A TRANSFORM BASED MULTIPLIER FREE DEEP NEURAL NETWORK FOR AUTOENCODER ARCHITECTURES

Doğukan GÖZLER - Beyazıt ISIK
Graduation Project
Department of Electrical and Electronics Engineering
June 2022

TC.
ESKİŞEHİR TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

18149717154 nolu **Doğukan GÖZLER** ve **23293664308** nolu **Beyazıt ISIK** tarafından hazırlanan "**A Transform Based Multiplier Free Deep Neural Network for Autoencoder Architectures"** başlıklı **Elektrik-Elektronik Mühendisliği Uygulamaları Dersi Projesi, tarafımızdan** kabul edilmiş ve proje savunması başarılı bulunmuştur.

20.06.2022

	Adı-Soyadı	İmza
Üye (Tez Danışmanı)	: Prof. Dr. ÖMER NEZİH GEREK	....
Üye	: Doç. Dr. TANSU FİLİK	....

ABSTRACT

Graduation Thesis

A TRANSFORM BASED MULTIPLIER FREE DEEP NEURAL NETWORK FOR AUTOENCODER ARCHITECTURES

Doğukan GÖZLER - Beyazıt ISIK

Eskişehir Technical University
Engineering Faculty
Electrical and Electronics Engineering

Supervisor: Prof. Dr. ÖMER NEZİH GEREK

2022, 28 pages

In this thesis, we propose “A Transform Based Multiplier Free Deep Neural Network for Autoencoder Architectures”. In today's world, the importance of deep learning and computer vision technology is increasing. These technologies, which can be used in many areas such as healthcare, automotive and agriculture etc., make great contributions making people's lives easier, accelerating, and automating production processes. While these latest technological approaches bring a solution to the aforementioned areas, they also bring a problem. This problem is the expensive hardware requirement that comes with the increased computational complexity in the deep learning model being trained. The high number of parameters and the cost of the multiplication operations used are the foremost reasons why the desired results cannot be obtained efficiently. This situation prevents the computer vision technologies to be used in systems with relatively low computing capacity such as embedded systems. Transformation-based operations were used to achieve this acceleration process. One of these operations, the Walsh Hadamard transform, is used in this study to replace 1x1 convolutions. In this way, the computation performance is increased by reducing the number of parameters. The proposed acceleration approach can be used in Mobilenet and other similar deep neural network architectures that detect objects.

Keywords: CNN, MobilenetV2, Walsh-Hadamard Transform, Artificial Intelligence

ÖZET

Lisans Tezi

**OTAMATİK KODLAYICI MİMARİLERİ İÇİN
DÖNÜŞÜM TABANLI
ÇARPAMASIZ DERİN ÖĞRENME AĞI**

Doğukan GÖZLER - Beyazıt ISIK

**Eskişehir Teknik Üniversitesi
Mühendislik Fakültesi
Elektrik-Elektronik Mühendisliği**

Danışman: Prof. Dr. ÖMER NEZİH GEREK

2022, 28 pages

Bu tezde, “Otomatik Kodlayıcı Mimarileri için Dönüşüm Tabanlı Çarpansız Bir Derin Sinir Ağı” öneriyoruz. Günümüz dünyasında derin öğrenme ve bilgisayarla görme teknolojisinin önemi giderek artmaktadır. Sağlık, otomotiv, tarım gibi birçok alanda kullanılabilen bu teknolojiler, insanların hayatlarını kolaylaştırma, hızlandırma ve üretim süreçlerini otomatikleştirme konusunda büyük katkılar sağlamaktadır. Bu son teknolojik yaklaşımlar, bahsedilen alanlara bir çözüm getirirken, aynı zamanda bir sorunu da beraberinde getirmektedir. Bu sorun, eğitilen derin öğrenme modelindeki artan hesaplama karmaşıklığı ile birlikte gelen maliyetli donanım gereksinimidir. Parametre sayısının fazla olması ve kullanılan çarpma işlemlerinin maliyeti istenilen sonuçların verimli bir şekilde alınamamasının en önemli sebepleridir. Bu durum bilgisayarlı görü teknolojilerinin gömülü sistemler gibi nispeten düşük hesaplama kapasitesine sahip sistemlerde kullanılmasını engellemektedir. Bu hızlandırma sürecini gerçekleştirmek için dönüşüm tabanlı işlemler kullanılmıştır. Bu işlemlerden biri olan Walsh Hadamard dönüşümü, bu çalışmada 1x1 evrişimlerinin yerini almak için kullanılmıştır. Bu sayede parametre sayısı azaltılarak hesaplama performansı arttırılmaktadır. Önerilen hızlandırma yaklaşımı, Mobilenet ve nesneleri algılayan diğer benzer derin sinir ağı mimarilerinde de kullanılabilir.

