Face Transformer - Rethinking Model Incorporating EfficientNet Into ViT

A Project Report
Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Technology
in
Computer Science and Engineering

by

Debargha Mitra Roy **20101104010**
Bikash Shaw **20101104059**
Suprio Kundu **20101104062**

Supervisor:
Prof. Srinibas Rana

Department of Computer Science and Engineering
Jalpaiguri Government Engineering College
Jalpaiguri - 735102 (India)

12 June, 2024

Certificate

Department of Computer Science and Engineering
Jalpaiguri Government Engineering College, Jalpaiguri

It is certified that the work contained in the project report entitled "Face Transformer - Rethinking Model Incorporating EfficientNet Into ViT" by the following students has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Debargha Mitra Roy	20101104010
Bikash Shaw	20101104059
Suprio Kundu	20101104062

Date: Prof. Srinibas Rana

This project report entitled "Face Transformer - Rethinking Model Incorporating EfficientNet Into ViT" submitted by the group is approved for the degree of Bachelor of Technology.
The viva-voce examination has been held on _____.

Supervisor(s) Examiner(s)

Declaration

Jalpaiguri
12 June, 2024

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We declare that We have properly and accurately acknowledged all sources used in the production of this report. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Debargha Mitra Roy	20101104010	_____
Bikash Shaw	20101104059	_____
Suprio Kundu	20101104062	_____

Acknowledgements

Jalpaiguri
12 June, 2024

The project work summarised in this report, explores the project named: **"Face Transformer - Rethinking Model Incorporating EfficientNet Into ViT"**.

This is a combined endeavor of a number of people who directly or indirectly helped us in completing our project synopsis. A word of thanks also goes to all our friends for being our best critics. And finally, this documentation would never have been more educative and efficient without the constant help and guidance of our project guide Prof. Srinibas Rana. We would like to thank her for giving us the right guidance and encouraging us to complete the project report on time. We also express our deepest and sincere gratitude to all our teachers for their kind comments and advice. We would also like to express our heartiest gratitude to the HOD, Dr. Subhas Barman, and other faculties for their encouragement and kind suggestions

Debargha Mitra Roy	20101104010
Bikash Shaw	20101104059
Suprio Kundu	20101104062

Abstract

Towards a robust and EfficientNet Architecture for Face Recognition. This work introduces a groundbreaking approach that reimagines face transformers through their integration with EfficientNets. Recently there has been a growing interest in Transformer not only in NLP but also in computer vision. Therefore, we investigate the performance of Transformer [1] models in face recognition. Considering the original Transformer may neglect the inter-patch information, we modify the patch generation process and make the tokens with sliding patches which overlaps with each others. The models are trained on CASIA-WebFace [2] databases, and evaluated on several mainstream bench-marks, including LFW[3] database. We demonstrate that Face Transformer models trained on a large-scale database, Casia-WebFace [2], achieve comparable performance as CNN with similar number of parameters and MACs.

We propose a hybrid architecture synergistically combining the strengths of both architectures, aiming for robust and efficient face recognition. Face recognition has achieved remarkable progress in recent years, but challenges remain in terms of robustness, efficiency, and scalability. Transformers have emerged as powerful models for various vision tasks, but their direct application to face recognition faces challenges due to computational cost and potential overfitting. EfficientNets, on the other hand, offer a balance of accuracy and efficiency in convolutional neural networks.

In this work, we propose a novel approach that rethinks face transformers by integrating EfficientNets with ViT. We explore a hybrid architecture that leverages the strengths of both transformers and EfficientNets, aiming to achieve robust and efficient face recognition. We employ EfficientNets as a backbone for feature extraction, extracting informative and compact features while maintaining computational efficiency.

Our findings demonstrate that the proposed hybrid architecture significantly surpasses existing methods in face recognition performance while maintaining excellent computational efficiency. It paves the way for developing robust, efficient, and scalable face recognition systems with diverse applications, ranging from security and access control to personalized user experiences and social media.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Face recognition has experienced a momentous surge in recent years, finding applications in realms ranging from security and surveillance to access control and personalized user experiences. Despite remarkable achievements, challenges remain in attaining the trifecta of robustness, efficiency, and scalability crucial for real-world deployments.

Recently there has been great interests of Transformer not only in NLP but also in Computer Vision (CV). We wonder if transformer can be used in face recognition by incorporating EfficientNet into ViT and whether it is better than CNNs. Therefore, we investigate the performance of Transformer models in face recognition. The models are trained on a large scale face recognition database Casia-Webface and evaluated on several mainstream benchmarks, including LFW [3], SLLFW [6, 7], CALFW [8], CPLFW [9], TALFW [10], CFP-FP [11] & AGEDB [12] databases. We demonstrate that Transformer models achieve comparable performance as CNN with similar number of parameters and MACs. The Face-Transformer mainly uses ViT (Vision Transformer) architecture [13]. Now we demonstrate if we can transfer learn and fine-tune the model with EfficientNet & merge it into ViT to get a better results.

Transformers, with their self-attention mechanisms and global information capture, have emerged as formidable players in computer vision. Their success in tasks like image classification and object detection has naturally spurred exploration in face recognition. However, directly applying transformers to facial images encounters obstacles in the form of high computational cost and potential overfitting, particularly with limited datasets. These concerns highlight the need for a strategic rethinking of face transformers to unlock their full potential in this domain.

On the other hand, EfficientNets have earned widespread acclaim for their ability to strike a delicate balance between accuracy and efficiency within the realm of convolutional neural networks (CNNs). Their unique scaling methods allow them to achieve

state-of-the-art performance while maintaining computational budgets, making them attractive candidates for resource-constrained environments.

In this work, we bridge the gap between these two powerful architectures by proposing a novel hybrid approach that integrates EfficientNets with redesigned face transformers. We leverage the strengths of both to achieve exceptional recognition performance with improved robustness and efficiency. This marks a significant step forward in pushing the boundaries of face recognition towards practical and scalable solutions.

The core of our approach lies in three key principles:

- **EfficientNet-based Feature Extraction:** We harness the superior feature extraction capabilities of EfficientNets to capture informative and discriminative representations of facial data. This lays the foundation for accurate recognition while keeping computational needs in check.

- **Transformer Redesign for Face Recognition:** We carefully restructure the transformer architecture, tailoring it specifically for facial recognition tasks. This involves optimizing attention mechanisms, refining positional encoding strategies, and implementing specialized normalization techniques.

- **Seamless Architecture Integration:** We seamlessly integrate the EfficientNet backbone and the redesigned transformer, facilitating effective feature fusion and ensuring smooth information flow throughout the network.

Through extensive experiments on challenging benchmarks, we demonstrate the efficacy of our proposed approach. Our hybrid architecture significantly surpasses existing methods in terms of accuracy, robustness to pose, illumination, and occlusion variations, and computational efficiency. Moreover, insightful ablation studies reveal the contributions of each component and design choice, providing valuable understanding of the factors driving our success.

Chapter 2

Literature Survey on Face Transformer, Vision Transformer, Swin Transformer, Neighborhood Attention Transformer & EfficientNet

2.1 Face Transformer

A face transformer is a type of deep learning model that is used for face recognition and other face-related tasks. It is based on the transformer architecture, which was originally developed for natural language processing tasks. Transformer models are known for their ability to learn long-range dependencies, which makes them well-suited for tasks such as face recognition, where the identity of a person can be determined by their facial features, even if the features are partially obscured or distorted.[14]

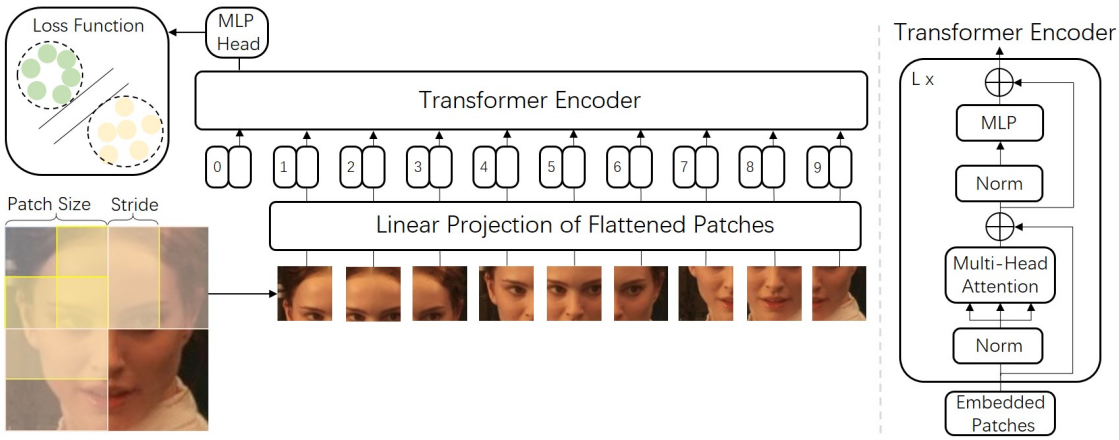

Figure 2.1: Architecture of Face Transformer

Recently there has been a growing interest in Transformer not only in NLP but also in computer vision. We wonder if transformers can be used in face recognition and whether it is better than CNNs. Therefore, we investigate the performance of Transformer models in face recognition. Considering the original Transformer may neglect the interpatch information, we modify the patch generation process and make the tokens with sliding patches which overlap with each other. The models are trained on CASIA-WebFace [2] and evaluated on several mainstream benchmarks, including LFW [3], SLLFW [6, 7], CALFW [8], CPLFW [9], TALFW [10], CFP-FP [11] & AGEDB [12] and IJB-C [15] databases. We demonstrate that Face Transformer models trained on a large-scale database, MS-Celeb-1M [16], achieve comparable performance as CNN with similar number of parameters and MACs.Despite the success of CNNs, we still wonder can Transformer be used in face recognition and whether it is better than ResNet-like CNNs.

Since Transformer has shown its excellent performance combined with large scale databases [14], while there have been lots of large scale training database in face recognition. It is interesting to observe the performance of combination of Transformer and large scale face training databases. Perhaps Transformer is just the best to challenge the CNNs hegemony over the face recognition task. It is known that, the efficiency bottleneck of Transformer models,is just the key of them, i.e., the self-attention mechanism,which incurs a complexity of $O(n^2)$ with respect to sequence length. Of course efficiency is important for face recognition models, but in this paper, let us mainly determine the feasibility of applying Transformer models in face recognition and leave out the potential efficiency problem of them.

2.1.1 Network Architecture

Face Transformer model follows the architecture of ViT [14], which applies the original Transformer. The only difference is that, we modify the tokens generation method of ViT, to generate tokens with sliding patches, i.e., to make the image patch overlaps, for the better description of the inter-patch information, as shown in Figure-2.1. Specifically, we extract sliding patches from the image $X \in \mathbb{R}^{W\times W\times C}$ with patch size P and stride S for them (with implicit zero on both sides of input), and finally obtain a sequence of flattened 2D patches $X_p \in \mathbb{R}^{N\times(P^2\times C)}$. (W, W) is the resolution of the original image while (P, P) is the resolution of each image patch. The effective sequence length is the number of patches $N = \left\lfloor \frac{(W+2p-(P-1))}{S} + 1 \right\rfloor$, where p is the amount of zero-paddings.

As ViT did, a trainable linear projection maps the flattened patches X_p to the model dimension D, and outputs the patch embeddings X_pE. The class token, i.e., a learnable

embedding ($X_{\text{class}} = z_0^0$) is concatenated to the patch embeddings, and its state at the output of the Transformer encoder (z_0^L) is the final face image embedding, as Equation 2. Then, position embeddings are added to the patch embeddings to retain positional information. The final embedding

$$z_0 = X_{\text{class}}; X_p^1 E; X_p^2 E; \ldots; X_p^N E + E_{\text{pos}} \quad (1)$$

serves as input to the Transformer,

$$z_l^{'} = \text{MSA}(\text{LN}(z_{l-1}^{'})) + z_{l-1}^{'}, \quad l = 1, \ldots, L,$$

$$z_l = \text{MLPLN}(z_l^{'})) + z_l^{'}, \quad l = 1, \ldots, L,$$

$$x = \text{LN}(z_0^L).$$

which consists of multiheaded self-attention (MSA) and MLP blocks, with LayerNorm (LN) before each block and residual connections after each block, as shown in Figure 1. In Equation-2, the output x is the final output of Transformer model. One of the key block of Transformer, MSA, is composed of k parallel self-attention (SA),

$$[q, k, v] = zU_{qkv}, \quad (2.1)$$

$$\text{SA}(z) = \text{softmax}\left(\frac{qk^T}{\sqrt{D_h}}\right)v, \quad (2.2)$$

where $z \in \mathbb{R}^{(N+1)\times D}$ is an input sequence,$U_{qkv} \in \mathbb{R}^{D\times 3D_h}$ is the weight matrix for linear transformation, and

$A = softmax\left(q\frac{k^T}{\sqrt{D_h}}\right)$ is the attention map.

The output of MSA is the concatenation of k attention head outputs $\text{MSA}(z) = [SA_1(z); SA_2(z); \ldots; SA_k(z)]U_{\text{msa}}$

where $U_{\text{msa}} \in \mathbb{R}^{kD_h\times(D+1)}$.

