Unconstrained and Constrained Ear Recognition Using Deep Learning Architectures

Sameh Makkie

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
July 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Zeki Bayram
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Önsen Toygar
Supervisor

Examining Committee

1. Prof. Dr. Önsen Toygar

2. Assoc. Prof. Dr. Yıltan Bitirim

3. Assoc. Prof. Dr. Mehtap Köse Ulukök

ABSTRACT

For a long time now, a means to identify individuals by their physical traits has become crucial. Indeed, identification can be the difference between freedom and imprisonment, it can be the difference between a safer world or a more dangerous one. While each biometric has its own advantages and disadvantages, the ear, in particular, is one of, if not the most useful for forensic investigators because of its slowly changing nature, empowering them can mean empowering justice. This research proposes a new deep-learning model, that is trained and tested on both constrained (AMI) and unconstrained (AWE, EarVN) ear databases of individuals. The model takes an ear image as an input, then it outputs the predicted identity of the individual from the database it was trained on. The proposed approach takes advantage of various factors that affect a deep learning model's accuracy. The most significant of which are the feature extractor, image augmentation, regularization, optimizer and fine-tuning. This research develops the model with these factors in mind and explores finding the best combination of these techniques to maximize the accuracy and robustness of the system. The developed approach has shown very promising results, consistently identifying at least 94.7% of individuals in the EarVN database and even higher in the other two.

Keywords: Deep Learning, Computer Vision, Biometrics, Constrained Ear Recognition, Unconstrained Ear Recognition

ÖZ

Uzun bir süredir, bireyleri fiziksel özelliklerine göre tanımlamanın bir yolu çok önemli hale geldi. Gerçekten de kimlik saptama, özgürlük ve tutsaklık arasındaki fark olabilir, daha güvenli bir dünya ile daha tehlikeli bir dünya arasındaki fark olabilir. Her biyometriğin kendi avantajları ve dezavantajları olsa da, özellikle kulak, yavaş değişen doğası nedeniyle adli tıp araştırmacıları için en yararlı olmasa da biridir, onları güçlendirmek adaleti güçlendirmek anlamına gelebilir. Bu araştırma, bireylerin hem kısıtlanmış (AMI) hem de kısıtlanmamış (AWE, EarVN) kulak veritabanları üzerinde eğitilmiş ve test edilmiş yeni bir derin öğrenme modeli önermektedir. Model, bir kulak görüntüsünü girdi olarak alır, ardından eğitildiği veri tabanından bireyin tahmin edilen kimliğini çıkarır. Önerilen yaklaşım, bir derin öğrenme modelinin doğruluğunu etkileyen çeşitli faktörlerden yararlanır. Bunların en önemlileri, öznitelik çıkarıcı, görüntü çoğaltma, düzenleme, optimize edici ve ince ayardır. Bu araştırma, modeli bu faktörleri göz önünde bulundurarak geliştirir ve sistemin doğruluğunu ve sağlamlığını en üst düzeye çıkarmak için bu tekniklerin en iyi kombinasyonunu bulmayı araştırır. Geliştirilen yaklaşım, EarVN veri tabanındaki bireylerin en az %94,7'sini tutarlı bir şekilde ve diğer ikisinde daha da yüksek bir şekilde tanımlayarak çok umut verici sonuçlar göstermiştir.

Anahtar Kelimeler: Derin Öğrenme, Bilgisayarla Görü, Biyometri, Kısıtlı Kulak Tanıma, Kısıtlamasız Kulak Tanıma

Dedicated to all the innocent, victims, and their families. So that no crime goes unpunished, and no innocent may be jailed. For a safer and more just world. So that you may recoup at least some of what you lost

ACKNOWLEDGMENT

I am deeply grateful to Allah (S.W.T.) for the abundant blessing bestowed upon me and my dear ones. No words can possibly describe the immense gratitude I feel in my heart for his divine mercy and guidance.

I express my heartfelt gratitude to my beloved parents and family for their unwavering support and guidance throughout my journey. Without their invaluable presence, I would not have been able to successfully complete my master's degree and thesis.

I am immensely grateful to my exceptional supervisor, Prof. Dr. Önsen Toygar, for her infinite patience, attention to detail and insightful ideas. Without her guidance, I would not have discovered this amazing topic.

I would like to extend my gratitude to my dear friends who have trusted my capabilities and supported me, especially my best friend Zaid Mahasneh who unwaveringly encouraged me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AMI	Mathematical Analysis of Images
AWE	Annotated Web Ears
AWED	Annotated Web Ears Database
AWEx	Annotated Web Ears Extended
BiT	Big Transfer
CNN	Convolutional Neural Network
CVLED	Color Visible Light Ear Database
EarVN	Ear Images In The Wild From Vietnam
ELU	Exponential Linear Unit
HOG	Histogram Of Oriented Gradients
KNN	K-Nearest Neighbors
L_BC	Left Ear Images With Different Levels Of Occlusion
L_BCBU	Left Ear Images With Occlusion And Background Noise
L2	Weight Decay
LAMB	Layerwise Adaptive Moments Based
LBP	Local Binary Patterns
LR-ASPP	Lite Reduced Atrous Spatial Pyramid Pooling
ML	Machine Learning
PCA	Principal Component Analysis
R_A	Right Ear Images With No Occlusion
ResNet	Residual Network
ResNetXt	Residual Network With Aggregated Transformations

RNN	Recurrent Neural Networks
ScNet-5	Self-Calibrated Convolution
SGD	Stochastic Gradient Descent
SGDM	Stochastic Gradient Descent With Momentum
SGDW	Stochastic Gradient Descent With Weight Decay
SPP	Spatial Pyramid Pooling
SVM	Support Vector Machines
USTB-Hello	University of Science And Technology Beijing
VGG	Visual Geometry Group
VGG-Face	Visual Geometry Group Face

Chapter 1

INTRODUCTION

Catching criminals is no easy task, it often requires extensive investigation into the crime and suspects followed by a court case that can take years to resolve. It can sometimes result in seizing the wrong person and jailing him/her. Therefore, pieces of evidence are especially vital, not least because it allows for the capturing of criminals and holding them accountable but also to bring justice to the victims alike, whether it is a jailed individual who was mistakenly held or the actual direct victim of the crime. Luckily there is a specific field that has been emerging ever since it was first used in the late 19th century that can provide substantial pieces of evidence to the table. The field is called Biometrics. Biometrics is the field of using the unique physical trait of an individual for the purpose of identification. When we speak about biometrics most people tend to think of the face recognition system present in their phones. However, many other biometric traits can be used for identification such as the iris, fingerprint, and ear.

1.1 Why The Ear?

While there are many unique biometric traits, the success of the most widely spread face, fingerprint and iris biometrics overshadow their disadvantages. Each biometric trait has its advantages and disadvantages. One of the disadvantages of the face biometric is the fact that the face changes with time as people age. For fingerprints, it is how susceptible to scratches. For the iris, it needs to be captured in close proximity to the camera.

One rarely familiar biometric trait is the ear. While it is unlikely anyone would like to replace it with the face recognition widely available in nowadays smartphones. It, however, provides significant advantages for the right use case. First, one of the major advantages of the ear is that it does not change much with time. Second, it doesn't require the cooperation of the subject as an image of the ear of the suspect can be taken from a distance or from an old or previously stored photo. These advantages make it especially useful for forensic investigations. It can even help solve old cases where the suspect is already dead or unreachable. Another advantage is that due to its unpopularity, it is much less likely that the criminal knows of its existence and therefore is unlikely to take measures against it.

For a trait to be a biometric trait, it needs to be unique, the same applies to the ear too, the ear needs to look unique for it to be used as a useful means of identification. The appearance of the ear is shaped by various features shown in Figure 1. There are 11 ear features that make it unique. They are the Helix, Superior Crus of Antihelix, Antihelix, concha, Antitragus, Lobe, Foseta, Inferior Curs of Antihelx, Crus of Helix, Tragus and Incisura.

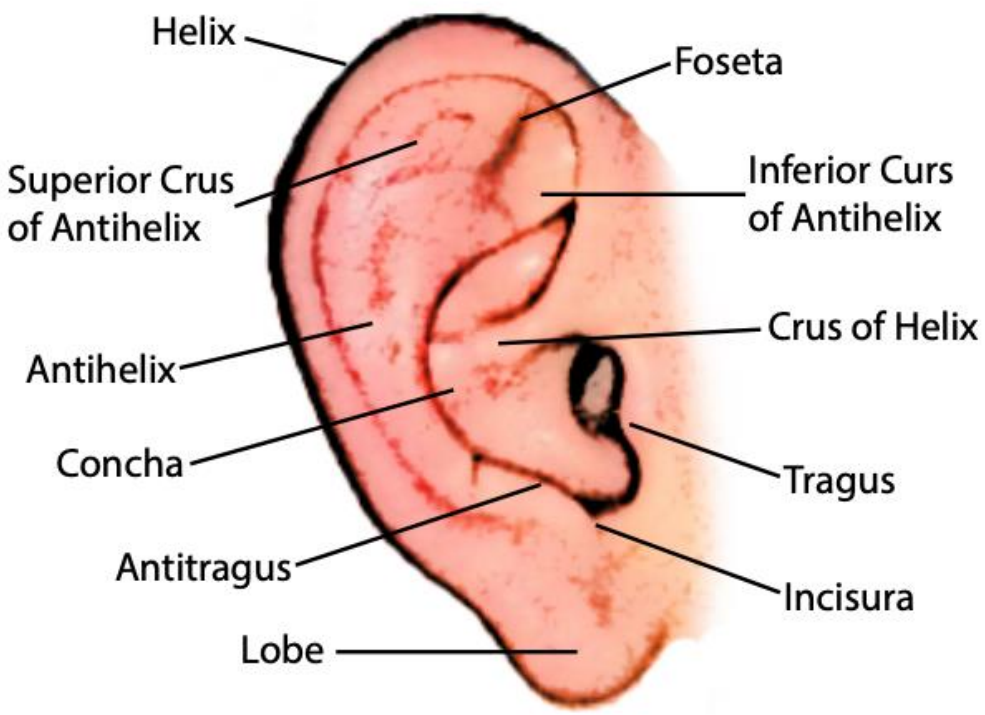

Figure 1: Ear structure [2].

