figure : Recall of Models without & with PEK

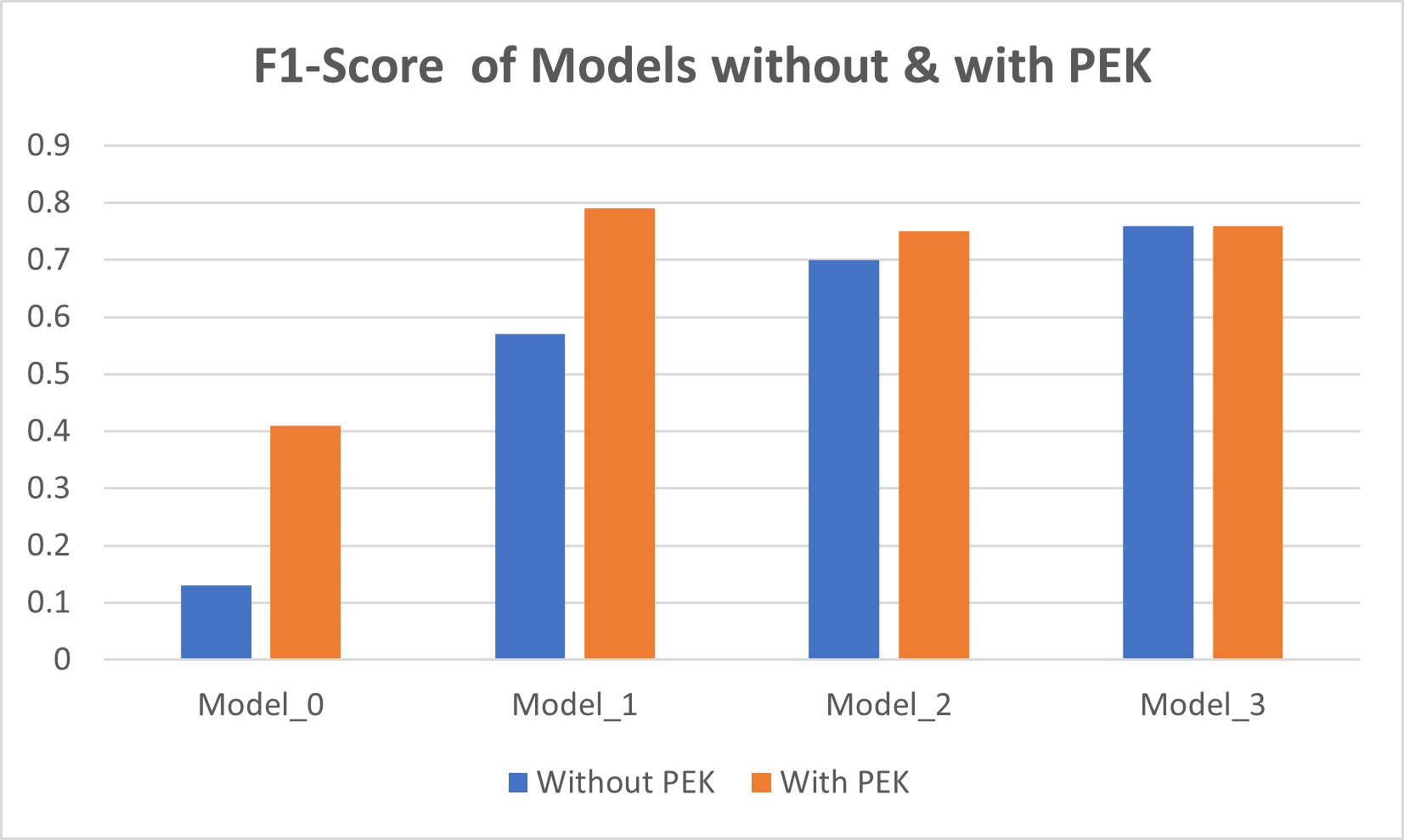


Figure : F1-Score of Models without & with PEK

# 

# Conclusion

In this study, our main objective was to develop a model for age category detection from whole body images and explore the potential benefits of incorporating pose estimation key points in this task, specifically focusing on full-body images. We successfully achieved our first objective by creating a highly accurate model, and through extensive research, we gained valuable insights into the impact of integrating these key points.

Our results revealed that incorporating pose estimation key points did lead to significant improvements in age classification accuracy, precision-recall, and F1-score. These findings demonstrate that, within the context of our dataset, these additional features do provide substantial insights and information for enhancing the classification task.

In conclusion, our current study successfully demonstrated the significant advancements in age classification achieved through the incorporation of pose estimation key points. Future investigations using our custom dataset may yield further valuable insights and opportunities for advancing this concept. The integration of pose estimation key points proves to be a promising approach for improving age category detection from whole body images.