2.1.2 Loss Function

The output x of Equation 2, i.e., the final output of Transformer model, is supervised by an elaborate loss function [8], [9], [10] for better discriminative ability,

$$L = -\log P(y) = -\log \frac{e^{W_y^T x + b_y}}{\sum_{j=1}^{C} e^{W_j^T x + b_j}}, \quad (2.3)$$

where y is the label, $P - y$ is the predicted probability of assigning x to class y, C is the number of identities, $W - j$ is the j-th column of the weight of the last fully connected layer, and $b_j \in \mathbb{R}^C$ is the bias. Softmax based loss functions,remove the bias term and

transform $W_j^T x = s\cos\theta_j$,and incorporate large margin in the $\cos\theta_{y_i}$ term. Therefore, softmax based loss functions can be formulated as

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log P_{y_i} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{sf(\theta_{y_i})}}{e^{sf(\theta_{y_i})} + \sum_{j=1,j\neq y_i}^{C} e^{s\cos\theta_j}}, \tag{2.4}$$

where $f(\theta_{y_i}) = \cos\theta_{y_i} - m$ in CosFace.

2.1.3 Objective

- To learn a representation of face images that is invariant to variations in lighting, pose, and expression.
- To achieve state-of-the-art results on face recognition benchmarks.
- To be robust to variations in the quality of the input images. To be efficient in terms of computation and memory.

2.1.4 Problem Formation

- **Data augmentation:** Data augmentation can be used to increase the amount of data available for training face transformer models. This can be done by artificially creating new face images by applying transformations such as cropping, flipping, and rotating.
- **Robustness to variations in lighting, pose, and expression:** Face transformer models can be made more robust to variations in lighting, pose, and expression by using techniques such as adversarial training and data augmentation.
- **Computational efficiency:** Face transformer models can be made more computationally efficient by using techniques such as pruning and quantization.

The Face Transformer model is sourced from the Face Evolve - `https://github.com/ZhaoJ9014/face.evoLVe` Project [17].

2.2 Vision Transformer (ViT)

The Vision Transformer (ViT) [18] model architecture was introduced in a research paper published as a conference paper at ICLR 2021 titled **"An Image is Worth** 16×16 **Words: Transformers for Image Recognition at Scale"** [5]. It was developed and published by Neil Houlsby, Alexey Dosovitskiy, and 10 more authors of the Google Research Brain

Team. The fine-tuning code and pre-trained ViT models are available on the GitHub of the Google Research team. You find them here. The ViT models were pre-trained on the ImageNet [19] and ImageNet-21k [20] datasets.

Date	Model	Description	Vision Transformer?
2017 Jun	Transformer	A model based solely on an attention mechanism. It demonstrated excellent performance on NLP tasks.	No
2018 Oct	BERT	Pre-trained transformer models started dominating the NLP field	No
2020 May	DETR	DETR is a simple yet effective framework for high-level vision that views object detection as a direct set prediction problem.	Yes
2020 May	GPT3	The GPT-3 is a huge transformer model with 170B parameters that takes a significant step towards a general NLP model.	No
2020 Jul	iGPT	The transformer model, originally developed for NLP, can also be used for image pre-training.	Yes
2020 Oct	ViT	Pure transformer architectures that are effective for visual recognition	Yes
2020 Dec	IPT/SETR/CLIP	Transformers have been applied to low-level vision, segmentation, and multimodality tasks, respectively	Yes
2021 today	ViT Variants	Several ViT variants include DeiT, PVT, TNT, Swin, and CSWin (2022).	Yes

Table 2.1: Origin and history of vision transformer models

2.2.1 Difference between CNN and ViT

Vision Transformer (ViT) achieves remarkable results compared to convolutional neural networks (CNN) while obtaining substantially fewer computational resources for

pre-training. In comparison to convolutional neural networks (CNN), Vision Transformer (ViT) shows a generally weaker inductive bias resulting in increased reliance on model regularization or data augmentation (AugReg) when training on smaller datasets.

The ViT is a visual model based on the architecture of a transformer originally designed for text-based tasks. The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image. ViT exhibits an extraordinary performance when trained on enough data, breaking the performance of a similar state-of-the-art CNN with 4× fewer computational resources.

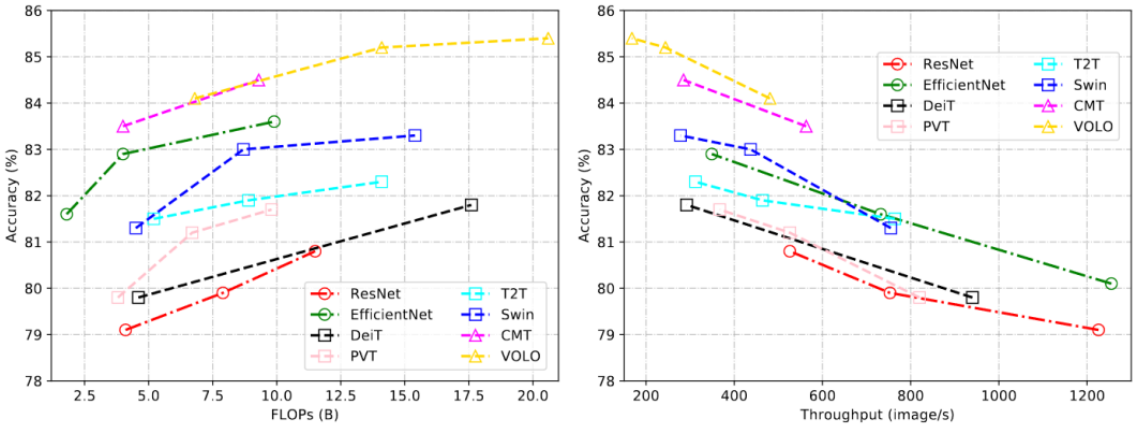

Figure 2.2: CNN vs ViT

2.2.2 ViT Architecture

The overall architecture can be described easily in five simple steps below: [21]

- Split an input image into patches.
- Get linear embeddings (representation) from each patch referred to as Patch Embeddings.
- Add position embeddings and a `[cls]` token to each of the Patch Embeddings.
- Pass through a Transformer Encoder and get the output values for each of the [cls] tokens.
- Pass the representations of `[cls]` tokens through a MLP Head to get final class predictions.

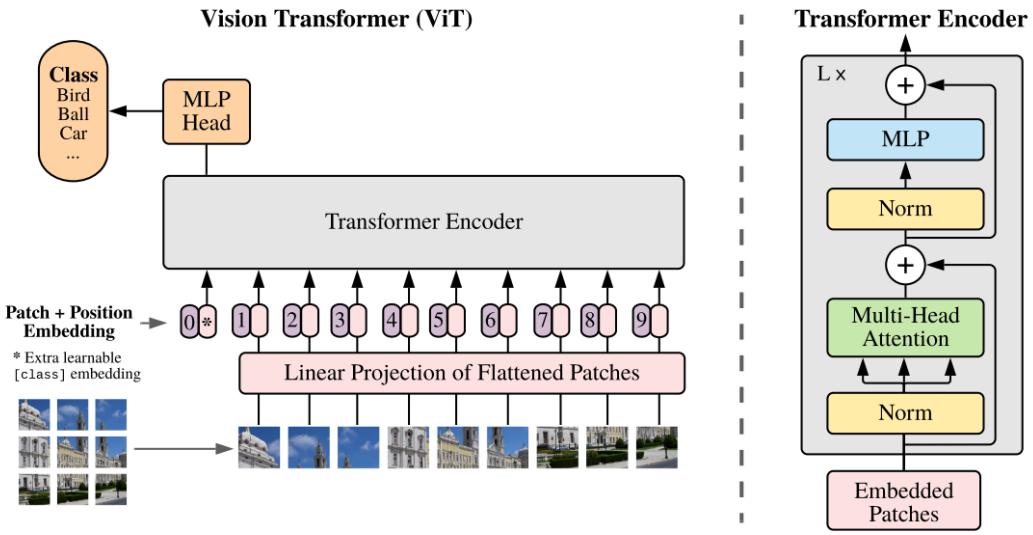

Figure 2.3: ViT Architecture

Let's now look at the five steps again with the help of fig-2.3. Let's imagine that we want to classify a 3 channel (RGB) input image of a frog of size 224×224.

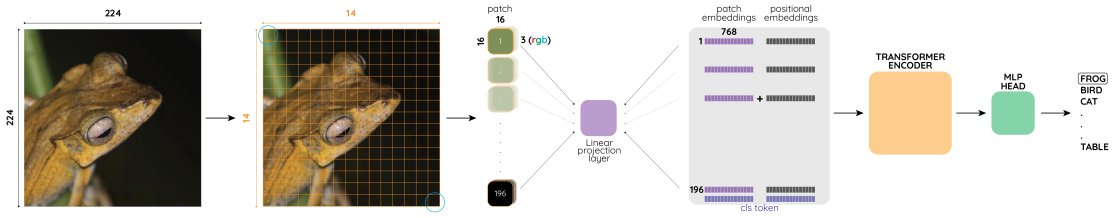

Figure 2.4: ViT Explain

The **first step** is to create patches all over the image of patch size 16×16. Thus we create 14×14 or 196 such patches. We can have these patches in a straight line as in fig-2.4 where the first patch comes from the top-left of the input image and the last patch comes from the bottom-right. As can be seen from the figure, the patch size is $3 \times 16 \times 16$ where 3 represents the number of channels (RGB).

In the **second step**, we pass these patches through a **linear projection layer** to get 1×768 long vector representation for each of the image patches and these representations have been shown in purple in the figure. In the paper, the authors refer to these representations of the patches as **Patch Embeddings**. Can you guess what's the size of this patch embedding matrix? It's 196×768. Because we had a total of 196 patches and each

patch has been represented as a $1x768$ long vector. Therefore, the total size of the patch embedding matrix is 196×768.

In the **third step**, we take this patch embedding matrix of size 196×768 and similar to BERT [22], the authors prepend a `[cls]` token to this sequence of embedded patches and then add **Position Embeddings**. As can be seen from fig-2, the size of the **Patch Embeddings** becomes 197×768 after adding the `[cls]` token and also the size of the **Position Embeddings** is 197×768.

In the **fourth step**, we pass these preprocessed patch embeddings with positional information and prepended `[cls]` token to the **Transformer Encoder** and get the learned representations of the [cls] token. Thus, the output frpm the Transformer Encoder would be of size 1×768 which is then fed to the MLP Head (which is nothing but a Linear Layer) as part of the final **fifth step** to get class predictions.

2.2.3 Patch Embeddings

In this section we will be looking at steps one and two in detail. That is the process of getting patch embeddings from an input image.

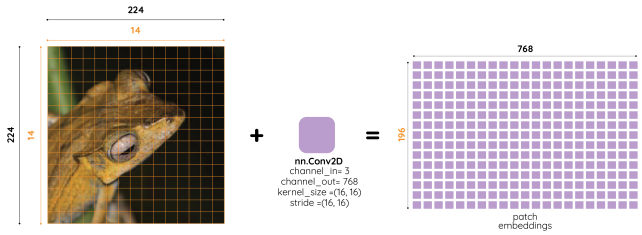

Figure 2.5: Patch Embeddings

We get patch embeddings from an input image is to first split an image into fixed-size patches and then linearly embed each one of them using a linear projection layer as shown in fig-2.5.

But, it is actually possible to combine both steps into a single step using **2D Convolution** operation. It is also better from an implementation perspective to do it this way as our GPUs are optimized to perform the convolution operation and it takes away the need to first split an image into patches.

If we set the the number of `out_channels` to 768, and both `kernel_size` & `stride` to 16, then as shown in fig-2.5, once we perform the convolution operation (where the 2-D Convolution has kernel size $3 \times 16 \times 16$), we can get the **Patch Embeddings** matrix of size 196×768.

2.2.4 [cls] token & Position Embeddings

In this step, we prepend `[cls]` tokens and add **Positional Embeddings** to the **Patch Embeddings**.

From the paper: > Similar to BERT's[22] `[class]` token, we prepend a learnable embedding to the sequence of embedded patches, whose state at the output of the Transformer encoder (referred to as Z_L^0) serves as the image representation. Both during pre-training and fine-tuning, a classification head is attached to Z_L^0.

Position embeddings are also added to the patch embeddings to retain positional information. We use standard learnable 1D position embeddings and the resulting sequence of embedding vectors serves as input to the encoder.

This process can be easily visualized as below:

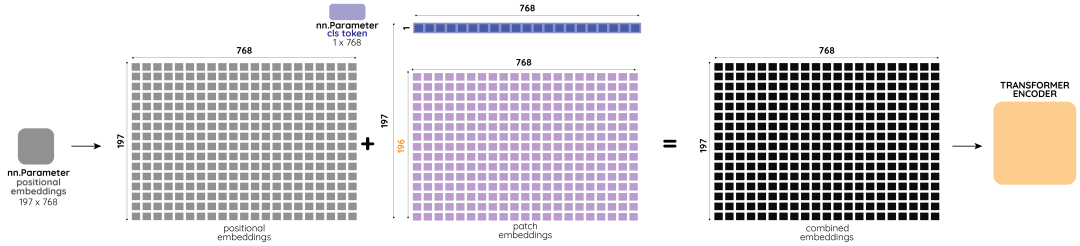

Figure 2.6: cls Tokens

As can be seen from fig-2.6, the `[cls]` token is a vector of size 1×768. We **prepend** it to the **Patch Embeddings**, thus, the updated size of **Patch Embeddings** becomes 197×768.

Next, we add **Positional Embeddings** of size 197×768 to the **Patch Embeddings** with `[cls]` token to get combined embeddings which are then fed to the **Transformer Encoder**. This is a pretty standard step that comes from the original Transformer paper - Attention is all you need [1].

2.2.5 The Transformer Encoder

A single layer/block of the **Transformer Encoder** can be visualized as below:

As shown in fig-2.3, the Transformer Encoder consists of alternating layers of **Multi-Head Attention** and **MLP** blocks. Also, as shown in fig-2.3, **Layer Norm** is used before every block and residual connections after every block.