1.2 Challenges and Contributions

Since an ear recognition system is most likely to be used by forensic investigators it needs to fit their needs. This means that the biometric system needs to be accurate, especially under unconstrained environments where the background, illumination, position and occlusion are relevant. However, the handcrafted approach tends to perform poorly in an unconstrained environment, therefore using deep learning is the key that is also employed in this study. Moreover, three databases were selected for training and testing the system in this thesis; they are AMI, AWE and EarVN. Two of the three databases were captured under unconstrained conditions. While all three are relatively small databases, the AMI and AWE databases are extremely small which can hurt the performance of the deep-learning model which is sensitive to the quality and quantity of the images and although the EarVN database is somewhat better on that side, it poses its own challenges, like the fact that it is unbalanced with a different

number of images for different subjects which can cause the model to be biased. This study proposes a deep learning model that accurately identifies at least 94.7% (EarVN) of individuals in the used databases using a different feature extractor with other techniques such as image augmentation, regularization, optimizer and fine-tuning and compares it with state-of-the-art methods where it outperforms the majority of them.

Some of the challenges that an ear recognition model must overcome are the illumination, occlusions, position, resolution, and image color. Examples of these challenges can be seen in Figure 2.

(a) (b) (c) (d) (e)

Figure 2: Ear recognition challenges: (a) Illumination, (b) Occlusion, (c) Position, (d) Resolution, (e) Grayscale.

Chapter 2

LITERATURE REVIEW

Generally, all ear recognition methods based on deep learning use transfer learning to extract features from an ear image. A list of some of the transfer learning models used is in Table 1, along with other related information about the approach, such as the database used and whether or not image augmentation, regularization, fine-tuning and ensemble techniques were applied. Some articles also mentioned the optimizer used, the common one being the Stochastic Gradient Descent (SGD). Table 1 also shows the accuracy achieved by every approach.

The approach proposed by [4] used a hybrid database composed of multiple image sources including the AWED and CVLED databases as well as ear images for 50 more individuals from the internet. The authors came to the conclusion that the best-performing deep learning model on the hybrid database is using the SqueezeNet feature extractor as well as image augmentation and fine-tuning, however, the authors did not use any other regularization methods. The model correctly identified 62% of the individuals in the database.

The system developed by [5] used the AWE database, however, the feature extraction is done using a hybrid approach with CNN and HOG. This approach applied image augmentation to produce a higher accuracy with regularization applied. The optimizer

used by the deep learning model is the Adam optimizer. The accuracy scored by this approach has scored 75.6%.

Zhang et al. [6] used VGG-Face and SPP for feature extraction on the USTB-Hello ear database with image augmentation, fine-tuning and ensemble learning techniques applied. This approach recorded an accuracy of 96.08% in R_A (right ear images with no occlusion), 93.89% in L_BC (left ear images with different levels of occlusion) and 86.98% in L_BCBU (left ear images with occlusion and background noise).

Emeršič et al. [7] developed a model that uses ScNet-5 for feature extraction on images in the AWEx database, the only techniques mentioned were image augmentation and ensemble learning. The model attained an accuracy of 62.8%.

Alshazly et al. [8] proposed another deep-learning model that uses ResNetXt-101 for feature extraction on the EarVN image database. The authors applied image augmentation, fine-tuning and ensemble learning with the LAMB optimizer to produce a model that accurately identifies 95.85% of individuals in the database.

The authors of [9] managed a 97.5% accuracy on an ensemble of the VGG-13-16-19 feature extractors on the AMI database. The author used the Stochastic Gradient Descent (SGD) optimizer with a combination of image augmentation, regularization, and fine-tuning.

A model that uses the AlexNet feature extractor for transfer learning on a custom database made by the authors was proposed by [11]. The model uses the Stochastic

Gradient Descent with Momentum (SGDM) optimizer with fine-tuning and ensemble learning to achieve 100% accuracy.

Dodge et al. [12] developed an ear recognition model that was trained on the AWE database. The developed system uses ResNet-18 feature extractor with the SGD optimizer. The approach applies image augmentation, fine-tuning and ensemble learning to produce a 68.50% accuracy.

Another ResNet based model was proposed by [13]. It uses the ResNet-50 feature extractor with image augmentation and ensemble learning with SGD optimizer to achieve a 99% accuracy on the AMI database.

Table 1: Summary of the literature review.

Reference	Database	Feature Extractor	Image Augmentation	Regularization	Optimizer	Fine-Tuning	Ensemble	Accuracy (%)
[4]	Hybrid (AWED + CVLED + 50 Individuals from the internet)	SqueezeNet	✔	✘	-	✔	-	SqueezeNet: 62%
[5]	AWE	CNN + HOG	✔	✔	Adam	-	-	75.6%
[6]	USTB-Hello Ear	VGG-Face + SPP	✔	✘	-	✔	✔	R_A: 96.08% L_BC: 93.89% L_BU: 91.78% L_BCBU: 86.98%
[7]	AWEx	ScNet-5	✔	-	-	-	✔	62.8%
[8]	EarVN	ResNetXt-101	✔	✘	LAMB	✔	✔	95.85%
[9]	UERC-2017	NASNET	✔	-	Adam	✔	✘	50.4%
[10]	AMI	VGG-13-16-19	✔	✔	SGD	✔	✔	97.50%
[11]	Custom	AlexNet	-	-	SGDM	✔	✘	100%
[12]	AWE	ResNet-18	✔	-	SGD	✔	✔	68.50%
[13]	AMI	ResNet-50	✔	-	SGD	-	✔	99%

The systems developed by the works of literature have shown that selecting the right feature extractor and optimizer as well as using image augmentation, regularization, fine-tuning and ensemble learning techniques increases the accuracy of the model. In some cases, it is possible to achieve high accuracy without all the mentioned techniques depending on the database. Almost all methods in Table 1 used image augmentation and the most popular optimizer is Stochastic Gradient Descent (SGD). The most frequently used feature extractors are the ones based on a residual network such as ResNetXt-101, ResNet-18 and ResNet-50. Fine-Tuning and Ensemble learning are other popular techniques that are commonly used.

Chapter 3

APPLIED METHODS

Artificial Intelligence (AI) is a large field that contains many subfields, one of the most well-known ones is machine learning (ML) and in that subfield is another important subfield known as deep learning. As is the case with all AI subfields, Deep Learning too aims to automate tasks. Deep learning uses a neural network that is composed of 3 or more layers that work together to automate tasks. One of the tasks that can be automated using a neural network is ear recognition. For a Deep Learning model to recognize ears, it must first be trained on a database or a dataset so that the values of the neurons in the neural network are updated to perform that task.

Ear recognition can also be performed using the handcrafted approaches. Unlike the deep learning approach which learns from raw input data, the handcrafted approach uses feature engineering which requires domain expertise in the field to manually engineer specific and relevant features for the ear. The handcrafted approach then uses algorithms such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), or decision trees to perform classification. Another difference between both approaches is that for a deep learning model to generalize well, it has to be trained on a large pool of labeled data. In return, the deep learning approach performs better in unconstrained environments. Some examples of deep learning strategies include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Siamese Networks.

Some examples of handcrafted strategies include Scale-Invariant Feature Transform (SIFT), Local Binary Patterns (LBP) and Principal Component Analysis (PCA).

The performance of a deep-learning model is directly dependent on its architecture, techniques applied and hyperparameters. The better they are, the more accurate and robust the model is. The following subsections describe the selected model architecture, the feature extractors chosen for transfer learning, the image augmentation techniques applied, the regularization techniques used and the fine-tuning step.

3.1 Model Architecture

The model architecture used by the proposed approach is composed of 11 layers placed in sequential order, the first of which is the Input layer that takes the image as an input with its dimensions. The next layer is the feature extractor layer responsible for transfer learning. The following layer is the fully connected layer i.e. Dense with 2048 units, followed by an ELU (Exponential Linear Unit) activation layer, a Batch Normalization layer and a dropout layer. The upcoming four layers are the same as the previous four. Finally, the sequential model ends with a final classification Dense layer. Figure 3 shows the deep learning model architecture. The code for a variant of this model can be found in the APPENDIX.