Anahtar Kelimeler: CNN, MobilenetV2, Walsh-Hadamard Dönüşümü, Yapay Zeka

TEŞEKKÜR

Bu çalışmanın yürütülmesi sırasında desteğini esirgemeyen danışmanımız Prof. Dr. Ömer Nezih Gerek'e en içten teşekkürlerimizi sunarız.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

With the development of deep neural networks, many sectors have started to benefit from artificial intelligence technology. Automotive, healthcare, agriculture, food, electronics and companies using smart systems are among the main sectors where this technology is used. However, convolutional neural networks, which are the specialized area of deep neural networks for computer vision, have shown an incredible development in the last 10 years and accelerated the digital transformation of the mentioned sectors. Models such as AlexNet [1], VGG-16[2] and ResNet [3], which were developed with the ImageNet competition held in 2012, showed the potential of convolutional neural networks and the development of the related technology continues rapidly today.

As the name suggests, convolutional neural networks use the convolution operation to detect and classify objects. However, in addition to the convolution operation, it also includes the activation functions, bias values and back-propagation algorithm that are already used in deep neural networks. Here, the biggest benefit of convolution operations is that the feature matrix can be extracted from the related image regardless of the position of the object on the image. In order to create these feature matrices, the matrix containing the kernel weights whose values are updated with the back-propagation algorithm is subjected to convolution on the image, and the result matrix containing the feature matrix can be obtained. By repeating these related processes continuously, it can be estimated which object is in a given picture.

Recent research shows that the performance of deep convolutional neural networks is increasing. CNN (Convolution Neural Network) structures prove its success in important areas such as classification, object detection, semantic segmentation. In order to obtain efficient results from CNN models, the model must have a deep architecture, but this causes an increase in the number of parameters and computational costs. One of the biggest challenges in deep learning networks, memory consumption is becoming a problem. Deep CNN models are insufficient for real-time applications, especially in devices with limited computational capabilities such as embedded systems and mobile phones. Although 1x1 convolution operations, one of the approaches offered as a solution to the problems we mentioned, reduce the calculation cost and the number of parameters, this approach cannot achieve the desired efficiency. In order to meet the aforementioned requirements, we propose to use the fast Walsh

Hadamard transformation. We reduced the number of parameters by replacing the 1x1 convulsion process in the last layer of the Mobilenetv2 architecture with the Walsh Hadamard Layer we created. In order to get the Walsh Hadamard, transform of a given input tensor, zero padding is applied to the obtained Walsh matrices.

2.RELATED WORK

2.1. Reduced MobilenetV2 for CIFAR10

This paper proposed by Maneesh Ayi and Mohamed El-Sharkwy in 2020. They proposed new architecture called Reduced MobilenetV2(RMNv2) for CIFAR10 dataset [12]. They obtained a network that has the total number of parameters 52.2% lesser than the vanilla MobilenetV2. Accuracy loss is about 1.9% less than the original MobilenetV2 architecture. They swapped the depthwise convolutions with the kernel-based convolutions. Bottleneck layers replaced except, the layers needed for downsampling, with the heterogenous convolution block called HetConv layer. The modified architecture is given in Figure 2.1. As a secondary method, they used Mish activation function instead of ReLU activation function. Main difference between these two activation functions is Mish can be applied on the negative values. Lastly, they implemented their own data augmentation technic called AutoAugmentation. This technic is used for searching the improved data augmentation policies [4].

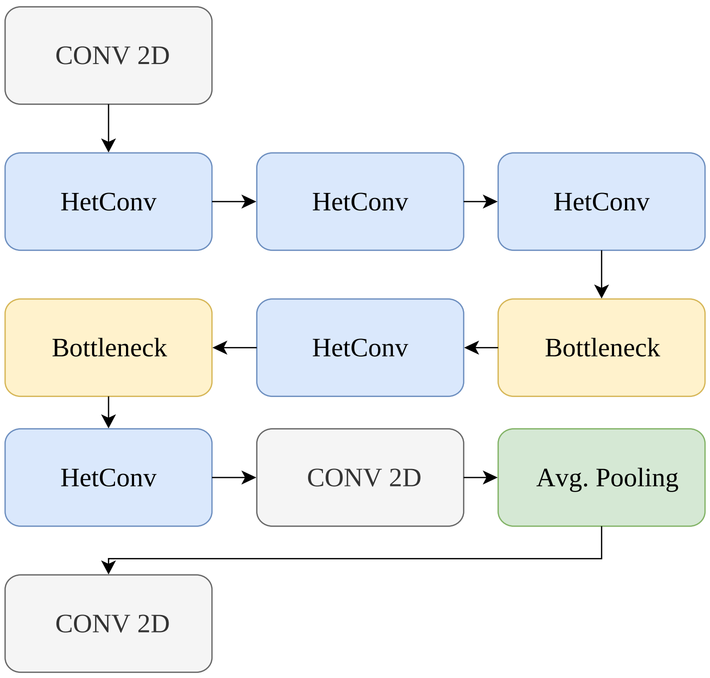

Figure 2.1: RMNv2 Architecture [4]