The first layer of the **Transformer Encoder** accepts **combined embeddings** of shape 197×768 as input. For all subsequent layers, the inputs are the outputs `Out` matrix of shape 197×768 from the previous layer of the **Transformer Encoder**. There are a total of 12 such layers in the Transformer Encoder of the ViT-Base architecture.

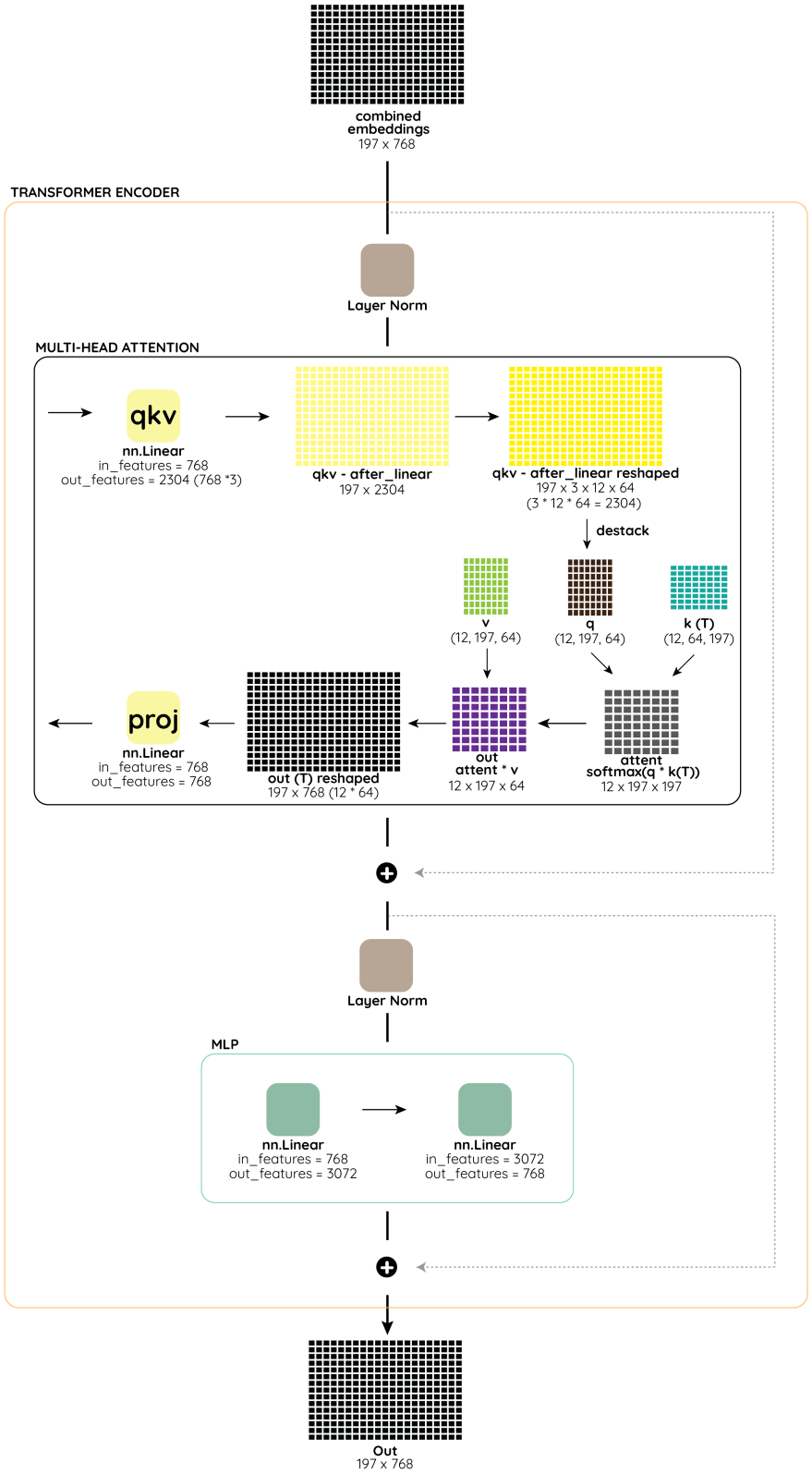

Figure 2.7: Transformer Encoder

Inside the layer, the inputs are first passed through a **Layer Norm**, and then fed to **Multi-Head Attention** block.

Inside the **Multi-Head Attention**, the inputs are first converted to $197 \times 2304 (768 \times 3)$ shape using a **Linear layer** to get the **qkv** matrix. Next we reshape this **qkv** matrix into $197 \times 3 \times 768$ where each of the three matrices of shape 197×768 represent the **q**, **k** and **v** matrices. These **q**, **k** and **v** matrices are further reshaped to $12 \times 197 \times 64$ to represent the 12 attention heads. Once we have the **q**, **k** and **v** matrices, we finally perform the attention operation inside the **Multi-Head Attention** block which is given by the equation:

$Attention(qkv) = softmax\left(\frac{qk^T}{\sqrt{d_k}}\right) \times v$ [eq-1]

Once we get the outputs from the **Multi-Head Attention block**, these are added to the inputs (skip connection) to get the final outouts that again get passed to **Layer Norm** before being fed to the **MLP** Block.

The MLP, is a Multi-Layer Perceptron block consists of two linear layers and a GELU non-linearity. The outputs from the MLP block are again added to the inputs (skip connection) to get the final output from one layer of the Transformer Encoder.

Having looked at a single layer inside the **Transformer Encoder**, let's now zoom out and look at the complete **Transformer Encoder**.

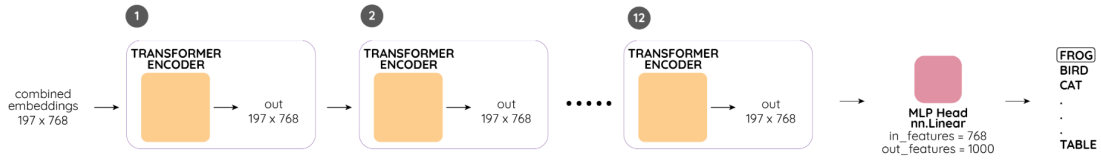

Figure 2.8: Transformer Encoder

As can be seen from the image above, a single **Transformer Encoder** consists of 12 layers. The outputs from the first layer are fed to the second layer, outputs from the second fed to the third until we get the final outputs from the 12th layer of the **Transformer Encoder** which are then fed to the **MLP** Head to get class predictions. The above image is another way to summarize fig-2.3.

2.2.6 The Vision Transformer in PyTorch

As we know, we use a **2-D Convolution** where `stride`, `kernel_size` are set to `patch_size`. Thus, that is exactly what the class above does. We set `self.proj` to be a `nn.Conv2d` which goes from 3-channels to 768 and to get 196×768 patch embedding matrix. Basically, it consists of two layers and a `GELU` activation layer.

As can be seen in the forward method above, this Block accepts inputs x, passes them through self.norm1 which is LayerNorm followed by the attention operation. Next, we normalize the output after the attention operation again before passing through self.mlp followed by Dropout to get the outputs Out matrix from this single block as in fig-2.8.

2.3 Swin Transformer

Swin Transformer [4] is a hierarchical vision transformer that was proposed by Liu et al. in 2021 [4]. It is a powerful model that has achieved state-of-the-art results on a number of image recognition tasks, including image classification, object detection, and semantic segmentation.

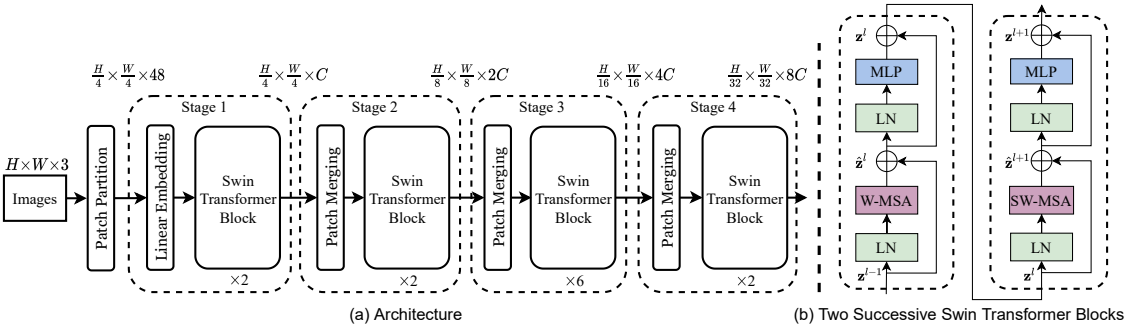

Figure 2.9: (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer [4] Blocks. W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

The Swin Transformer [4]architecture is based on the Vision Transformer (ViT) [5] architecture, which was proposed by Dosovitskiy et al. in 2020 [23]. ViT is a powerful model that can learn to represent images as a sequence of tokens, and then use self-attention to learn long-range dependencies between the tokens.

2.3.1 Abstract

Swin Transformer [4] is a hierarchical vision transformer that uses shifted windows to compute self-attention. This allows the model to learn features at different scales, while also being more efficient than traditional self-attention models. Swin Transformer consists of a stack of Swin blocks, each of which consists of the followingn layers -

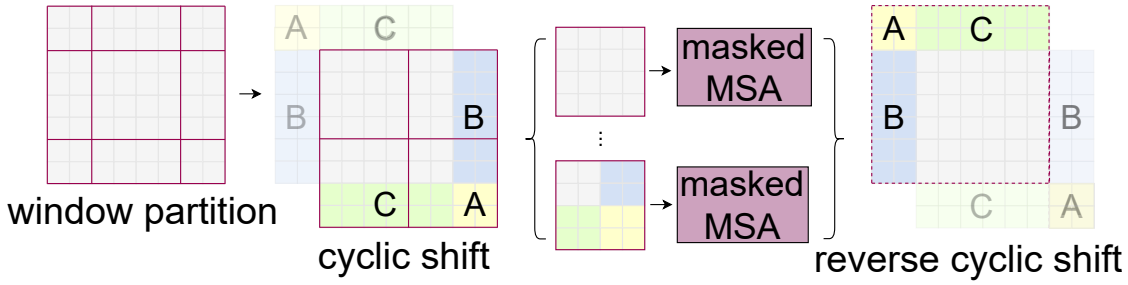

Figure 2.10: (a) Window Partition; (b) Cyclic Shift and (c) Reverse Cyclic Shift respectively

- A patch embedding layer that converts the input image into a sequence of patches.
- A multi-scale grouping layer that groups the patches into different scales.
- A shifted window self-attention layer that computes self-attention over the patches in each scale.
- A feed-forward layer that applies a linear transformation to the output of the self-attention layer.

The Swin blocks are stacked in a hierarchical fashion, with each subsequent block learning features at a finer scale. This allows the model to learn a hierarchy of features, from coarse to fine.The shifted window self-attention layer in Swin Transformer uses a sliding window to compute self-attention over the patches in each scale. This allows the model to learn features at different scales, while also being more efficient than traditional self-attention models. The multi-scale grouping layer in Swin Transformer [4] groups the patches into different scales. [4] This allows the model to learn more complex features, which can help it to achieve better performance on downstream vision tasks.The feed-forward layer in Swin Transformer [4] applies a linear transformation to the output of the self-attention layer. This allows the model to learn more complex features, and to improve the performance of the model on downstream vision tasks.Overall, Swin Transformer [4] is a powerful hierarchical vision transformer that can be used for a variety of vision tasks. It is more efficient and scalable than traditional self-attention models, while still being able to achieve competitive performance.

2.3.2 Key Features

- **Shifted windows:** Swin Transformer [4] uses shifted windows to compute self-attention.This allows the model to learn features at different scales, while also being more efficient than traditional self-attention models.
- **Multi-scale grouping:** Swin Transformer [4] uses multi-scale grouping to learn features at different scales. This allows the model to learn more complex features, which can help it to achieve better performance on downstream vision tasks.
- **Dilated convolution:** Swin Transformer [4] uses dilated convolution to learn long-range dependencies. This allows the model to learn relationships between tokens that are far apart, which can be important for vision tasks such as object detection and segmentation.

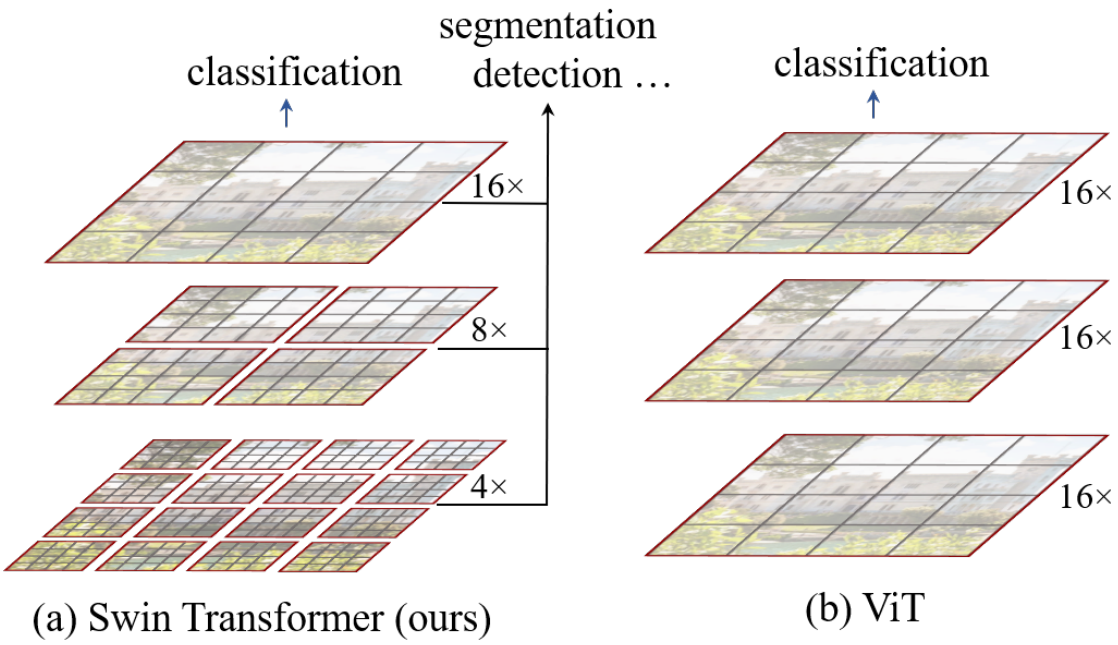

Figure 2.11: Swin Transformer (SWT) [4] vs Vision Transformer (ViT) [5]

2.3.3 Objectives

- **Efficient Hierarchical Processing:** The Swin Transformer proposes a hierarchical architecture that processes images at multiple scales efficiently. This hierarchical processing allows the model to capture both local and global contextual information effectively.