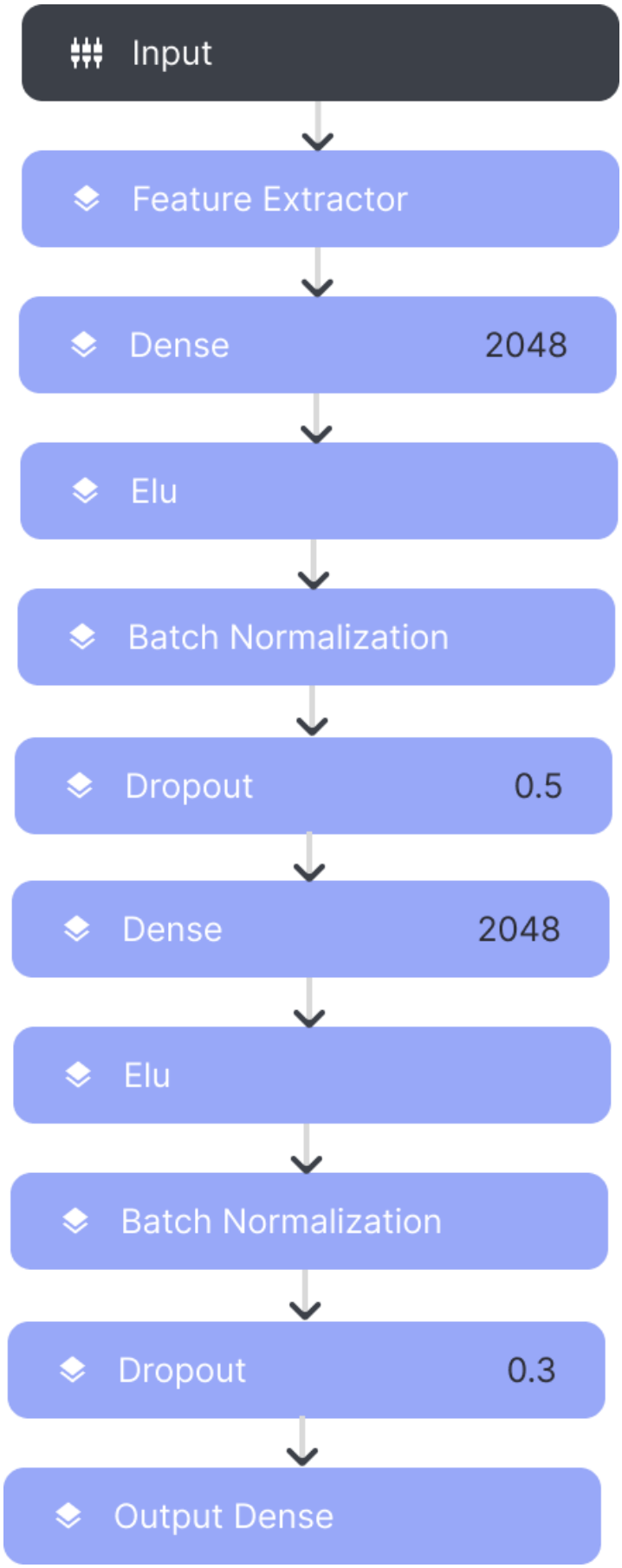

Figure 3: Deep learning model architecture.

3.2 Transfer Learning

One crucial and frequently used technique in deep learning is transfer learning. It allows for the repurpose (reuse) of a pre-trained model by transferring some of the features it learned on the dataset it trained on. Transfer learning allows the model to learn faster while requiring fewer data for training which is critical because the selected databases are small. MobileNetV3 Large, Inception-v3, EfficientNetV2-B3, Inception-ResNet-v2 and Bit S-R101x3 feature extractors are picked and compared in Chapter 4 section 2.

3.2.1 MobileNetV3 Large

The MobileNetV3 pre-trained model is a new improved version of the MobileNetV2. MobileNetV3 uses hardware-aware architecture search and novel advancements to optimize accuracy and latency trade-offs. It optimizes accuracy and latency by using hardware-aware architecture search and novel advancements. It also uses the new LR-ASPP efficient segmentation decoder. Figure 4 shows the MobileNetV3 block diagram.

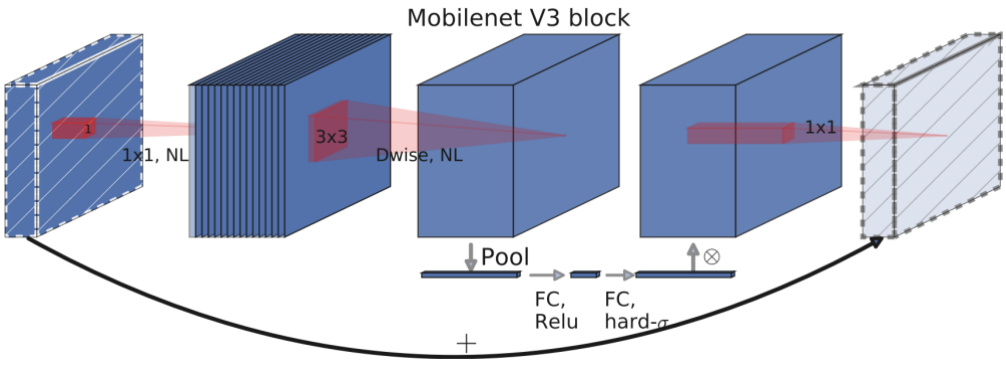

Figure 4: MobileNetV3 Large block diagram [18].

3.2.2 Inception-v3

The Inception-v3 deep learning model is the third version of the Inception architecture. The authors of [19] built the model on design principles such as increasing the width and depth of the network, balancing computational resources, and avoiding representation bottlenecks. Moreover, to reduce the parameter count and computation cost, the author factorized convolutions with large filter sizes which allowed for faster training and a larger network. Table 2 presents the network architecture of the model.

Table 2: Outline of Inception-v3 network architecture [19].

Type	**Patch size/stride** or remarks	**Input size**
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3 × Inception	As in Figure 5 in [19]	35×35×288
5 × Inception	As in Figure 5 [19]	17×17×768
2 × Inception	As in Figure 5 [19]	8×8×1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
SoftMax	Classifier	1 × 1 × 1000

3.2.3 EfficientNetV2-B3

The EfficientNetV2-B3 is a variant of the EfficientNetV2 family. It shares the same architecture as the EfficientNetV2-S but is larger. The main differences between them

are the depth, width and resolution. The authors of [20] used techniques such as training-aware neural architecture search and scaling to optimize the training speed and parameter efficiency. To increase the efficiency of the model even further the author used a method called progressive learning in which the size of the image is increased gradually during training while the regularization techniques are being adaptively adjusted. Table 3 shows the common architecture between the EfficientNetV2 family variants.

Table 3: EfficientNetV2-S deep learning model architecture [20].

Stage	**Operator**	**Stride**	**#Channels**	**#Layers**
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

3.2.4 Inception-ResNet-v2

Inception-ResNet-v2 is the Residual version of the Inception network. The network is composed of multiple components. Figure 5 shows the schema of the Inception-ResNet-v2 pre-trained model which shows the data flow from the input toward the final output SoftMax layer.

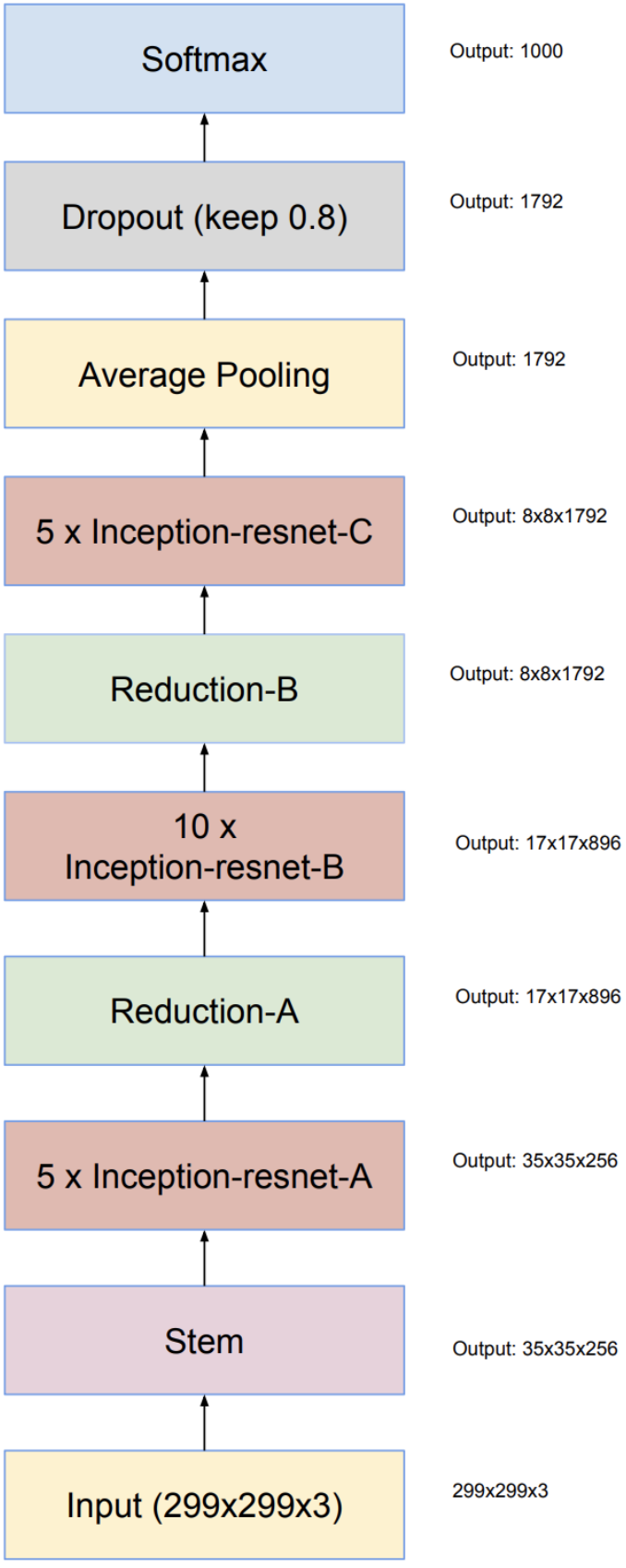

Figure 5: Inception-ResNet-v2 schema [21].