2.2. FNet: Mixing Tokens with Fourier Transforms

This paper proposed by James Lee-Thorp, Joshua Ainslie, and Ilya Eckstein in 2022. Transform structures is used in Natural Language Processing. They showed that Transformer encoder architectures can be speed up, with limited accuracy costs, by replacing the self-attention sublayers with simple linear transformations that "mix" input tokens. The architecture is given in Figure 2.2. Fnet is achieved the accuracy of the most accurate models while outpacing the fastest models across all sequence lengths on GPUs. As a result, they obtained a model that has light memory footprint and is particularly efficient at smaller model sizes [5].

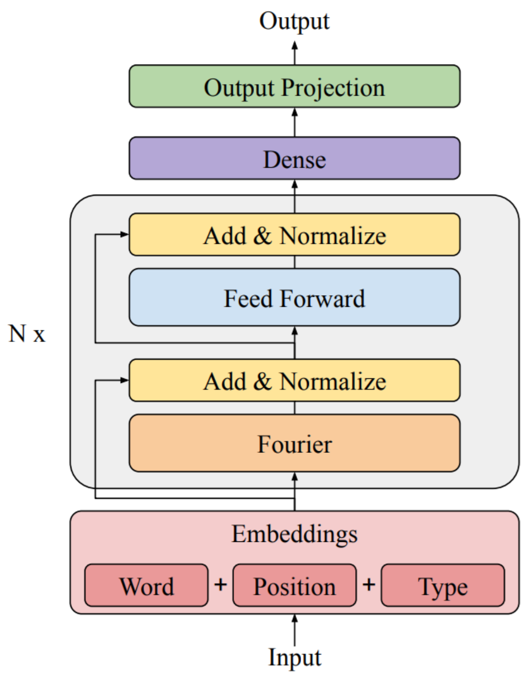

Figure 2.2: FNet Architecture with N Encoder Blocks [5]

3.THEORETICAL BACKGROUND

In this study, it is aimed to classify successfully and efficiently the CIFAR-10 dataset, which has ten different categories. To realize this process, modifications will be made on the Mobilenet-v2 architecture in a way that will contribute to this purpose. The acceleration methods currently used for running complex deep learning models on systems with relatively less computational power are “Quantization” [6, 7], “Hashing” [8], “Pruning” [9], “Vector Quantization” [10] and it includes methods such as “Huffman Encoding” [11]. In addition, simplifying the deep learning model by deleting some trained layers in the deep learning network is one of the simplification methods used [12]. As an alternative, binary neural networks can be used to reduce the number of parameters and speed up the deep learning model [13, 14].

As a solution to this problem, we will adopt a "non-crash" transform-based approach by re-modifying the "bottleneck" layers in the Mobilenet-v2 model in a way that can perform multiplication-free computation. A solution proposal similar to this approach is given in [15, 16] articles. Since the coefficients are only +1 and -1 in the mentioned studies, the need for a multiplication operation is eliminated. However, the recognition performance of the emerging deep neural network also decreases partially. In the conclusion part, it is shared how much the transformations and coefficients to be tried will decrease the basic recognition performance of the system and how much they will increase the working speed of the integrated system.

3.1. MobilenetV2 Architecture

MobilenetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. Basically, it is tailored for mobile and resource constrained environments such as cell phones and embedded devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. In addition, the MobilenetV2 architecture uses the "Depthwise Separable Convolutions" method instead of the standard convolutional operation during feature extraction with convolutional filters from images. Thanks to this technique, feature extraction can be done with eight- or nine-times fewer parameters than the standard convolution process. In the tests performed on the ImageNet dataset using the MobileNetV2 architecture, it has been determined that it performs very close to the deep architectures with much more parameters than itself [18].

The MobilenetV2 architecture uses two types of blocks as building blocks. One of them is a block with a stride value of 1. Skip connections are used in this block, which allows a deeper network to be created. In this way, a solution was provided to the vanishing gradient problem encountered in deep networks. Another block is a block with a stride value of 2. Thanks to this block, down sampling is performed without using any pooling layer. Also, this block does not contain any skip connections. These related blocks are implemented in 3 different layers in total. In the first layer, the channel size of the input tensor is expanded using 1x1 convolution operations. In the next layer, depthwise operation is performed. The filters used in second layers are 3x3 in size.