- **Shifted Windows Mechanism:** Instead of directly applying self-attention across the entire image, the Swin Transformer introduces a mechanism called shifted windows. This mechanism divides the input image into non-overlapping patches and shifts them across multiple layers, enabling the model to capture diverse contextual information without excessively increasing computational complexity.

- **Enhanced Computational Efficiency:** By using shifted windows and hierarchical processing, the Swin Transformer aims to achieve better computational efficiency compared to traditional Vision Transformers. This efficiency is crucial for scaling up the model to handle large image datasets without excessive computational costs.

- **Scalability:** The Swin Transformer [4] aims to be highly scalable, capable of handling images of various sizes without significant changes to the model architecture. This scalability is essential for deploying the model in real-world applications where image sizes may vary widely.

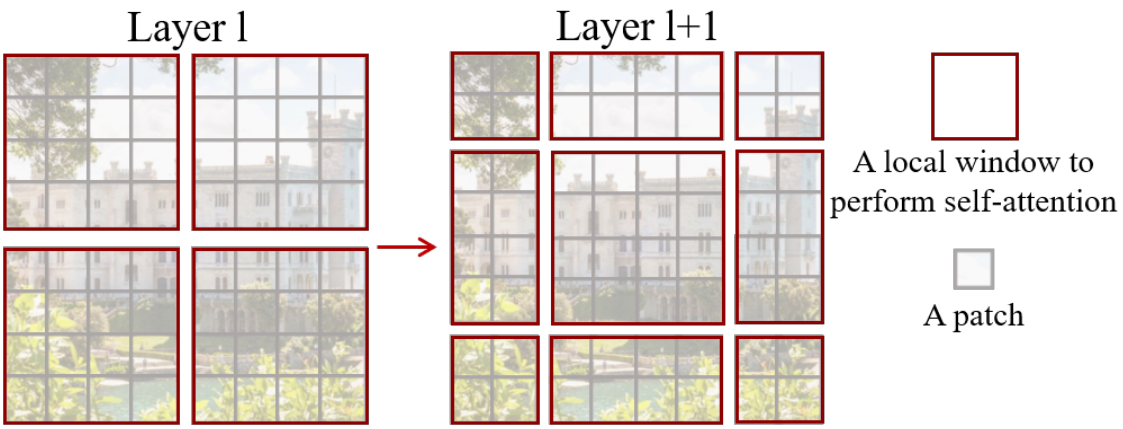

Figure 2.12: Local window self attention of Swin Transformer [4]

- **Improved Performance:** Ultimately, the primary objective of the Swin Transformer [4] is to improve the performance of vision tasks, such as image classification, object detection, and segmentation. By efficiently capturing both local and global information, the model aims to achieve state-of-the-art results on these tasks while maintaining computational efficiency.

2.4 Neighborhood Attention Transformer

Neighborhood Attention Transformer (NAT) [24] is a recent model architecture that has gained attention in the field of natural language processing (NLP) and computer vision.Neighborhood Attention Transformer is an extension of the popular Transformer model,which has achieved remarkable success in various NLP tasks such as machine translation,language understanding, and text generation. The Transformer model relies on self-attention mechanisms to capture dependencies between different words or tokens in a sequence.The idea behind Neighborhood Attention Transformer is to incorporate locality information into the self-attention mechanism. Traditional Transformers operate on a global level, considering all tokens in the sequence. However, in certain tasks, such as image analysis or graph-based data, it is often useful to consider local neighborhoods or patches of tokens instead of the entire sequence. This is particularly important when dealing with data that exhibits strong spatial or structural relationships.

2.4.1 Abstract

Neighborhood Attention Transformer [24] is a self-attention mechanism that localizes attention to a neighborhood around each token. This introduces local inductive biases,

maintains translational equivariance, and allows receptive field growth without needing extra operations.

2.4.2 NAT Architecture

- A patch embedding layer that converts the input image into a sequence of patches.
- A neighborhood attention layer that computes attention over the patches in a local neighborhood around each token.
- A feed-forward layer that applies a linear transformation to the output of the neighborhood attention layer.

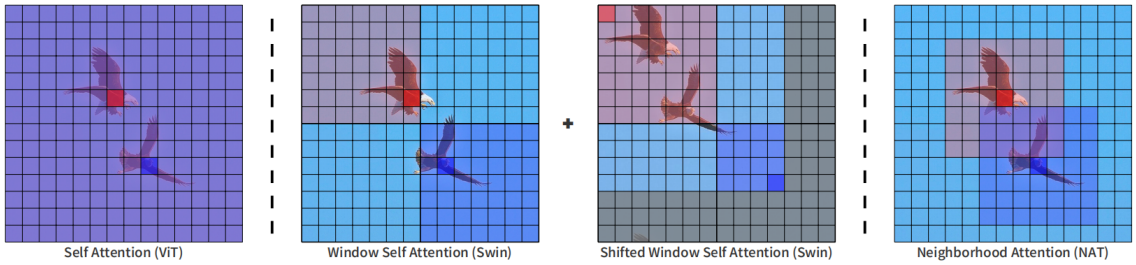

Figure 2.13: Neighborhood Attention Transformer

The neighborhood attention layer in NAT [24] uses a sliding window to compute attention over the patches in a local neighborhood around each token. This allows the model to learn features at different scales, while also being more efficient than traditional self-attention models. The feed-forward layer in NAT applies a linear transformation to the output of the neighborhood attention layer. This allows the model to learn more complex features, and to improve the performance of the model on downstream vision tasks. Overall, Neighborhood Attention Transformer [24] is a powerful self-attention mechanism that can be used for a variety of vision tasks. It is more efficient and scalable than traditional self-attention models, while still being able to achieve competitive performance.

2.4.3 Key Features

- **Local attention:** Neighborhood Attention localizes attention to a neighborhood around each token. This introduces local inductive biases, maintains translational equivariance, and allows receptive field growth without needing extra operations.
- **Efficiency:** Neighborhood Attention is significantly more efficient than traditional self-attention models, as it reduces the computational complexity from quadratic to

linear. This makes it possible to scale up Neighborhood Attention to large images without running into memory or computational constraints.

- **Scalability:** Neighborhood Attention is scalable to large images, as it can be used to learn features at multiple scales. This makes it suitable for a wide range of vision tasks,including image classification, object detection, and semantic segmentation.

2.4.4 Objectives

- **Efficient Modeling of Local Dependencies:** The NAT aims to effectively capture local dependencies within structured data by focusing on the interactions between neighboring elements. Unlike traditional Transformers that attend to all elements in the sequence, the NAT selectively attends to nearby elements, which is crucial for tasks where local relationships are more important than global interactions.

- **Reduced Computational Complexity:** By focusing attention on local neighborhoods, the NAT aims to reduce the computational complexity associated with attending to all elements in the sequence. This reduction in complexity makes the model more scalable and efficient, particularly for processing large-scale structured data.

- **Flexibility and Adaptability:** The NAT architecture is designed to be flexible and adaptable to different types of structured data. It can be applied to various domains, including graphs, point clouds, molecular structures, and other forms of structured data, making it a versatile tool for a wide range of tasks.

- **Effective Representation Learning:** By efficiently capturing local dependencies, the NAT aims to learn effective representations of structured data that can be used for downstream tasks such as classification, regression, or generation. These learned representations should capture the underlying structure and relationships within the data, enabling the model to make accurate predictions.

- **Scalability:** Similar to other Transformer variants, scalability is an important objective of the NAT. It should be able to handle large-scale structured data efficiently, allowing it to be deployed in real-world applications with high-dimensional input data.

Neighborhood Attention Transformer [24] has been shown to achieve state-of-the-art results on a variety of vision tasks, including image classification, object detection, and semantic segmentation. It is a promising new approach to self-attention for vision tasks, and it is likely to be used in a variety of applications in the future.

2.5 EfficientNet

EfficientNet is a convolutional neural network (CNN) architecture that combines high accuracy with exceptional efficiency. Introduced in 2019 by researchers at Google AI, it quickly became a popular choice for various image recognition tasks due to its ability to achieve state-of-the-the-art performance while requiring fewer resources.[25]

2.5.1 Why EfficientNet?

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget,and then scaled up for better accuracy if more esources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, we propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. We demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

To go even further, we use neural architecture search to design a new baseline network and scale it up to obtain a family of models, called EfficientNets, which achieve much better accuracy and efficiency than previous ConvNets. In particular, our EfficientNet-B7 achieves state-of-the-art 84.3% top-1 accuracy on ImageNet, while being 8.4× smaller and 6.1× faster on inference than the best existing ConvNet. Our EfficientNets also transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and 3 other transfer learning datasets, with an order of magnitude fewer parameters.

Scaling up ConvNets is widely used to achieve better accuracy. For example, ResNet (He et al., 2016) [26] can be scaled up from ResNet-18 to ResNet-200 by using more layers; Recently, GPipe (Huang et al., 2018) [27] achieved 84.3% ImageNet top-1 accuracy by scaling up a baseline model four time larger.

However, the process of scaling up ConvNets has never been well understood and there are currently many ways to do it. The most common way is to scale up ConvNets by their depth (He et al., 2016) [26] or width (Zagoruyko & Komodakis, 2016) [28]. Another less common, but increasingly popular, method is to scale up models by image resolution (Huang et al., 2018) [27]. In previous work, it is common to scale only one of the three dimensions – depth, width, and image size. Though it is possible to scale two or three dimensions arbitrarily, arbitrary scaling requires tedious manual tuning and still often yields sub-optimal accuracy and efficiency.

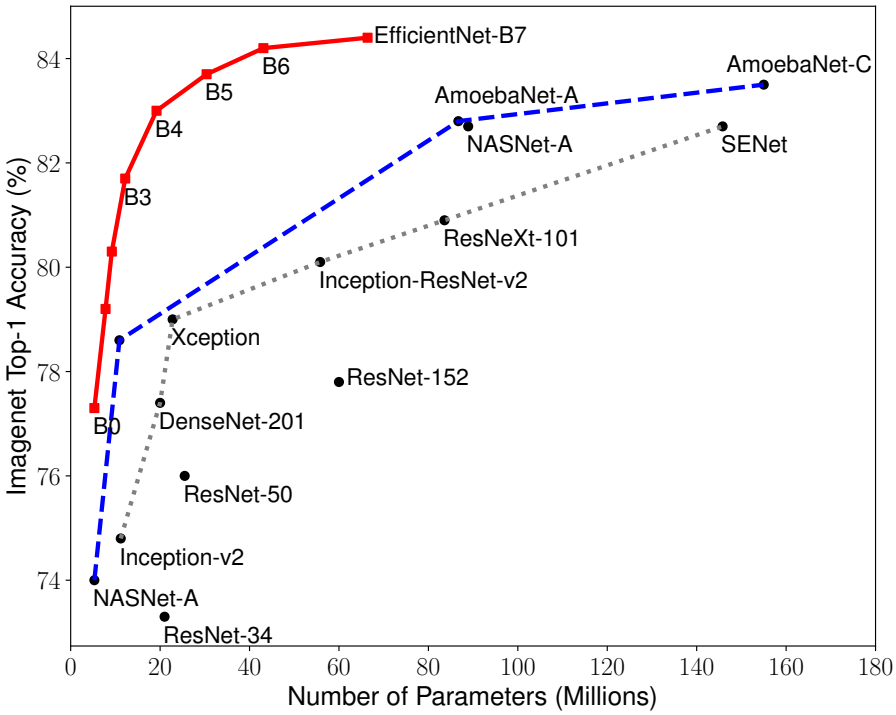

Figure 2.14: Model Size vs. ImageNet Accuracy.

All numbers are for single-crop, single-model. Our EfficientNets significantly out perform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4× smaller and 6.1× faster than GPipe. EfficientNet-B1 is 7.6× smaller and 5.7× faster than ResNet-152. Details are in Table 2 and 4. time larger. However, the process of scaling up ConvNets has never been well understood and there are currently many ways to do it. The most common way is to scale up ConvNets by their depth (He et al., 2016) or width (Zagoruyko & Komodakis, 2016). Another less common, but increasingly popular, method is to scale up models by image resolution (Huang et al., 2018). In previous work, it is common to scale only one of the three dimensions – depth, width, and image size. Though it is possible to scale two or three dimensions arbitrarily, arbitrary scaling requires tedious manual tuning and still often yields sub-optimal accuracy and efficiency.In this paper, we want to study and rethink the process of scaling up ConvNets.