3.2.5 Bit S-R101x3

The Bit S-R101x3 is a variant of the ResNet-101 pre-trained model. The Resnet model is pre-trained on a diverse dataset and scales up the model capacity to produce the Big Transfer (BiT) Variant since larger models and datasets produce better performance. Figure 6 shows the Residual network architecture that the Bit variant is based on.

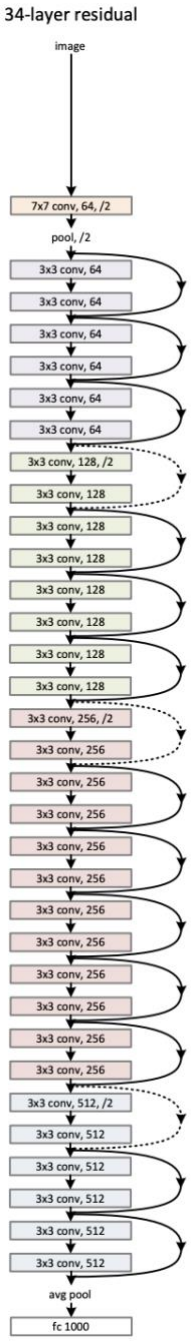

Figure 6: Residual network architecture [23].

3.3 Image Augmentation

Image augmentation is the process of producing an image by applying image operations on an input image, such as rotation, translation and zoom. Overfitting usually occurs when a deep-learning model memorizes the training dataset resulting in a high training accuracy but lower test accuracy. It can happen because the dataset is too small. Applying image augmentation to a database or dataset increases the number of images, therefore increasing test accuracy. Table 4 shows the image augmentation techniques used with their respective values applied using the Augmentor Python package. Figures 7, 8 and 9 show samples of the augmented AMI, AWE and EarVN databases.

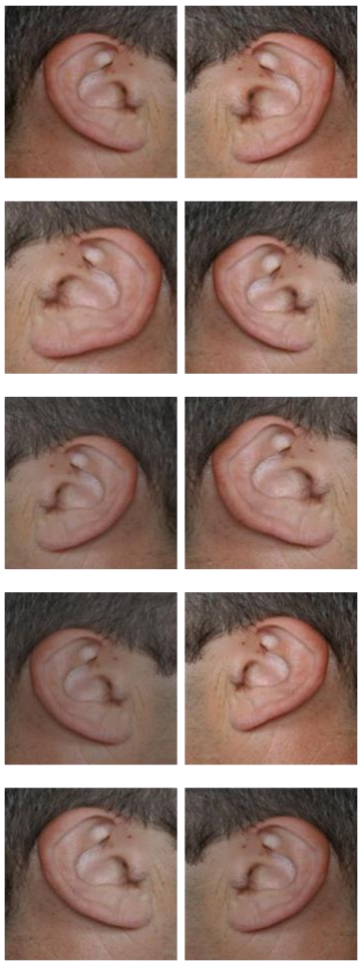

Figure 7: Augmented AMI database samples.

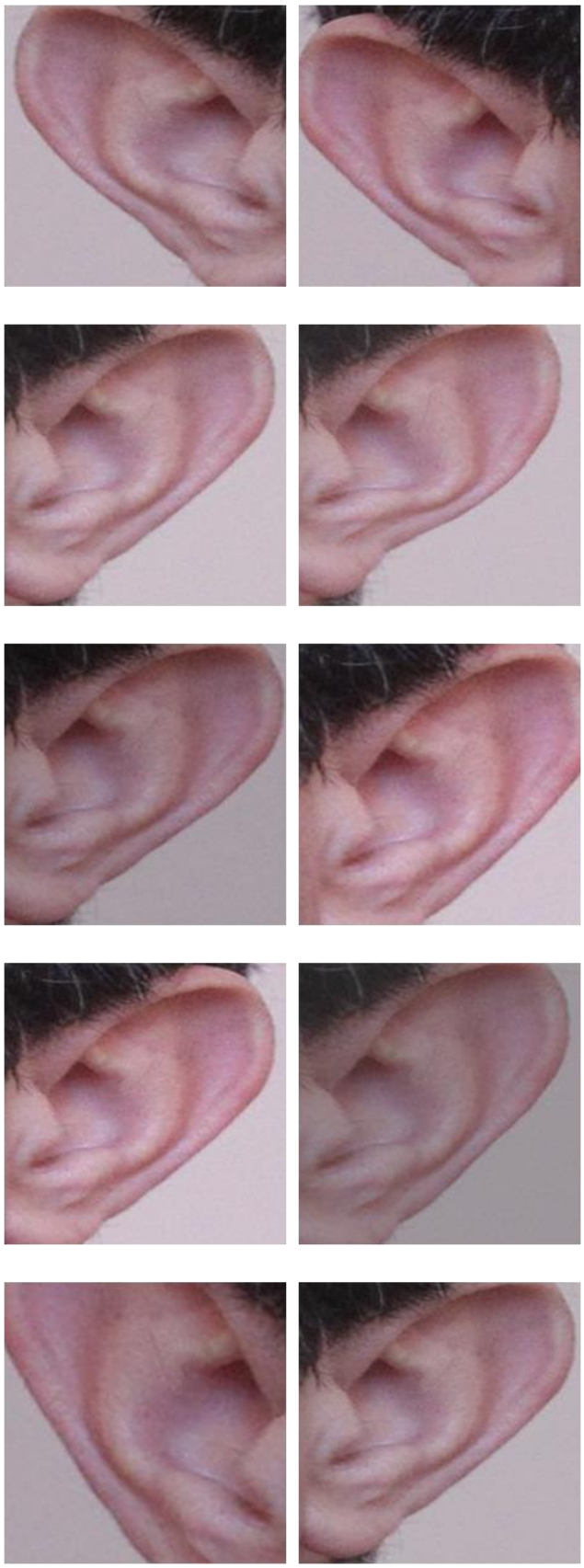

Figure 8: Augmented AWE database samples.

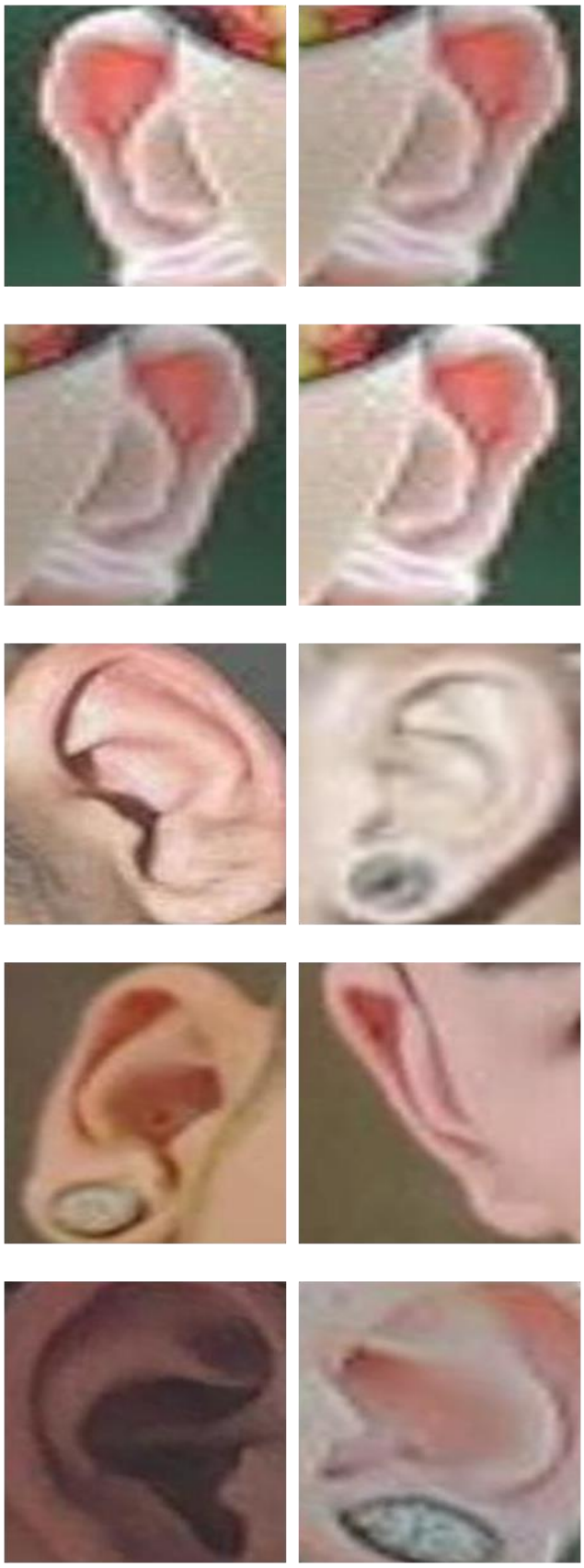

Figure 9: Augmented EarVN database samples.