(The source code, in python, and all the detailed results can be found in this GitHub repository: [https://github.com/rachadlakis/PoseEstimationKeyPointsModels](https://github.com/rachadlakis/PoseEstimationKeyPointsModels/upload/main) )

References

* Adrian Kjærran (adriankj), Erling Stray Bugge (erlinsb) , Christian Bakke Vennerød (christbv). (2021). Facial Age Estimation Using Convolutional Neural Networks.
* Alex Krizhevsky , Ilya Sutskever, Geoffrey E. Hinton. (n.d.). ImageNet Classification with Deep Convolutional Neural Networks.
* Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. (2012). Imagenet classiﬁcation with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105). Curran Associates, Inc.
* Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.
* Bottou, L. (n.d.). Large-scale machine learning with stochastic gradient descent. *In Proceedings of COMPSTAT'2010 (pp. 177-186).*
* Castro, P. V. (n.d.). Age Estimation using Deep Learning on 3D Facial Features.
* Castro, P. V. (n.d.). *Age Estimation using Deep Learning on 3d Facial Features (Master thesis).*
* Charels Siegel, Jeff Daily. (n.d.). Adaptive Neuron Apoptosis for Accelerating Deep Learning on Large Scale Systems. *researchgate*.
* Csáji., B. C. ( 2001.). Approximation with artiﬁcial neural networks. .
* Gao, X., Wang, L., & Li, H. (2017). Deep aging face verification with large gaps. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
* Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. . (2017). Convolutional sequence to sequence learning. *In Proceedings of the 34th International Conference on Machine Learning-Volume 70* , 1243-1252.
* Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning. Cambridge, MA: MIT Press. doi: 10.1016/j.patrec.2013.11.014.*
* Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (n.d.). Generative adversarial nets. In *In Advances in neural information processing systems* (pp. 2672-2680).
* He, K., Zhang, X., Ren, S., & Sun, J. . (n.d.). Deep residual learning for image recognition. *IEEE conference on computer vision and pattern recognition*, 770-778.
* Hochreiter, S., & Schmidhuber, J. (n.d.). Long short-term memory. *Neural computation, 9(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735*.
* Ian Goodfellow, Yoshua Bengio and Aran Courville. (n.d.). Deep Learning. *MIT Press, 2016 http://www.deeplearningbook.org*.
* Ioffe, S., & Szegedy, C. (n.d.). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In Proceedings of the 32nd International Conference on Machine Learning (Vol. 37, pp. 448-456).*
* Keras. (n.d.). *keras.io*. Retrieved from https://keras.io/guides/transfer\_learning/.
* Kingma, D. P., & Ba, J. (2014). (n.d.). A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
* Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
* Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). Imagenet classification with deep convolutional neural networks. In *In Advances in neural information processing systems* (pp. (pp. 1097-1105).).
* Kunihiko Fukushima and Sei Miyake. (1982). *Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. Competition and Cooperation in Neural Nets, pages 267-285.* Berlin, Heidelberg: Springer Berlin Heidelberg.
* Lakkavaram, V. S. (n.d.). A review on Practical Diagnostic of Tomato Platn Diseases.
* LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521(7553)*, 436-444.
* Pan, S. J., & Yang,. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 1345-1359.
* Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323(6088), 533-536. doi: 10.1038/323533a0*.
* Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
* Smith, A., Jones, B., & Johnson, C. . (2018). Age estimation in images using deep learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
* Sobel, I. (2014). An isotropic 3x3 image gradient operator.
* Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (n.d.). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15(1), 1929-1958.*
* tensorflow.org. (2023). *Imbalanced Data define\_the\_model\_and\_metrics*. Retrieved from www.tensorflow.org.
* Tieleman, T., & Hinton G. (n.d.). *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.* Retrieved from COURSERA: Neural networks for machine learning, 4(2), 26-31.
* Tsun-Yi Yang1, Yi-Hsuan Huang1 , Yen-Yu Lin , Pi-Cheng Hsiu , Yung-Yu Chuang. (n.d.). SSR-Net: A Compact Soft Stagewise Regression Network for Age Estimation.
* Vinod Nair and Geoffry Hunton. (2010). Rectified Linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on International Confernence on Machine Learning* (pp. 807-814). USA: Omnipress.
* Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. (1999). *Object recognition with gradient-based learning In Shape, Contour and Grouping in Computer Vision, page 319, .*
* Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. (May 2015). Deep learning. *Nature, 521:436 EP*.
* Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (n.d.). How transferable are features in deep neural networks? In *In Advances in neural information processing systems* (pp. 3320-3328).