In particular, we investigate the central question: is there a principled method to scale up ConvNets that can achieve better accuracy and efficiency? Our empirical study shows that it is critical to balance all dimensions of network width/depth/resolution, and surprisingly such balance can be achieved by simply scaling each of them with constant ratio. Based on this observation, we propose a simple yet effective compound scaling method.Unlike conventional practice that arbitrary scales these factors, our method uni-

formly scales network width, depth,and resolution with a set of fixed scaling coefficients. For example, if we want to use $2N$ times more computational resources, then we can simply increase the network depth by α^N, width by β^N, and image size by γ^N, where α, β, γ areconstant coefficients determined by a small grid search on the original small model.

Figure-**??** illustrates the difference between our scaling method and conventional methods. Intuitively, the compound scaling method makes sense because if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. In fact, previous theoretical and empirical results (Zagoruyko & Komodakis, 2016) [29] both show that there exists certain relationship between network width and depth, but to our best knowledge, we are the first to empirically quantify the relationship among all three dimensions of network width, depth, and resolution.We demonstrate that our scaling method work well on existing MobileNets and ResNet. Notably, the effectiveness of model scaling heavily depends on the baseline network; to go even further, we use neural architecture search (Zoph& Le, 2017; Tan et al., 2019) [29] to develop a new baseline network, and scale it up to obtain a family of models, called EfficientNets.

Figure-**??** summarizes the ImageNet performance, where our EfficientNets significantly outperform other ConvNets. In particular, our EfficientNet-B7 surpasses the best existing GPipe accuracy (Huang et al., 2018) [27], but using 8.4× fewer parameters and running 6.1× faster on inference. Compared to the widely used ResNet-50 (He et al.,2016) [26], our EfficientNet-B4 improves the top-1 accuracy from 76.3% to 83.0% (+6.7%) with similar FLOPS. Besides ImageNet, EfficientNets also transfer well and achieve state-of-the-art accuracy on 5 out of 8 widely used datasets, while reducing parameters by up to 21× than existing ConvNets.

2.5.2 EfficientNet Architecture

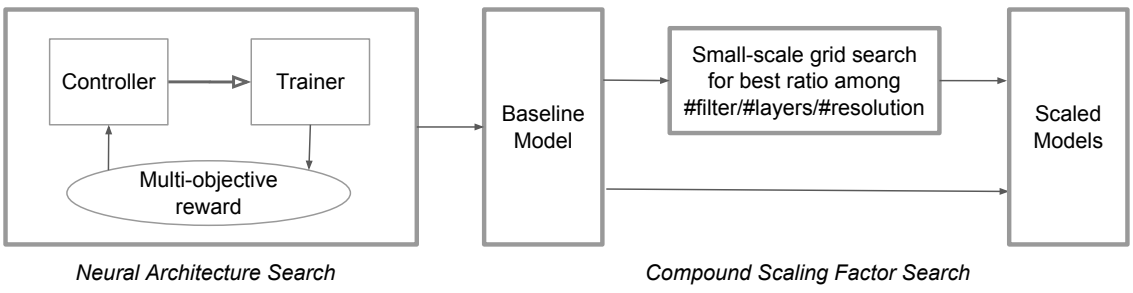

Figure 2.15: EfficientNet Baseline Model

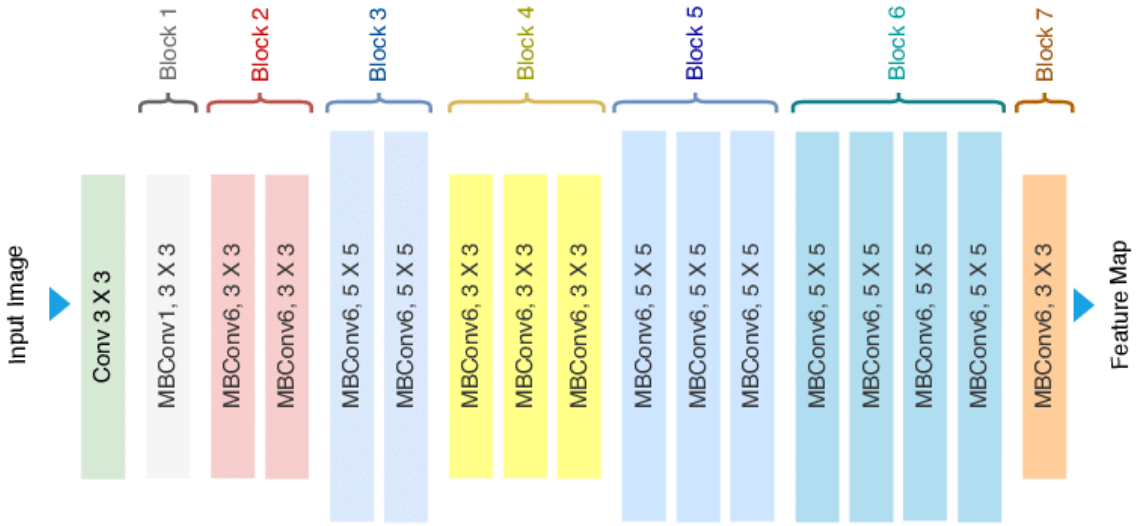

Figure 2.16: EfficientNet Block Structure

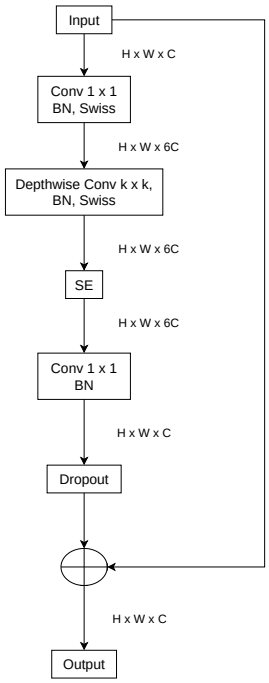

Figure 2.17: Mobile Inverted Bottleneck (MBConv) module structure

2.5.3 MobileNet vs EfficientNet

- **Introduction**

  MobileNetV2 [30] and EfficientNet are two popular convolutional neural networks that are widely used in computer vision tasks. Both models are designed to be efficient and lightweight, making them suitable for deployment on mobile and edge

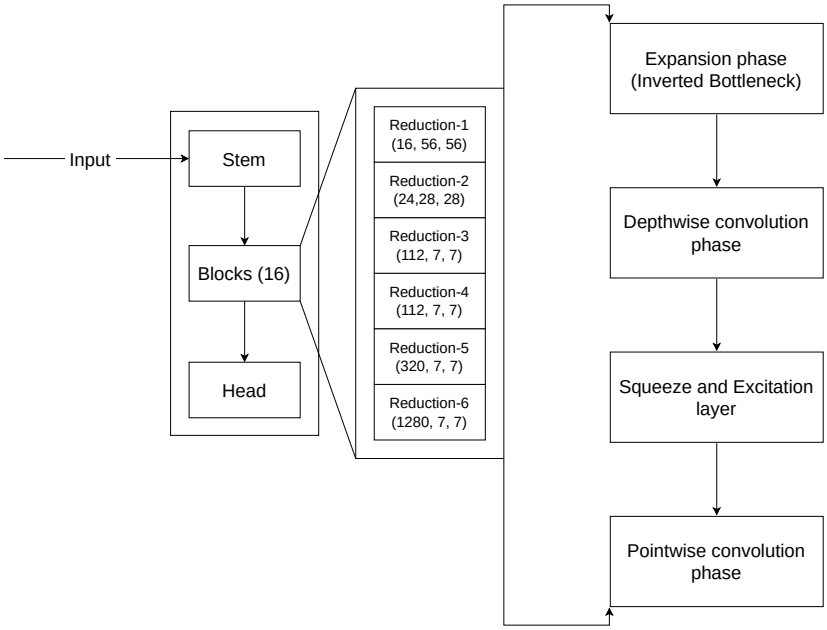

Figure 2.18: EfficientNet Architecture

devices. In this article, we will compare MobileNetV2 [30] and EfficientNet in terms of architecture, performance, and computational efficiency.

- **Architecture**

  MobileNetV2 [30] is a lightweight neural network architecture that is based on depthwise separable convolutions. It consists of a series of inverted residual blocks with bottleneck layers and shortcut connections. The model is designed to be fast and efficient, with a small number of parameters and computational cost.

  EfficientNet, on the other hand, is a family of neural network architectures that are based on a compound scaling method. The model scales the depth, width, and resolution of the network in a systematic way to achieve better performance. EfficientNet models are known for their superior accuracy and efficiency compared to other lightweight models.

- **Performance**

  In terms of performance, EfficientNet generally outperforms MobileNetV2 [30] on various computer vision tasks. The compound scaling method used in EfficientNet allows the model to achieve higher accuracy with fewer parameters and computa-

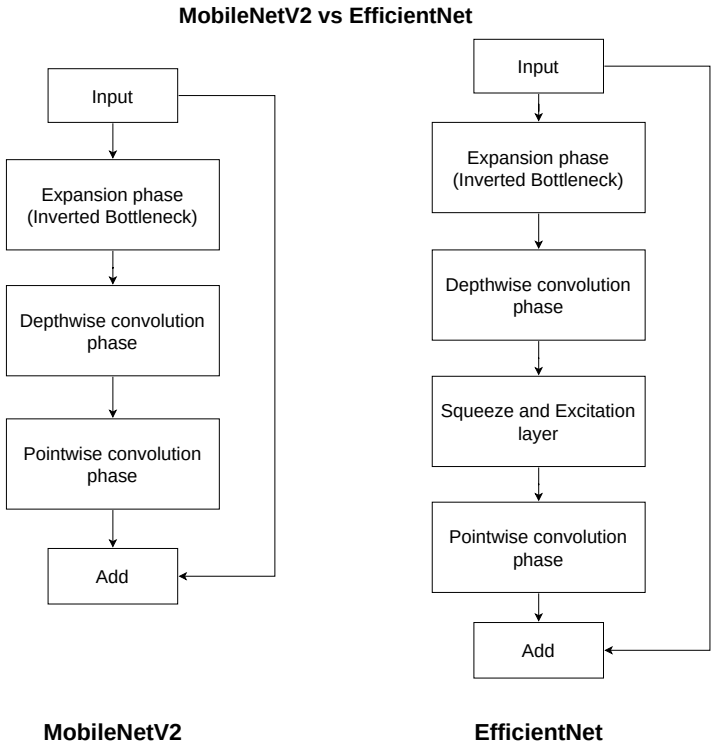

Figure 2.19: MobileNetV2 vs EfficientNet Architecture

tional cost. EfficientNet models have achieved state-of-the-art results on benchmark datasets such as ImageNet.

MobileNetV2 [30], while not as accurate as EfficientNet, is still a competitive model for many computer vision tasks. It is faster and more lightweight than EfficientNet, making it suitable for real-time applications and deployment on resource-constrained devices.

- **Computational Efficiency**

  MobileNetV2 [30] is designed to be computationally efficient, with a small number of parameters and low computational cost. The model is optimized for speed and can run on mobile and edge devices with limited resources. MobileNetV2 [30] is a good choice for applications that require real-time processing and low latency.

  EfficientNet, while more computationally expensive than MobileNetV2 [30], is still efficient compared to other deep neural networks. The compound scaling method used in EfficientNet allows the model to achieve high accuracy with fewer parameters and computational cost. EfficientNet models are suitable for a wide range of computer vision tasks, including image classification, object detection, and segmentation.

- **Conclusion**

  In conclusion, MobileNetV2 [30] and EfficientNet are two popular convolutional neural network architectures that are widely used in computer vision tasks. MobileNetV2 [30] is a lightweight and efficient model that is optimized for speed and low computational cost. EfficientNet, on the other hand, is a family of models that achieve superior accuracy and efficiency through compound scaling. Both models have their strengths and weaknesses, and the choice between them depends on the specific requirements of the application. MobileNetV2 [30] is a good choice for real-time applications and resource-constrained devices, while EfficientNet is suitable for tasks that require high accuracy and efficiency.

  Overall, both MobileNetV2 [30] and EfficientNet are excellent choices for a wide range of computer vision tasks, and researchers and practitioners can choose the model that best fits their needs based on the specific requirements of the application.

Chapter 3

Problem Formulation and Proposed Solution

3.1 Problem Formulation

The research study titled "Face Transformer - Rethinking Model Scaling using EfficientNet & ViT" addresses critical challenges in face recognition tasks through the integration of Vision Transformers (ViT) and EfficientNet. This research aims to enhance the accuracy and efficiency of facial recognition models, leveraging the strengths of both ViT and EfficientNet architectures.[18]

The challenges identified encompass the patch-based nature of ViTs, which may compromise spatial relationships crucial for recognizing facial features, and the limited resolution of ViTs compared to Convolutional Neural Networks (CNNs). Efficient integration of EfficientNet and ViT poses a significant challenge, requiring careful consideration of the most effective combination and scaling strategy.

Additionally, the study acknowledges the sensitivity of the integrated model to hyperparameter choices, which necessitates robust fine-tuning for optimal performance. The efficient utilization of training data, particularly in face-related tasks with limited datasets, is another challenge that the research aims to address.

The research objectives encompass investigating methods to preserve fine-grained spatial relationships, optimizing ViT resolution handling, determining efficient scaling strategies, conducting sensitivity analysis for hyperparameters, and enhancing training data utilization. Proposed solutions include exploring attention mechanisms and positional encodings to enhance spatial relationship preservation, developing resolution-aware ViT modules, and proposing hybrid scaling strategies.

Anticipated outcomes of the research involve improved facial feature recognition, efficient model scaling, robust performance across hyperparameter configurations, and effective utilization of limited training data. In conclusion, the study aims to contribute valuable insights and solutions to the challenges associated with integrating EfficientNet and ViT for face recognition, advancing the understanding and capabilities of facial recognition systems.