Table 4: Used image augmentation techniques.

Augmentation Operation	**Probability**	**Value**
Resize	1	244x244
Random Rotation	0.5	±15º
Random Crop	1	Percentage area = 0.9
Random Distortion	0.2	Grid width and height = 4, Magnitude = 8
Random Gaussian	0.2	Grid width and height = 4, Magnitude = 8
Random Brightness	0.5	Min = 0.9, Max = 1
Random Contrast	0.5	Min = 0.9, Max = 1
Random Color	0.5	Min = 0.9, Max = 1
Flip	0.5	Horizontal

3.4 Regularization

Regularization techniques are deep learning techniques that are used to improve the generalization ability of the model by minimizing overfitting. Regularization techniques reduce the sensitivity of the model to noise or irrelevant features such as the background of an image in the training set. They encourage the model to learn simpler and more robust features by introducing additional constraints or penalties during training.

There are several commonly used regularization techniques such as Dropout, Batch Normalization, L2 (weight decay) and early stopping. The model proposed in this thesis will use all the regularization techniques mentioned except for early stopping.

3.4.1 Dropout

One of the commonly used regularization techniques is the Dropout layer. The dropout layer is a layer that randomly drops a fraction of the input neurons during each step of the training. Since the dropped-out neurons are ignored in both the forward and backward path of the training, meaning they do not contribute to the model's learning for the running epoch iteration. By dropping out a fraction of the neurons, it forces the other non-dropped neurons to learn more features resulting in a more robust neural network that is not reliant on a small and specific set of neurons, therefore, preventing complex co-adaptations [14]. The Dropout layer is usually turned off during inference or testing and the output is scaled to ensure the output of the model remains consistent. The Dropout layer has been shown to be quite effective not only at reducing overfitting but also at improving the ability of the model to generalize.

3.4.2 Batch Normalization

Batch Normalization is a layer that is used to normalize the input of a neural network layer. It aims to address the problem of change in the distribution of layer inputs during training. The normalization process aids in training the neural network by bringing the input closer to a standard Gaussian distribution.

The Batch Normalization layer has two learnable parameters, they are shift and scale. Using these parameters, the network learns an optimal shifting and scaling of the normalization values which can benefit the subsequent layers. Batch normalization offers several benefits including improvement in the convergence speed, increased stability during training, improvement in the generalization performance and reduced sensitivity to the random weights initialized [15].

3.4.3 L2

L2 or weight decay is another regularization technique that helps prevent overfitting in a neural network. It works by adding a penalty to the loss function of the model to encourage the model to have smaller weights. This has a simplification effect on the model. Moreover, it reduces the reliance on specific features or weights resulting in a model that performs better on images outside the training images. L2 has a regularization parameter λ, which determines how strong the regularization is on the layers it is placed on, the larger it is the stronger the regularization is. Using a large λ can cause the model to underfit. Underfitting is when a model performs poorly because it is not able to capture the relationship between the input and output accurately enough. However, using a low λ value may not provide enough regularization to prevent overfitting.

3.5 Fine Tuning

One of the Deep Learning techniques that are typically used in combination with transfer learning is Fine-Tuning. It is a technique that is applied to the used feature extractor to further improve the accuracy of the model. It works by leveraging the knowledge and representations from a pre-trained model and adapting them to the new task. This technique is especially useful when the target database or dataset is small and may be insufficient to train a model on it from scratch. Since feature extractors are typically trained on a large database such as ImageNet, some of the learned features can be transferred to the new target database by unfreezing layers in the feature extractor and training the model slowly (using a small learning rate) on the new dataset so that it does not lose those useful previously learned features.

Chapter 4

DATABASES AND RESULTS

In this chapter, the databases used for both training and testing the model will be explained in Section 4.1, then in Section 4.2, the experimental setup of all the models will be described and the stages of the model improvement will be explained. Next, the performance of feature extractor variants of the models will be compared in terms of accuracy and prediction runtime.

4.1 Databases

Ear image databases play a significant role in ear recognition systems, it is not only used for testing the accuracy but also more importantly as the training data for a deep-learning model. Without training data, a deep learning model can do nothing, therefore it is crucial to select the right database to train the model on as it can significantly affect the performance of the model. We need the databases to be mirroring the real-world scenarios that the system will be deployed to. Doing so can give us a more accurate expectation of the real-world accuracy of the system. Moreover, it allows the deep-learning model to work in these environments. There are two types of ear image databases, they are either constrained or unconstrained. The constrained databases are the ones that were captured under a controlled environment such as in a lab with limited variations usually in the same location but under different lighting conditions and angles of photo shooting. Unconstrained databases are the ones that were captured in an unconstrained environment. Unconstrained images usually vary significantly, some images could be taken indoors others outside, some could have occlusion, and

some may not. They could have many other variations such as resolution, scale, pose and so on. In this thesis, the proposed ear recognition system is tested on three databases, one constrained (AMI) and two unconstrained (AWE, EarVN). Their details are described below.

4.1.1 Constrained Databases

4.1.1.1 Mathematical Analysis of Images (AMI)

AMI [1] is a database of ear images taken in a controlled environment using a Nikon D100 camera. It contains ear images of 100 individuals with an age group encompassing a range between 19-65 years of age. All images were captured under the same lighting conditions, placing each subject 2 meters from the camera with a preset focus point to look at. Seven images were taken for each individual, 6 of which were of the right ear and 1 for the left. The images were taken with a slight variant in angle. Furthermore, the photos were cropped. Figure 10 shows images of a selected subject in the AMI database.

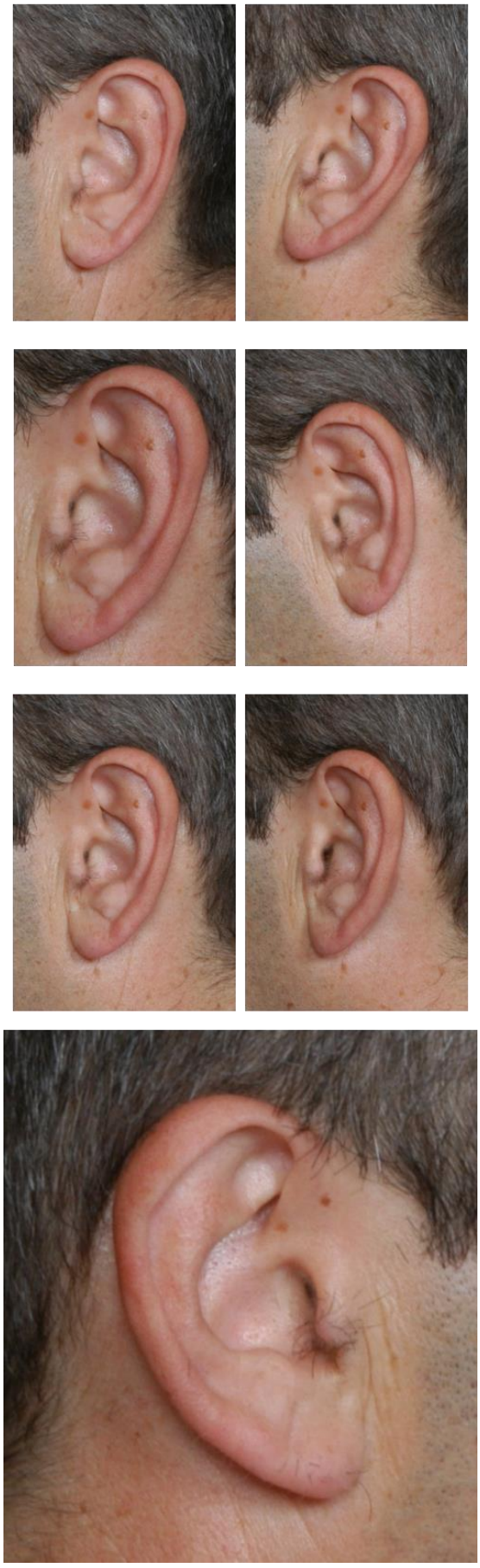

Figure 10: AMI database samples.

4.1.2 Unconstrained Databases

4.1.2.1 Annotated Web Ears (AWE)

AWE [2] is a database of ear images collected from the web. Since these images are not captured under a controlled environment, the database is in the unconstrained category. The database features ear images of 100 individuals of varying ages, ethnicities, and gender. For each individual, 10 images were saved and cropped. Collecting these images from the web ensures variability. Figure 11 shows images of a selected subject in the AWE database.

Figure 11: AWE database samples.

4.1.2.2 EarVN 1.0

EarVN [3] is another unconstrained ear image database. It features a total of 28412 images of 164 individuals. Since not all individuals have the same number of images, it makes it an unbalanced database. The database undergoes challenging variations in illumination, scale, pose, resolution, and occlusion. Figure 12 shows images of a selected subject in the EarVN database.

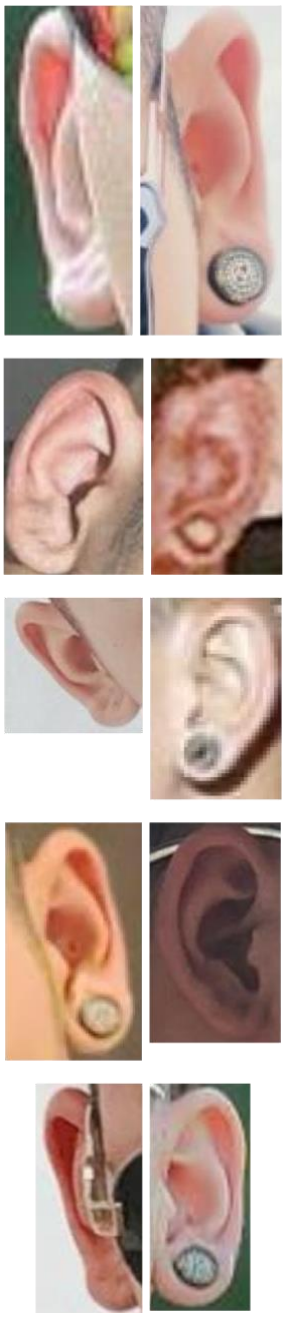

Figure 12: EarVN database samples.

4.2 Results

In any experiment, the experimental setup is crucial for finding consistent and correct results. All the experiments shown in the tables below use the same experimental setup. First, image augmentation will be applied to every database except for the first experiment in Table 5. Each database will be split into three datasets: 70% for training, 20% for validation and 10% for testing. The models will be trained on all subjects in the databases except for EarVN; only 50 out of 164 individuals will be used due to the computation complexity. The optimizer selected for the model is the SGDW with a learning rate of 0.001, momentum of 0.9 and weight decay of 0.0001. The used loss function is the Categorial Crossentropy. All dense layers used the same L2 weight of 0.0001. The rate of the dropout layers was set to 0.5 and 0.2.

Table 5 shows the stages of improvement the model went through starting with the base model i.e. the model without image augmentation, regularization techniques (Dropout, Batch Normalization and L2 weights) and fine-tuning. Table 5 also presents how the changes affect the test accuracy on the EarVN database since it is the most challenging one reaching a maximum test accuracy of 94.7%.

Table 5: Model improvement.

Method	**Test Accuracy on EarVN (%)**
Base Model	52.3%
Base Model + Image Augmentation	59.4%
Base Model + Image Augmentation + Regularization Techniques	70.8%
Base Model + Image Augmentation + Regularization Techniques + Fine Tuning	**94.7%**

In Table 6 the same model was tested on different databases with the only variation being the feature extractor selected for fine-tunning. It showed that the Bit S-R101x3 scored the highest test accuracy of 99.9% for AMI, 94.7% for EarVN and 100% for AWE. Table 6 also shows how all other models performed significantly worse on the EarVN dataset.

Table 6: Test accuracy comparison of different feature extractors.

Architecture	**Test Accuracy (%)**		
	Constrained	Unconstrained	
	AMI	**EarVN**	**AWE**
MobileNetV3 Large	98.9%	63.8%	94%
Inception-v3	97.2%	54%	91.4%
EfficientNetV2-B3	99.6%	79.1%	99.8%
Inception-ResNet-v2	91.1%	63.8%	83.6%
Bit S-R101x3	**99.9%**	**94.7%**	**100%**

Table 7 illustrates the time it takes to predict (in seconds) the identity of one individual for every database. It shows that the Bit S-R101x3 variant of the model is the most computation-intensive variant taking around 4 seconds to run, this is likely due to its large parameter size of ~398 million parameters making it the largest model in the list. The EfficientNetV2-B3 variant took the least amount of time (~0.5 seconds) to run with a parameter size of 20 million parameters.

Table 7: Prediction runtime comparison of different feature extractors.

Architecture	**Prediction Run Time (seconds)**		
	Constrained	Unconstrained	
	AMI	**EarVN**	**AWE**
MobileNetV3 Large	0.96	0.96	1.19
Inception-v3	0.61	0.6	0.74
EfficientNetV2-B3	0.5	0.48	0.5
Inception-ResNet-V2	1.49	1.5	1.48
Bit S-R101x3	**4.1**	**3.89**	**3.82**

Chapter 5

COMPARISON WITH THE STATE-OF-THE-ART METHODS

In order to understand whether the proposed deep learning model is a good performer, an objective evaluation must be carried out with other state-of-the-art methods. For the evaluation to be objective, the accuracy as well as other aspects of the model must be compared with the state-of-the-art methods on the same databases. There are several studies that developed deep learning ear recognition models, some of the models were trained and tested on constrained databases and others on unconstrained databases. Three databases were selected in this study for comparison which are the constrained AMI database and the unconstrained AWE and EarVN databases.

The best-performing proposed deep learning approach is compared with other state-of-the-art methods in Table 8 with columns including information about the reference, database, feature extractor, image augmentation, regularization, optimizer, fine-tuning, ensemble learning method and accuracy. The entries are sorted by two fields (Database and Accuracy). The result of the proposed approach is styled in bold to make it easier to find in Table 8. As can be seen, the proposed method outperforms all other methods in Table 8 in two out of the three databases used. When comparing the proposed method with the ResNetXt method [8] which has the highest accuracy on the EarVN database, we can find that the proposed approach uses a different feature

extractor. It also uses regularization techniques and a different optimizer but does not apply ensemble learning.

Priyadharshini et al. [16] proposed a deep learning model that was trained on the AMI database using a CNN-based architecture. It uses image augmentation and regularization, with the RmsProp optimizer to achieve a 96.99% accuracy.

The second model in Table 8 is an Ensemble learning model that uses the VGG-13-16-19 (ensemble of VGG-13, VGG-16, and VGG-19) [10] feature extractors on the AMI database. It uses the SGD optimizer with image augmentation, regularization and fine-tuning which results in an accuracy of 97.5%.

A model that uses the ResNet 50 feature extractor was developed by [13], the model was trained on the AMI database with the SGD optimizer. It attained an accuracy of 99% using techniques such as Image Augmentation and Ensemble learning.

Alshazly et al. [17] developed a model that achieved 99.64% on the AMI database which is the second-best approach closest to the proposed method which achieved 99.9% accuracy. The authors of [17] used the SGD optimizer with the ResNet-101-152 ensemble learning feature extractors with image augmentation, regularization and fine-tuning techniques enacted. The authors also managed a 67.25% accuracy on the AWE database but used the ResNet-34-50-101-152 feature extractors instead.

Dodge et al. [12] used the SGD optimizer to train their model on the AWE database. The authors realized an accuracy of 68.50% using the ResNet-18 feature extractor with image augmentation, fine-tuning and ensemble learning.

Another approach used the Adam optimizer to reach an accuracy of 75.6% [5]. The accuracy of that model is the second-best accuracy compared to the proposed model in this study which achieved 100% accuracy which shows more than a 24% difference in accuracy. The authors in [5] used a hybrid feature extractor that uses CNN and HOG with image Augmentation and regularization techniques to achieve the mentioned accuracy.

One of the very few studies that trained and tested their model on the EarVN database is [8]. It uses the LAMB optimizer with ResNetXt-101 in addition to image augmentation, fine-tuning and Ensemble learning to accomplish a 95.85% accuracy which is marginally better than the 94.7% accuracy obtained by this study.

Table 8: Comparison with the state-of-the-art methods.

Reference	Database	Feature Extractor	Image Augmentation	Regularization	Optimizer	Fine-Tuning	Ensemble	Accuracy (%)
[16]	AMI	CNN	✔	✔	RmsProp	✘	✘	96.99%
[10]	AMI	VGG-13-16-19	✔	✔	SGD	✔	✔	97.50%
[13]	AMI	ResNet-50	✔	-	SGD	-	✔	99%
[17]	AMI	ResNet-101-152	✔	✔	SGD	✔	✔	99.64
Proposed Method	**AMI**	**Bit S-R101x3**	✔	✔	**SGDW**	✔	✘	**99.9%**
[17]	AWE	ResNet-34-50-101-152	✔	✔	SGD	✔	✔	67.25%
[12]	AWE	ResNet-18	✔	-	SGD	✔	✔	68.50%
[5]	AWE	CNN + HOG	✔	✔	Adam	-	-	75.6%
Proposed Method	**AWE**	**Bit S-R101x3**	✔	✔	**SGDW**	✔	✘	**100%**
[8]	EarVN	ResNetXt-101	✔	✘	LAMB	✔	✔	95.85%
Proposed Method	**EarVN**	**Bit S-R101x3**	✔	✔	**SGDW**	✔	✘	**94.7%**

The majority of the models selected for comparison of the proposed model with the state-of-the-art methods has shown impressive accuracy across various databases whether they are constrained or unconstrained databases, showing the effectiveness and versatility of the deep learning approach. In the first selected database (AMI), the developed model by this study surpassed all other models in the table scoring an impressive 99.9% accuracy. When tested on the AWE database, the same model has shown outstanding results surpassing expectations and achieving an accuracy of 100%. On the EarVN database, the model developed by Alshazly et al. [8] obtained the highest accuracy at 95.85% and although the proposed model could not surpass it, it still attained a respectful close accuracy of 94.7%.