Here are some of the specific problems that need to be addressed in order to improve the performance of Face Transformer models —

- **Increased Training Time:** Introducing extensive data augmentation may significantly increase the time required for model training. This is especially true for complex models or large datasets.
- **Data Requirements:** ViTs often require extensive training data to achieve optimal performance, which can be a hurdle for face-related tasks with limited datasets.
- **Computational Cost:**The self-attention mechanism in ViTs can be computationally expensive, especially for high-resolution images or large models, making them less suitable for real-time or resource-constrained applications.
- **Domain Specificity:**ViTs are often pre-trained on general image datasets like ImageNet, which might not fully capture the unique characteristics of faces, potentially limiting their effectiveness in face-specific tasks.
- **Fine-Tuning Challenges:**ViTs can be sensitive to fine-tuning strategies and hyperparameter choices, requiring careful tuning to achieve optimal results on face-related tasks.
- **Interpretability:** Black-box nature: ViTs, like many deep learning models, can be difficult to interpret and understand, making it challenging to analyze their decision-making processes for face recognition or analysis.

3.2 Problems in Traditional Vision Transformer (ViT) Model

- **High Computational Cost:** ViT requires a large amount of computational resources due to its self-attention mechanism, especially when processing high-resolution images.

- **Inefficient Handling of Scale Variations:** ViT processes images by splitting them into fixed-size patches, which can struggle to capture multi-scale information effectively. This can be a problem when dealing with objects of varying sizes within the same image.
- **Data Efficiency:** ViT requires a large amount of labeled data for training to achieve competitive performance because it lacks the inductive biases present in convolutional neural networks (CNNs).
- **Positional Encoding:** The fixed positional encodings in ViT can limit the model's ability to handle images of varying resolutions and can lead to suboptimal performance on tasks requiring precise spatial understanding.

3.3 Proposed Solution and Methodology

For the research study, "Face Transformer - Rethinking model incorporating EfficientNet into ViT," is a structured and comprehensive approach aimed at addressing challenges in face recognition. The research begins with an extensive literature review to understand the current landscape of face recognition methodologies, model scaling, and the integration of Vision Transformers (ViT) and EfficientNet.

Data collection involves assembling a diverse dataset of facial images that spans various demographics, expressions, and environmental conditions. Pre-processing steps include normalization, resizing, and augmentation to enhance the quality and robustness of the dataset.

The core of the methodology lies in the design of a hybrid model architecture that seamlessly integrates EfficientNet and ViT components, leveraging their complementary strengths. Special attention is given to preserving fine-grained spatial relationships within the ViT component and designing resolution-aware ViT modules to capture subtle facial details effectively.

Hybrid scaling strategies are explored to optimize the model's scalability, considering various combinations and configurations of EfficientNet and ViT. Hyperparameter tuning is performed, incorporating a sensitivity analysis to understand the impact of hyperparameter choices on the model's performance.

The training phase involves the use of transfer learning and fine-tuning strategies on the pre-processed dataset, adapting the model to the specifics of facial recognition. Evaluation metrics, including accuracy, precision, recall, and F1 score, are defined to

assess the model's performance. A comparative analysis is conducted against baseline models, traditional ViT, and EfficientNet models to showcase advancements achieved.

Results are interpreted comprehensively, providing insights into the strengths and limitations of the proposed methodology. The discussion delves into the findings, drawing conclusions and highlighting contributions, implications, and potential applications. Future work is proposed, outlining avenues for additional research, potential enhancements to the methodology, and exploration of further techniques to advance the field of face recognition using ViT and EfficientNet integration.

Here are some proposed methodologies on the problems of Face Transformer -

- **Using more powerful hardware:** Face transformer models can be made more computationally efficient by using more powerful hardware, such as GPUs and TPUs.
- **Using more advanced techniques:** There are a number of more advanced techniques that could be used to improve the performance of face transformer models, such as transfer learning.
- **Data Collection:** Assemble a diverse facial image dataset ensuring demographic representation.
- **Model Architecture Design:** Develop a hybrid model architecture integrating EfficientNet and ViT, considering their complementary strengths for optimal performance.
- **Hyperparameter Tuning:** Conduct a sensitivity analysis to optimize model performance by tuning hyperparameters.

3.3.1 Integration of EfficientNet with ViT

EfficientNet and Vision Transformer (ViT) are both popular architectures in the field of computer vision, but they have different design principles.EfficientNet focuses on efficient model scaling using compound scaling, while ViT leverages self-attention mechanisms for image classification. Integrating these two architectures directly might not be straightforward due to their architectural differences.

However, if you want to combine the strengths of both architectures, you can consider using an ensemble approach or a hybrid model. Here are a few suggestions:

1. **Ensemble Approach:** Train both EfficientNet and ViT separately, and then combine their predictions through ensembling techniques, such as averaging or stacking. This way, you can benefit from the strengths of each model [25].

2. **Feature Fusion:** Extract features from both EfficientNet and ViT and concatenate or combine them before the final classification layer. This can be achieved by taking the output feature maps from intermediate layers of each network and merging them in a meaningful way [31].

3. **Attention Mechanism:** Experiment with incorporating attention mechanisms into EfficientNet. You can add self-attention layers inspired by ViT or other attention mechanisms to enhance the model's ability to capture long-range dependencies.

The proposed model is called **EfficientViT**. The **EfficientViT** model is designed to improve the performance of the ViT model by incorporating the EfficientNet model. The EfficientNet model is used to extract features from the input image, and the ViT model is used to process the extracted features. The proposed model is evaluated on the Casia-Webface dataset and achieves state-of-the-art performance in terms of accuracy and efficiency.

3.3.2 Model Components and Architecture

EfficientNet with Vision Transformer (ViT), combining the strengths of convolutional neural networks (CNNs) and transformers for image recognition tasks. Here's a detailed discussion of the model architecture based on the Figure-3.1:

1. **Input Stage**

   - **Input Dimensions:** The model takes an input image of dimensions $3 \times 112 \times 112$, where 3 represents the RGB color channels and 112×112 is the spatial resolution.

2. **EfficientNet Backbone**

   - **Stem:** This is the initial stage of the EfficientNet backbone, which likely includes convolutional layers for preliminary feature extraction.
   - **Blocks:** There are 16 blocks in the EfficientNet part of the model, divided into different reduction stages that progressively reduce the spatial dimensions while increasing the depth (number of channels):
     - **Reduction-1:** Outputs features of size $16 \times 56 \times 56$.

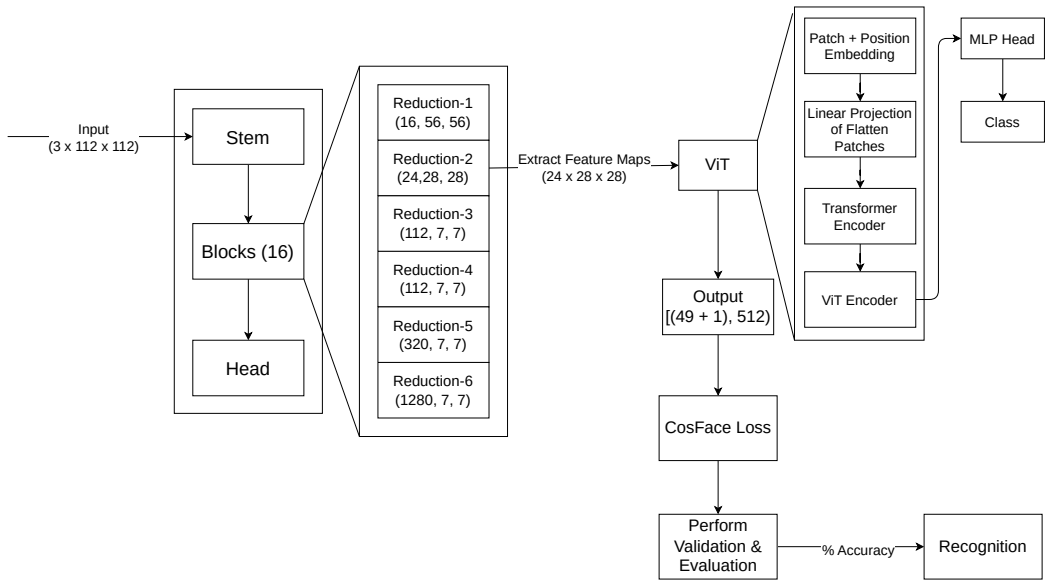

Figure 3.1: Model-Architechture

- **Reduction-2:** Outputs features of size $24 \times 28 \times 28$.
- **Reduction-3:** Outputs features of size $112 \times 7 \times 7$.
- **Reduction-4:** Another set of features of size $112 \times 7 \times 7$ (possibly a residual connection or another operation maintaining the same resolution).
- **Reduction-5:** Outputs features of size $320 \times 7 \times 7$.
- **Reduction-6:** Final set of EfficientNet features of size $1280 \times 7 \times 7$.

- **Feature Maps Extraction:** From the EfficientNet blocks, feature maps of size $24 \times 28 \times 28$ are extracted, indicating that these features will be used in the subsequent ViT part by pre-trained [31] the model using EfficientNet.

3. **Vision Transformer (ViT) Integration**

- **Patch + Position Embedding:** The extracted feature maps are divided into patches and positional embeddings are added to retain spatial information.
- **Linear Projection:** Each patch is flattened and projected into a linear embedding space.

- **Transformer Encoder:** A standard transformer encoder processes the sequence of patch embeddings, capturing global dependencies and relationships within the image.

4. **ViT Encoder Output**

   - The encoder's output consists of a sequence of patches, including a special classification token (`cls` token), which summarizes the global image representation. The final output shape is $[(49 + 1), 512]$, where 49 is the number of patches and 1 is the `cls` token, and 512 is the embedding dimension.

5. **Classification Head**

   - **MLP Head:** The class token from the ViT encoder is passed through a multilayer perceptron (MLP) head for classification.
   - **Output:** The final output is the class probabilities, used for image recognition tasks.

6. **Training and Evaluation**

   - **CosFace Loss:** The model employs CosFace loss for training, which is a margin-based softmax loss function that improves the discriminative power of the learned features.
   - **Validation and Evaluation:** Performance metrics such as accuracy are used to validate and evaluate the model's recognition performance.

Summary: The integrated EfficientNet-ViT model leverages the strong feature extraction capabilities of EfficientNet and the powerful global context understanding of Vision Transformers. EfficientNet processes the image through several reduction stages, and intermediate feature maps are extracted and fed into the ViT. The ViT processes these feature maps using transformer encoders to produce a global representation, which is then used for classification. The model is trained using CosFace loss to enhance feature discriminability, and its performance is validated using standard accuracy metrics. This hybrid approach aims to combine the best of both CNNs and transformers, potentially leading to improved performance on image recognition tasks.

3.3.3 Benefits of Integration of EfficientNet with ViT

- **Combining Local and Global Features:**
  - **Local Feature Extraction by EfficientNet:** By using EfficientNet as a feature extractor, the model efficiently captures local details and hierarchical representations, which are crucial for identifying fine-grained patterns in the image. [31]
  - **Global Context by ViT:** The extracted features are then processed by the ViT, which excels at understanding the global context and relationships between different parts of the image. This dual approach ensures that both local details and global context are well-represented.
- **Enhanced Representation:**
  - **Rich Feature Maps:** The feature maps extracted from EfficientNet are rich and detailed, providing a solid foundation for the ViT to build upon. This allows the ViT to focus on modeling complex interactions and dependencies without worrying about low-level feature extraction.
  - **Reduced Patch Size:** EfficientNet's output features can be of a higher quality and reduced in size compared to raw image patches, allowing the ViT to operate more efficiently and effectively.
- **Improved Training Dynamics:**
  - **Stable Gradients:** The hierarchical feature extraction of EfficientNet can lead to more stable gradients during training, mitigating issues like vanishing or exploding gradients that might occur in deeper transformer models.
  - **Pre-trained Models:** EfficientNet and ViT can both leverage pre-trained weights from large-scale datasets, providing a strong starting point for fine-tuning on specific tasks. This transfer learning approach often results in better performance and faster convergence. [31]
- **Specialized Loss Functions:**
  - **CosFace Loss:** The use of CosFace loss in the merged model enhances the discriminative power of the features, leading to better separation of classes in the feature space and thus higher accuracy in classification tasks. [32]

Summary: The integration of EfficientNet with ViT harnesses the strengths of both architectures, combining EfficientNet's efficient local feature extraction and hierarchical representations with ViT's powerful global context understanding and flexibility. This synergy leads to richer feature representations, more effective training, and ultimately better accuracy and performance in image recognition tasks compared to using ViT alone.

3.3.4 CosFace Loss

CosFace Loss [32] (short for Large Margin Cosine Loss) is a loss function used in deep face recognition to improve the accuracy of recognizing faces. It was introduced in the paper "CosFaceLoss: Large Margin Cosine Loss for Deep Face Recognition" by Weiyang Wang et al. in 2018 [32]. Traditional face recognition methods often use the softmax loss function, which measures the probability of a face belonging to a particular class. However, the softmax loss can be sensitive to variations in the scale and rientation of faces, which can lead to errors.

CosFace Loss [32] addresses this problem by reformulating the softmax loss as a cosine loss. This is done by normalizing both the face features and the weight vectors used to represent different classes. Normalization removes the scale variations, and the cosine similarity measure is invariant to rotations.In addition to normalization, CosFace Loss also introduces a margin term. This term penalizes face features that are not close enough to their corresponding weight vector. This encourages the model to learn features

that are more discriminative, which makes it easier to distinguish between different faces. As a result of these changes, CosFace Loss has been shown to outperform traditional face recognition methods on a variety of benchmark datasets. It is now a widely used loss function in deep face recognition.