Chapter 6

CONCLUSION

In conclusion, this research has proposed a new deep-learning model for ear recognition. Due to the advantages that the ear offer such as its slowly changing nature and the fact that capturing an image of it does not require the cooperation of the subject it makes ear recognition most useful for forensic investigators, therefore, using a deep learning-based approach instead of a handcrafted approach allows for its use in unconstrained environments (wild) since the handcrafted approach for ear biometrics performs poorly in unconstrained environments which is the most likely scenario that the investigators will encounter. The proposed model was trained and tested on three databases, two of which are unconstrained (AWE, EarVN) and the last is constrained (AMI). The model has shown very promising results, consistently identifying at least 94.7% of individuals on the EarVN database and even higher on the other two. The proposed approach achieved this result by developing a model that takes advantage of various factors that affect the performance of a deep learning model. The most significant of which are the feature extractor, image augmentation, regularization, optimizer, and fine-tuning. Selecting the best-performing combination results in the highest accuracy.

The system is ready for use on a small scale, but further research is needed to make a more lightweight model that performs at least as well but is also more scalable for a

larger suspect base. Further research is needed in experimenting with different feature extractors and more optimal image augmentation operations and values.

REFERENCES

[1] Gonzalez-Sanchez, E. (2008). Biometria de la oreja. P. hd Thesis, *Universidad de Las Palmas de Gran Canaria*.

[2] Emeršič, Ž., Štruc, V., & Peer, P. (2017). Ear recognition: More than a survey. *Neurocomputing*, 255, 26-39.

[3] Hoang, V. T. (2019). EarVN1. 0: A new large-scale ear images dataset in the wild. *Data in brief*, 27.

[4] Hansley, E. E., Segundo, M. P., & Sarkar, S. (2018). Employing fusion of learned and handcrafted features for unconstrained ear recognition. *IET Biometrics*, 7(3), 215-223.

[5] Emeršič, Ž., Štepec, D., Štruc, V., & Peer, P. (2017). Training convolutional neural networks with limited training data for ear recognition in the wild. *arXiv preprint arXiv:1711.09952*.

[6] Zhang, Y., Mu, Z., Yuan, L., & Yu, C. (2018). Ear verification under uncontrolled conditions with convolutional neural networks. *IET Biometrics*, 7(3), 185-198.

[7] Emeršič, Ž., SV, A. K., Harish, B. S., Gutfeter, W., Khiarak, J. N., Pacut, A., ... & Štruc, V. (2019, June). The unconstrained ear recognition challenge 2019. In 2019 International Conference on Biometrics (ICB) (pp. 1-15). *IEEE*.

[8] Alshazly, H., Linse, C., Barth, E., & Martinetz, T. (2020). Deep convolutional neural networks for unconstrained ear recognition. *IEEE Access*, 8, 170295-170310.

[9] Radhika, K., Devika, K., Aswathi, T., Sreevidya, P., Sowmya, V., & Soman, K. P. (2020). Performance analysis of NASNet on unconstrained ear recognition. *Nature inspired computing for data science*, 57-82.

[10] Alshazly, H., Linse, C., Barth, E., & Martinetz, T. (2019). Ensembles of deep learning models and transfer learning for ear recognition. *Sensors*, 19(19), 4139.

[11] Abd Almisreb, A., Jamil, N., & Din, N. M. (2018, March). Utilizing AlexNet deep transfer learning for ear recognition. In 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP) (pp. 1-5). *IEEE*.

[12] Dodge, S., Mounsef, J., & Karam, L. (2018). Unconstrained ear recognition using deep neural networks. *IET Biometrics*, 7(3), 207-214.

[13] Sharkas, M. (2022). Ear recognition with ensemble classifiers; A deep learning approach. *Multimedia Tools and Applications*, 1-27.

[14] Baldi, P., & Sadowski, P. J. (2013). Understanding dropout. *Advances in neural information processing systems*, *26*.

[15] Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in neural information processing systems, 31*.

[16] Ahila Priyadharshini, R., Arivazhagan, S., & Arun, M. (2021). A deep learning approach for person identification using ear biometrics. *Applied intelligence*, 51, 2161-2172.

[17] Alshazly, H., Linse, C., Barth, E., Idris, S. A., & Martinetz, T. (2021). Towards explainable ear recognition systems using deep residual networks. *IEEE Access*, 9, 122254-122273.

[18] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. *In Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1314-1324).

[19] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[20] Tan, M., & Le, Q. (2021, July). Efficientnetv2: Smaller models and faster training. In *International conference on machine learning* (pp. 10096-10106). PMLR.

[21] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 31, No. 1).

[22] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big transfer (bit): General visual representation learning. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16 (pp. 491-507). *Springer International Publishing*.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

APPENDIX

Deep Learning Model Code

```
import itertools
import matplotlib.pylab as plt
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_addons as tfa
import datetime, os
import zipfile
print("TF version:", tf.__version__)
print("Hub version:", hub.__version__)
print("GPU is", "available" if tf.config.list_physical_devices('GPU') else "NOT AVAILABLE")
# Load the TensorBoard notebook extension
%load_ext tensorboard
path = "/content/EarVN.zip"
with zipfile.ZipFile(path,"r") as zip_ref:
    zip_ref.extractall()
#@title
model_name = "bit_s-r50x1" # @param ['efficientnetv2-s', 'efficientnetv2-m', 'efficientnetv2-l', 'efficientnetv2-s-21k', 'efficientnetv2-m-21k', 'efficientnetv2-l-21k', 'efficientnetv2-xl-21k', 'efficientnetv2-b0-21k', 'efficientnetv2-b1-21k', 'efficientnetv2-b2-21k', 'efficientnetv2-b3-21k', 'efficientnetv2-s-21k-ft1k', 'efficientnetv2-m-21k-ft1k', 'efficientnetv2-l-21k-ft1k', 'efficientnetv2-xl-21k-ft1k', 'efficientnetv2-b0-21k-ft1k', 'efficientnetv2-b1-21k-ft1k', 'efficientnetv2-b2-21k-
```

```
ft1k', 'efficientnetv2-b3-21k-ft1k', 'efficientnetv2-b0', 'efficientnetv2-b1',
'efficientnetv2-b2', 'efficientnetv2-b3', 'efficientnet_b0', 'efficientnet_b1',
'efficientnet_b2', 'efficientnet_b3', 'efficientnet_b4', 'efficientnet_b5', 'efficientnet_b6',
'efficientnet_b7', 'bit_s-r50x1', 'inception_v3', 'inception_resnet_v2', 'resnet_v1_50',
'resnet_v1_101', 'resnet_v1_152', 'resnet_v2_50', 'resnet_v2_101', 'resnet_v2_152',
'nasnet_large', 'nasnet_mobile', 'pnasnet_large', 'mobilenet_v2_100_224',
'mobilenet_v2_130_224', 'mobilenet_v2_140_224', 'mobilenet_v3_small_100_224',
'mobilenet_v3_small_075_224', 'mobilenet_v3_large_100_224',
'mobilenet_v3_large_075_224']
# https://tfhub.dev/google/bit/s-r50x1/1
# https://tfhub.dev/google/bit/s-r152x4/1
model_handle_map = {
  "efficientnetv2-s":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_s/feature_vector/2",
  "efficientnetv2-m":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_m/feature_vector/2"
,
  "efficientnetv2-l":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_l/feature_vector/2",
  "efficientnetv2-s-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_s/feature_vector/2"
,
  "efficientnetv2-m-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_m/feature_vector/2
",
```

```
  "efficientnetv2-l-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_l/feature_vector/2"
,
  "efficientnetv2-xl-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_xl/feature_vector/2
",
  "efficientnetv2-b0-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_b0/feature_vector/
2",
  "efficientnetv2-b1-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_b1/feature_vector/
2",
  "efficientnetv2-b2-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_b2/feature_vector/
2",
  "efficientnetv2-b3-21k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_b3/feature_vector/
2",
  "efficientnetv2-s-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_s/feature_vect
or/2",
  "efficientnetv2-m-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_m/feature_ve
ctor/2",
```

```
  "efficientnetv2-l-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_l/feature_vect
or/2",
  "efficientnetv2-xl-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_xl/feature_ve
ctor/2",
  "efficientnetv2-b0-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b0/feature_ve
ctor/2",
  "efficientnetv2-b1-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b1/feature_ve
ctor/2",
  "efficientnetv2-b2-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b2/feature_ve
ctor/2",
  "efficientnetv2-b3-21k-ft1k":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b3/feature_ve
ctor/2",
  "efficientnetv2-b0":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b0/feature_vector/2
",
  "efficientnetv2-b1":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b1/feature_vector/2
",
```

```
  "efficientnetv2-b2":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b2/feature_vector/2
",
  "efficientnetv2-b3":
"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b3/feature_vector/2
",
  "efficientnet_b0": "https://tfhub.dev/tensorflow/efficientnet/b0/feature-vector/1",
  "efficientnet_b1": "https://tfhub.dev/tensorflow/efficientnet/b1/feature-vector/1",
  "efficientnet_b2": "https://tfhub.dev/tensorflow/efficientnet/b2/feature-vector/1",
  "efficientnet_b3": "https://tfhub.dev/tensorflow/efficientnet/b3/feature-vector/1",
  "efficientnet_b4": "https://tfhub.dev/tensorflow/efficientnet/b4/feature-vector/1",
  "efficientnet_b5": "https://tfhub.dev/tensorflow/efficientnet/b5/feature-vector/1",
  "efficientnet_b6": "https://tfhub.dev/tensorflow/efficientnet/b6/feature-vector/1",
  "efficientnet_b7": "https://tfhub.dev/tensorflow/efficientnet/b7/feature-vector/1",
  "bit_s-r50x1": "https://tfhub.dev/google/bit/s-r50x1/1",
  "inception_v3": "https://tfhub.dev/google/imagenet/inception_v3/feature-vector/4",
  "inception_resnet_v2":
"https://tfhub.dev/google/imagenet/inception_resnet_v2/feature_vector/5",
  "resnet_v1_50": "https://tfhub.dev/google/imagenet/resnet_v1_50/feature-vector/4",
  "resnet_v1_101":          "https://tfhub.dev/google/imagenet/resnet_v1_101/feature-
vector/4",
  "resnet_v1_152":          "https://tfhub.dev/google/imagenet/resnet_v1_152/feature-
vector/4",
  "resnet_v2_50": "https://tfhub.dev/google/imagenet/resnet_v2_50/feature-vector/4",
```

```
  "resnet_v2_101":          "https://tfhub.dev/google/imagenet/resnet_v2_101/feature-
vector/4",
  "resnet_v2_152":          "https://tfhub.dev/google/imagenet/resnet_v2_152/feature-
vector/4",
  "nasnet_large": "https://tfhub.dev/google/imagenet/nasnet_large/feature_vector/4",
  "nasnet_mobile":
"https://tfhub.dev/google/imagenet/nasnet_mobile/feature_vector/4",
  "pnasnet_large":
"https://tfhub.dev/google/imagenet/pnasnet_large/feature_vector/4",
  "mobilenet_v2_100_224":
"https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4",
  "mobilenet_v2_130_224":
"https://tfhub.dev/google/imagenet/mobilenet_v2_130_224/feature_vector/4",
  "mobilenet_v2_140_224":
"https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4",
  "mobilenet_v3_small_100_224":
"https://tfhub.dev/google/imagenet/mobilenet_v3_small_100_224/feature_vector/5",
  "mobilenet_v3_small_075_224":
"https://tfhub.dev/google/imagenet/mobilenet_v3_small_075_224/feature_vector/5",
  "mobilenet_v3_large_100_224":
"https://tfhub.dev/google/imagenet/mobilenet_v3_large_100_224/feature_vector/5",
  "mobilenet_v3_large_075_224":
"https://tfhub.dev/google/imagenet/mobilenet_v3_large_075_224/feature_vector/5",
}
model_image_size_map = {
```

"efficientnetv2-s": 384,

"efficientnetv2-m": 480,

"efficientnetv2-l": 480,

"efficientnetv2-b0": 224,

"efficientnetv2-b1": 240,

"efficientnetv2-b2": 260,

"efficientnetv2-b3": 300,

"efficientnetv2-s-21k": 384,

"efficientnetv2-m-21k": 480,

"efficientnetv2-l-21k": 480,

"efficientnetv2-xl-21k": 512,

"efficientnetv2-b0-21k": 224,

"efficientnetv2-b1-21k": 240,

"efficientnetv2-b2-21k": 260,

"efficientnetv2-b3-21k": 300,

"efficientnetv2-s-21k-ft1k": 384,

"efficientnetv2-m-21k-ft1k": 480,

"efficientnetv2-l-21k-ft1k": 480,

"efficientnetv2-xl-21k-ft1k": 512,

"efficientnetv2-b0-21k-ft1k": 224,

"efficientnetv2-b1-21k-ft1k": 240,

"efficientnetv2-b2-21k-ft1k": 260,

"efficientnetv2-b3-21k-ft1k": 300,

"efficientnet_b0": 224,

"efficientnet_b1": 240,

```
  "efficientnet_b2": 260,
  "efficientnet_b3": 300,
  "efficientnet_b4": 380,
  "efficientnet_b5": 456,
  "efficientnet_b6": 528,
  "efficientnet_b7": 600,
  "inception_v3": 299,
  "inception_resnet_v2": 299,
  "nasnet_large": 331,
  "pnasnet_large": 331,
}
model_handle = model_handle_map.get(model_name)
pixels = model_image_size_map.get(model_name, 224)
print(f"Selected model: {model_name} : {model_handle}")
IMAGE_SIZE = (pixels, pixels)
print(f"Input size {IMAGE_SIZE}")
BATCH_SIZE = 32#@param {type:"integer"}
data_dir = "/content/EarVN"
train_dir = "/content/EarVN/train"
val_dir = "/content/EarVN/val"
test_dir = "/content/EarVN/test"
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    interpolation="bicubic",
```

```
    # Seed needs to provided when using validation_split and shuffle = True.
    # A fixed seed is used so that the validation set is stable across runs.
    seed=123,
    image_size=IMAGE_SIZE,
    batch_size=1)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    val_dir,
    label_mode="categorical",
    interpolation="bicubic",
    # Seed needs to provided when using validation_split and shuffle = True.
    # A fixed seed is used so that the validation set is stable across runs.
    seed=123,
    image_size=IMAGE_SIZE,
    batch_size=1)
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    label_mode="categorical",
    interpolation="bicubic",
    # Seed needs to provided when using validation_split and shuffle = True.
    # A fixed seed is used so that the validation set is stable across runs.
    seed=123,
    image_size=IMAGE_SIZE,
    batch_size=1)
class_names = tuple(train_ds.class_names)
train_size = train_ds.cardinality().numpy()
```

```
train_ds = train_ds.unbatch().batch(BATCH_SIZE)
train_ds = train_ds.repeat()
normalization_layer = tf.keras.layers.Rescaling(1. / 255)
preprocessing_model = tf.keras.Sequential([normalization_layer])
do_data_augmentation = False #@param {type:"boolean"}
if do_data_augmentation:
  preprocessing_model.add(
      tf.keras.layers.RandomRotation(20))
  preprocessing_model.add(
      tf.keras.layers.RandomTranslation(0, 0.1))
  preprocessing_model.add(
      tf.keras.layers.RandomTranslation(0.1, 0))
  # preprocessing_model.add(
  #     tf.keras.layers.RandomContrast(0.2, seed=123)
  # )
  # Like the old tf.keras.preprocessing.image.ImageDataGenerator(),
  # image sizes are fixed when reading, and then a random zoom is applied.
  # If all training inputs are larger than image_size, one could also use
  # RandomCrop with a batch size of 1 and rebatch later.
  preprocessing_model.add(
      tf.keras.layers.RandomZoom(0.2, 0.2))
  preprocessing_model.add(
      tf.keras.layers.RandomFlip(mode="horizontal"))
train_ds = train_ds.map(lambda images, labels:
                        (preprocessing_model(images), labels))
```

```
# val_ds = build_dataset("validation")
valid_size = val_ds.cardinality().numpy()
val_ds = val_ds.unbatch().batch(BATCH_SIZE)
val_ds = val_ds.map(lambda images, labels:
                    (normalization_layer(images), labels))
do_fine_tuning = True #@param {type:"boolean"}
# model_handle = "https://tfhub.dev/google/bit/s-r101x3/1"
print("Building model with", model_handle)
model = tf.keras.Sequential([
    # Explicitly define the input shape so the model can be properly
    # loaded by the TFLiteConverter
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)),
    hub.KerasLayer(model_handle, trainable=do_fine_tuning),
    tf.keras.layers.Dense(2048,                                    activation="elu",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(2048,                                    activation="elu",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(rate=0.3),
    tf.keras.layers.Dense(len(class_names),
                          kernel_regularizer=tf.keras.regularizers.l2(0.0001))
])
model.build((None,)+IMAGE_SIZE+(3,))
```

```
model.summary()
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
# Define the EarlyStopping callback
early_stopping_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath='/content/bit-s-r101x3_adam_earvn.h5',
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
# tfa.optimizers.SGDW(learning_rate=0.001, momentum=0.9, weight_decay=0.0001)
model.compile(
  optimizer='adam',
  loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
steps_per_epoch = train_size // BATCH_SIZE
validation_steps = valid_size // BATCH_SIZE
hist = model.fit(
    train_ds,
    epochs=30, steps_per_epoch=steps_per_epoch,
    validation_data=val_ds,
```

```
validation_steps=validation_steps,
callbacks=[tensorboard_callback, early_stopping_callback, checkpoint_callback]
).history
```