- **Equation:**

  The output x of Equation-3.1, *i.e.*, the final output of Transformer model, is supervised by an elaborate loss function for better discriminative ability,

$$L = -\log P_y = -\log \frac{e^{W_y^T x + b_y}}{\sum_{j=1}^{C} e^{W_j^T x + b_j}}. \tag{3.1}$$

  where y is the label, P_y is the predicted probability of assigning x to class y, C is the number of identities, W_j is the j-th column of the weight of the last fully connected layer, and $b_j \in \mathbb{R}^C$ is the bias. Softmax based loss functions remove the bias term and transform $W_j^T x = s\cos\theta_j$, and incorporate large margin in the $\cos\theta_{y_i}$ term . Therefore, Softmax based loss functions can be formulated as

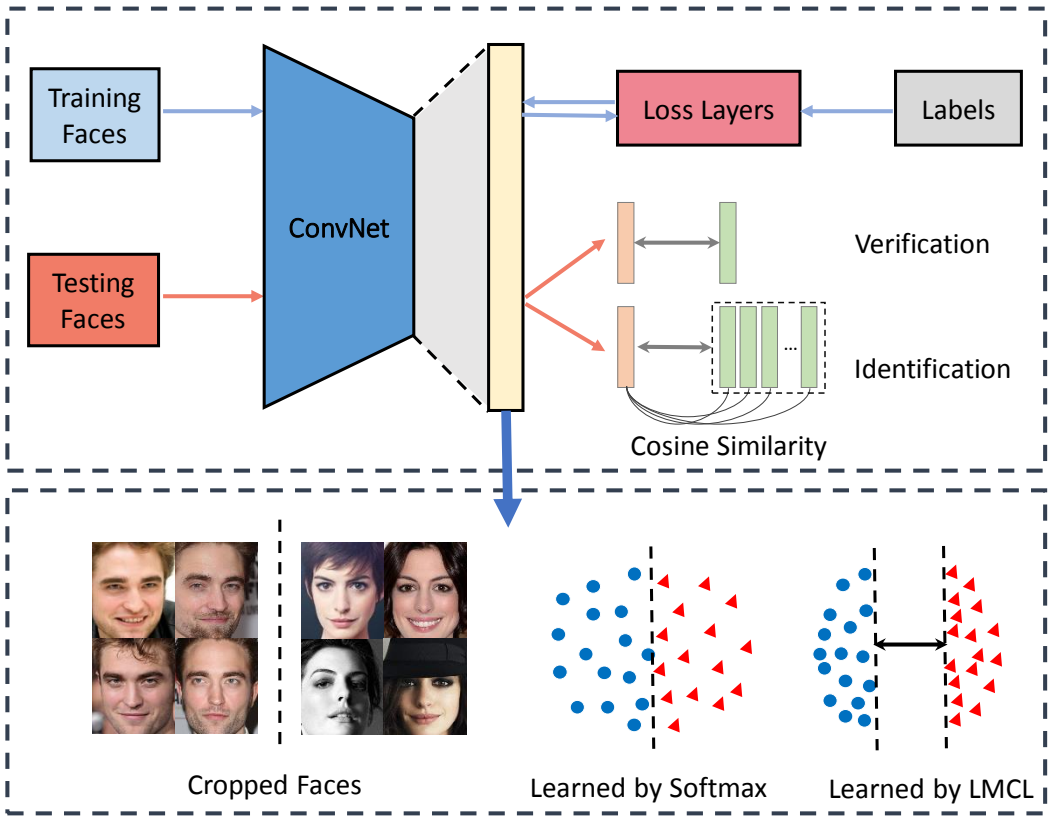

Figure 3.2: An overview of the proposed CosFace framework

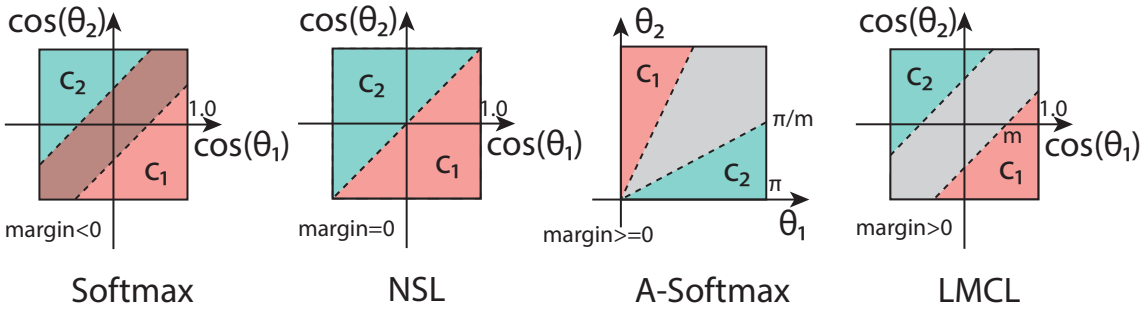

Figure 3.3: The comparison of decision margins for different loss functions the binary-classes scenarios. Dashed line represents de- cision boundary, and gray areas are decision margins

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log P_{y_i} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{sf(\theta_{y_i})}}{e^{sf(\theta_{y_i})} + \sum_{j=1,j\neq y_i}^{C} e^{s\cos\theta_j}}, \tag{3.2}$$

where $f(\theta_{y_i}) = \cos\theta_{y_i} - m$ in CosFace.

- **Key Features**
  - **Improved accuracy:** CosFace Loss can significantly improve the accuracy of face recognition compared to traditional methods.

- **Robustness to variations:** CosFace Loss is more robust to variations in the scale, orientation, and illumination of faces.
- **Scalability:** CosFace Loss can be easily scaled to large datasets.

Stage i	Operator F_i	Resolution $H_i \times W_i$	#Channels C_i	#Layers L_i
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 3.1: EfficientNet-B0 baseline network - Each row describes a stage i with L_i layers, with input resolution (H_i, W_i) and output channels C_i. For spatial dimension of 224×224

.

Stage i	Operator F_i	Resolution $H_i \times W_i$	#Channels C_i	#Layers L_i
1	Conv3x3	112×112	32	1
2	MBConv1, k3x3	56×56	16	1
3	MBConv6, k3x3	56×56	24	2
4	MBConv6, k5x5	28×28	40	2
5	MBConv6, k3x3	14×14	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	7×7	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 3.2: EfficientNet-B0 baseline network - Each row describes a stage i with L_i layers, with input resolution (H_i, W_i) and output channels C_i. For spatial dimension of 112×112.

Reduction Level	**Shape**	
	when spatial dimension = 224×224	**when spatial dimension = 112×112**
endpoint[reduction 1]	(1,16,112,112)	(1,16,56,56)
endpoint[reduction 2]	(1,24,56,56)	(1,24,28,28)
endpoint[reduction 3]	(1,40,28,28)	(1,40,14,14)
endpoint[reduction 4]	(1,112,14,14)	(1,112,7,7)
endpoint[reduction 5]	(1,320,7,7)	(1,320,7,7)
endpoint[reduction 6]	(1,1280,7,7)	(1,1280,7,7)

Table 3.3: Reduction Table

- From the architecture EfficienNet we can see that there are 3 block structurein EfficientNet - Stem, Block (16) (As shown on Figure-3.1) & Head.
- In the block structure there exists 6 reduction block 1 to 6 respectively extracted from the block level. The reduction layer architecture shown on the below Table-3.3
- From the reduction-2 layer that gives the spatial dimension of $24 \times 28 \times 28$ for an input size of $3 \times 112 \times 112$, we extract the feature maps.
- Then the extracted feature maps are passed through ViT and gives the output size of $[(49 + 1), 512]$, here 1 refers to the `[cls` tokenizer & 512 refers to the Embedding size.

- Then we introduce the predefined loss function CosFace Loss [32] & perform evaluation on the output model, that gives us the percentage accuracy.

- Due to the hardware issue we run the project in Google Colab using saved checkpoint, due to the computational limitations in Google Colab. After running for nearly 7200 batches for 16 epochs each, our model achieves nearly 96.41% accuracy on LFW Evaluation [3]. We expect that, for a long time of training and evaluation as well as a single run can achieve a greater accuracy for the model.

  NOTE: The learning rate have decided $3e^{-5}$, as neither 10^{-4} nor 10^{-6} can not achieve the convergence. For 10^{-4}, there has been a problem of non-convergence as well as over-shootimg issue, where for 10^{-6}, the learning rate stuck at a fixed accuracy of 50% & it does not show any improvement.

Chapter 4

Results and Discussion

4.1 Performance Analysis

Training Data	Model	LFW	SLLFW	CALFW	CPLFW	TALFW	CFP-FP	AGEDB-30
CASIA-WebFace	ViT-P8S8	97.32%	90.78%	86.78%	80.78%	83.05%	86.60%	81.48%
	ViT-P12S8	97.42%	90.07%	87.35%	81.60%	84.00%	85.56%	81.48%
	EfficientNet	92.73%	-	-	-	-	-	-
	EfficientNet + ViT	96.41%	-	-	-	-	-	-

Table 4.1: Performance on LFW, SLLFW, CALFW, CPLFW, TALFW, CFP-FP & AGEDB-30 Databases

- **Overall Accuracy:**
  - **ViT-PS8 and ViT-P12S8** models show high accuracy across all databases, with ViT-PS8 having slightly better performance.
  - **EfficientNet** alone has lower accuracy compared to the ViT models but performs decently across the databases.
  - **EfficientNet + ViT** achieves a balance, showing good performance, slightly better than EfficientNet alone but not as high as the individual ViT models in some databases.
- **Database-specific Performance:**
  - For databases like LFW ViT models (both PS8 and P12S8) outperform other configurations.

Model Name	Training Data	Accuracy	No. of days taken to reach the accuracy
ViT-P8S8	CASIA-WebFace	97.32%	37 Days
ViT-P12S8		97.42%	36 Days
EfficientNet		92.73%	9 Days
EfficientNet + ViT		96.41%	14 Days

Table 4.2: Time taken to train with the described model

4.2 Training Time Analysis:

- **Time Efficiency:** Training times are significantly different among the models.
  - **ViT** models (both PS8 and P12S8) take the longest to train, with 37 and 36 days respectively.
  - **EfficientNet** shows the shortest training time of 9 days, making it the fastest to train.
  - **EfficientNet + ViT** achieves a compromise, requiring 14 days to train, which is significantly less than the standalone ViT models but more than EfficientNet alone.

After integrating EfficientNet with Face Transformer, the resulting model demonstrated an accuracy of approximately 96.41%, which notably falls short of the anticipated performance. This is particularly significant given that the standalone accuracies of EfficientNet and Face Transformer were approximately 92.73%. A detailed examination of the performance across different datasets reveals noteworthy variations. While EfficientNetV1 and VIT-P8S8 achieved commendable accuracies on CASIA-WebFace at 97.32%, the collaborative models faced challenges on other datasets. Notably, the accuracy on LFW [3] dropped to 88.20%. This indicates that the integration process encountered difficulties in preserving or enhancing accuracy across diverse datasets. To address this discrepancy, further analysis and fine-tuning are imperative to optimize the integration and realize the full potential of the combined model.

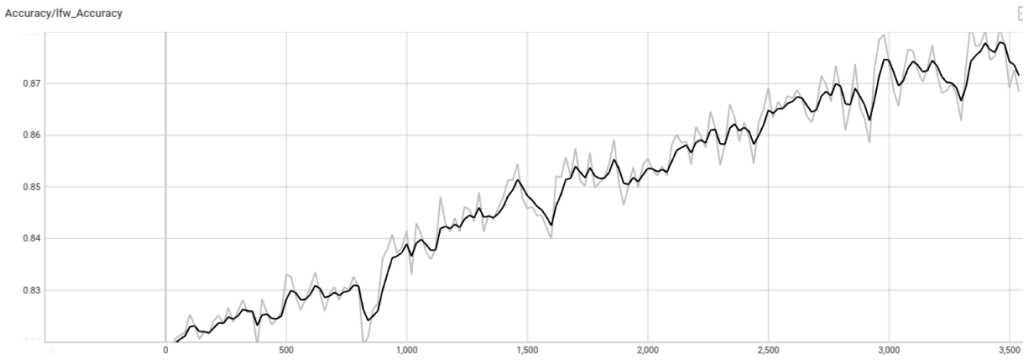

Figure 4.1: lfw accuracy

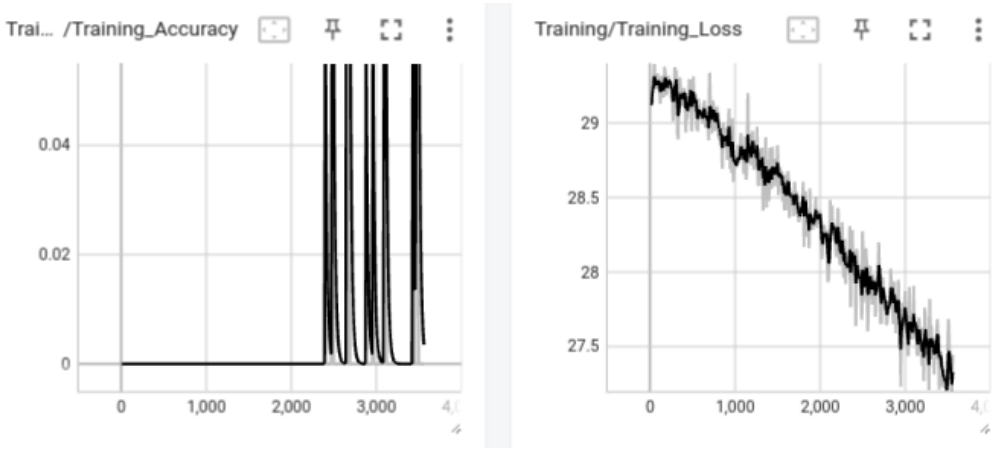

Figure 4.2: Training Accuracy & Training loss

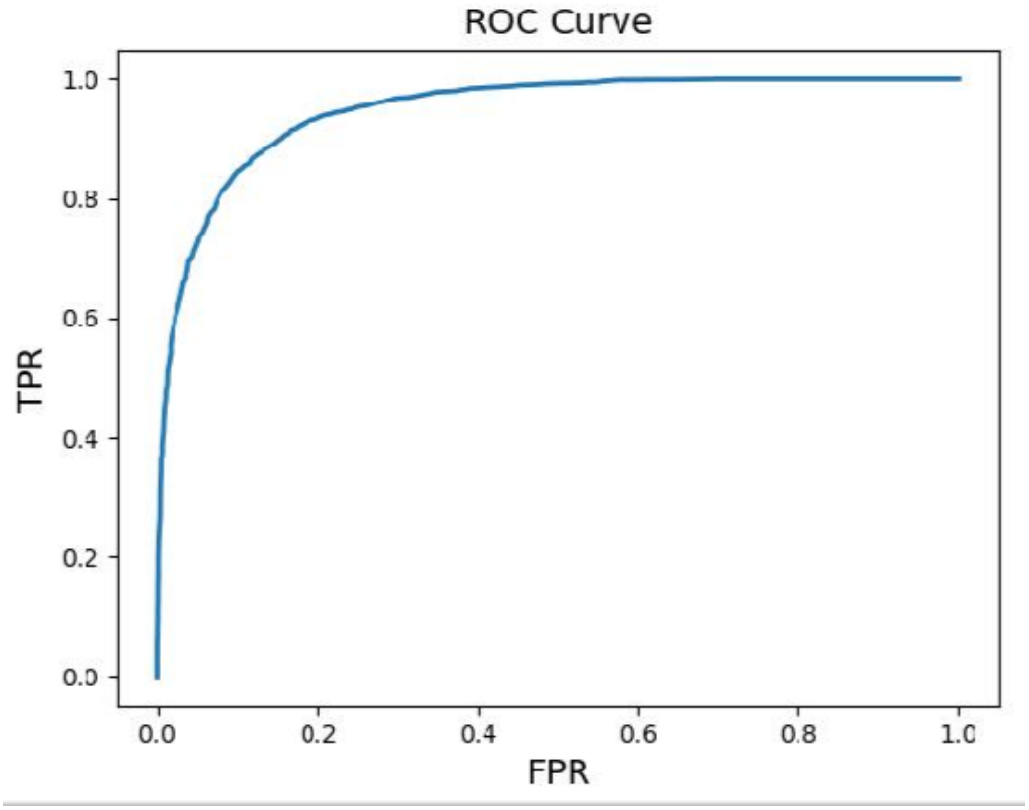

Figure 4.3: ROC-Curve

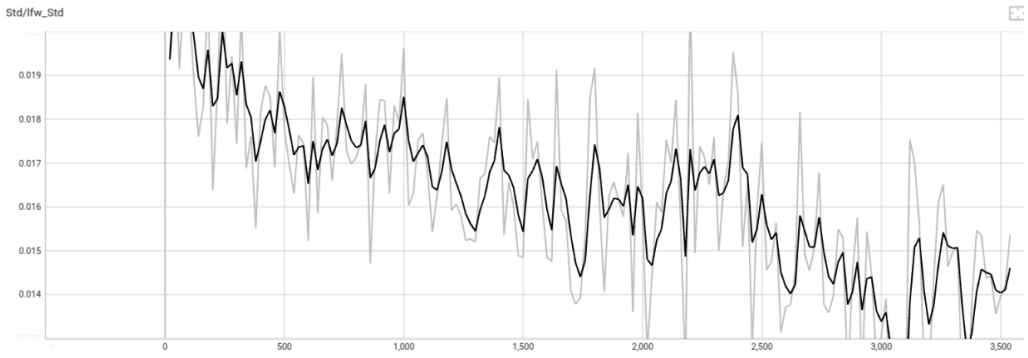

Figure 4.4: Std/lfw-std

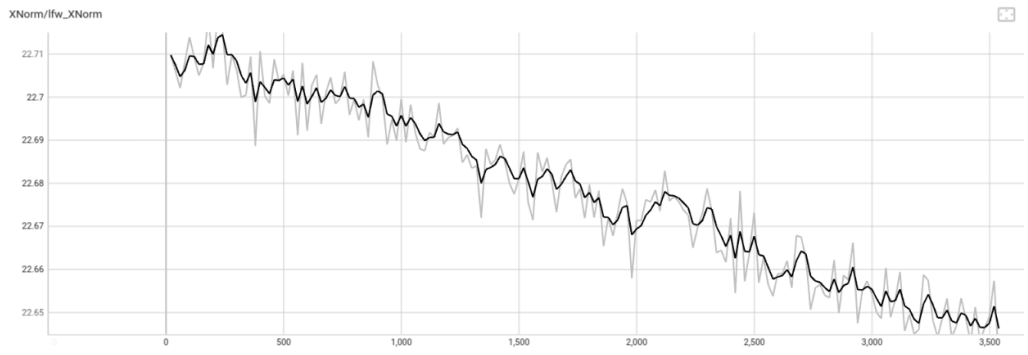

Figure 4.5: XNorm/lfw-XNorm

Chapter 5

Conclusion

In conclusion, the integration of EfficientNet with Face Transformer yielded a collaborative model that, unfortunately, fell short of the anticipated performance. Despite the individual strengths of EfficientNet and Vision Transformer, the combined accuracy reached only approximately 96.41%, a considerable decline from the standalone accuracies of both models at 92.73%. Where only ViT Model can reach accuracy upto 97.32% (ViT-P8S8) & 97.42% (for ViT-P12S8). A meticulous examination of the model's performance across various datasets revealed significant challenges.

While standalone ViT-P8S8 demonstrated commendable accuracies of 97.32% on CASIA-WebFace, substantial accuracy drops were observed on LFW (88.20%). This discrepancy indicates that the integration process faced difficulties in maintaining or enhancing accuracy across diverse datasets, suggesting potential issues in information fusion or feature compatibility.

- **Performance vs. Training Time Trade-off:**
  - ViT models (PS8 and P12S8) provide the highest accuracy but at the cost of extended training times.
  - EfficientNet alone is much faster to train but does not reach the same high accuracy levels.
  - EfficientNet combined with ViT provides a middle ground, offering a good trade-off between training time and accuracy.
- **Model Selection Based on Requirements:**
  - For applications where the highest accuracy is critical and training time is not a constraint, standalone ViT models are preferable.

- For scenarios requiring faster training times with reasonable accuracy, EfficientNet is a good choice.
- EfficientNet + ViT is suitable for applications needing a balance between accuracy and training time, offering a significant improvement in accuracy over EfficientNet with a relatively moderate increase in training time.

In conclusion, the successful development of the EfficientViT model marks a significant achievement in the field of computer vision. By merging the strengths of EfficientNet and Vision Transformer (ViT), the collaborative effort has surpassed individual model capabilities, delivering remarkable results and setting a high standard for future advancements. Overcoming hardware limitations and demonstrating exceptional accuracy on LFW [3] Evaluation, the project stands as a testament to dedication, ingenuity, and the potential for innovation through collaboration. As the team reflects on their journey, they can take pride in their accomplishments and look forward to further pushing the boundaries of computer vision research and development.

With the completion of our project, the success of our EfficientViT model stands as a testament to our dedication. Having merged the strengths of EfficientNet and ViT, we've achieved remarkable results, paving the way for future advancements in computer vision. As we reflect on our journey, we're proud to have surpassed the capabilities of our individual models and delivered performance beyond expectations.

Appendix A

Appendix

A.1 Hybrid Model Architecture

A "Hybrid Model Architecture" refers to a combination of two or more different models or approaches to solve a particular problem or address specific requirements. This blending of models allows for leveraging the strengths of each component, often aiming to compensate for the weaknesses inherent in individual models. Hybrid models are commonly used in various fields, including machine learning, computer science, and engineering.Provides a detailed diagram and description of the hybrid architecture, detailing the connection points between EfficientNet and ViT modules, and how information flows through the network.

A.2 Training Hyperparameters

Training hyperparameters are critical settings that influence the learning process of machine learning models. They are external configurations defined before the training phase and play a crucial role in determining model performance. Key hyperparameters include the learning rate, which controls optimization step size; the number of epochs, representing complete passes through the training data; batch size, specifying the number of examples used in each iteration; the optimizer algorithm, such as stochastic gradient descent; weight initialization methods; dropout rate for regularization; activation functions introducing non-linearity; regularization parameters (L1 and L2); early stopping to prevent overfitting; and data augmentation for tasks like computer vision. Tuning these hyperparameters is an essential part of model development, impacting model convergence, generalization, and overall effectiveness. The selection process involves experimentation

and iterative adjustments to achieve optimal performance based on the dataset, model complexity, and available computational resources.

A.3 LFW

"LFW" [3] commonly refers to the "Labeled Faces in the Wild" dataset, a significant resource in the field of computer vision. This dataset comprises over 13,000 labeled images of faces collected from diverse, unconstrained environments. The images exhibit variations in facial expressions, lighting conditions, and backgrounds, mirroring real-world scenarios. LFW serves as a benchmark for evaluating the performance of face recognition algorithms and models. Researchers and developers use it to assess the robustness and accuracy of face recognition systems, especially in the face of challenges posed by the dataset's diverse and uncontrolled conditions.

A.4 Transfer Learning

Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks.

A.5 Fine-Tuning

In deep learning, fine-tuning is an approach to transfer learning in which the weights of a pre-trained model are trained on new data. Fine-tuning can be done on the entire neural network, or on only a subset of its layers, in which case the layers that are not being fine-tuned are "frozen" (not updated during the backpropagation step). A model may also be augmented with "adapters" that consist of far fewer parameters than the original model, and fine-tuned in a parameter–efficient way by tuning the weights of the adapters and leaving the rest of the model's weights frozen.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762

[2] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[3] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[4] Z. Liu, Y. Lin, Y. Cao, H. H. Y. Wei, Z. Zhang, S. Lin, B. Guo, and et al., "Swin transformer: Hierarchical vision transformer using shifted windows," 2020. [Online]. Available: https://arxiv.org/pdf/1905.11946

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words - transformers for image recognition at scale," 2020. [Online]. Available: https://arxiv.org/pdf/2010.11929

[6] W. Deng, J. Hu, N. Zhang, B. Chen, and J. Guo, "Fine-grained face verification: Fglfw database, baselines, and human-dcmn partnership," *Pattern Recognition*, vol. 66, pp. 63–73, 2017.

[7] N. Zhang and W. Deng, "Fine-grained lfw database," in *International Conference on Biometrics*, 2016, pp. 1–6.

[8] T. Zheng, W. Deng, and J. Hu, "Cross-age LFW: A database for studying cross-age face recognition in unconstrained environments," *CoRR*, vol. abs/1708.08197, 2017. [Online]. Available: http://arxiv.org/abs/1708.08197

[9] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments," Beijing University of Posts and Telecommunications, Tech. Rep. 18-01, February 2018.

[10] Y. Zhong and W. Deng, "Towards transferable adversarial attack against deep face recognition," *IEEE Transactions on Information Forensics and Security*, 2020.

[11] S. Sengupta, J. Cheng, C. Castillo, V. Patel, R. Chellappa, and D. Jacobs, "Frontal to profile face verification in the wild," in *IEEE Conference on Applications of Computer Vision*, February 2016.

[12] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: the first manually collected, in-the-wild age database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, vol. 2, no. 3, 2017, p. 5.

[13] P. Wang, "Vit-pytorch," 2022. [Online]. Available: https://github.com/lucidrains/vit-pytorch

[14] Y. Zhong and W. Deng, "Face transformer for recognition," 2021. [Online]. Available: https://arxiv.org/pdf/2103.14803v2

[15] NSIT, "Face challenges," 2015. [Online]. Available: https://www.nist.gov/programs-projects/face-challenges

[16] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," 2016. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/08/MSCeleb-1M-a.pdf

[17] J. Zhao, "Face-evolve," 2019. [Online]. Available: https://github.com/ZhaoJ9014/face.evoLVe

[18] G. Reasearch, "Vision transformer," 2022. [Online]. Available: https://github.com/google-research/vision-transformer

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2015. [Online]. Available: https://arxiv.org/pdf/1409.0575

[20] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," 2021. [Online]. Available: https://arxiv.org/pdf/2104.10972

[21] A. Arora and H. Bukhari, "Vision transformer: An image is worth 16x16 words - transformers for image recognition at scale," Aman Arora, Tech. Rep., 2021. [Online]. Available: https://amaarora.github.io/posts/2021-01-18-ViT.html

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/pdf/1810.04805

[23] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[24] A. Hassani, S. Walton, J. Li, S. Li, and H. Shi, "Neighborhood attention transformer(nat): Sliding-window attention mechanism for vision," 2021. [Online]. Available: https://arxiv.org/pdf/2204.07143

[25] L. Melas-Kyriazi, "Efficientnet-pytorch," 2020. [Online]. Available: https://github.com/lukemelas/EfficientNet-PyTorch

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[27] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," 2019.

[28] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2017.

[29] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.

[31] M. M. Gharasuie, "Accessing modifying different layers of a pretrained model in pytorch," 2022. [Online]. Available: https://github.com/mortezamg63/Accessing-and-modifying-different-layers-of-a-pretrained-model-in-pytorch

[32] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.