In the last layer, linear combination is performed using the 1x1 convolution operation. MobilenetV2 uses ReLU6 activation function in all its layers. However, in this last layer, only linear operations are used, in other words, it does not contain any activation functions. Figure 3.1 shows the structures of the 2 different blocks.

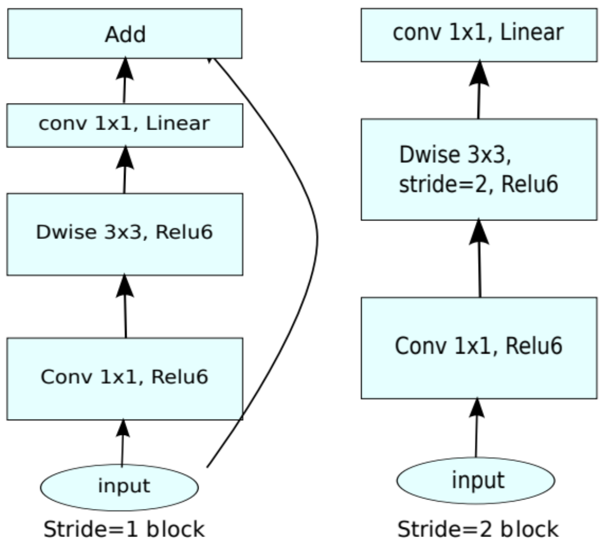

Figure 3.1: Depthwise Separable Convolution Flow Chart [18]

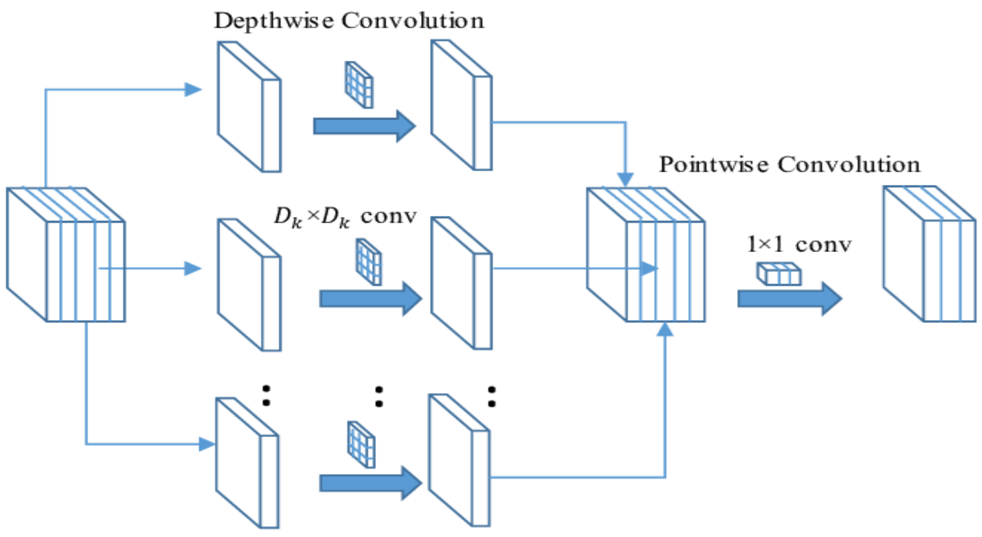

Figure 3.2: Two Different Blocks Used in MobilenetV2 Architecture

In MobilenetV2 architecture, depthwise separable convolution is used to reduce required parameter size and computation cost. While standard convolution performs the channelwise and spatial-wise computation in one step, Depthwise Separable Convolution splits the computation into two steps: depthwise convolution applies a single convolutional filter per each input channel and pointwise convolution is used to create a linear combination of the output of the depthwise convolution. In the Figure 3.2 the general structure of depthwise separable convolution can be seen. All kernel channel dimensions are 1. However, to perform this operation the input channel size and the number of used filters must be same. In this way, the channelwise convolution operation can be performed.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 3.1: MobilenetV2 architecture [18]

In MobilenetV2, D_k value is three. In addition, that, in pointwise stage there is no activation function is needed because of linear combination.

Table 3.1 shows all layers in the MobilenetV2 architecture. In the table above, the t value shows the expansion factor, the c value shows the number of output channels, and the n value shows the number of times the layer is repeated. The s value indicates the stride value. This value is the specified value in the table in the first iteration in repeating layers. However, the stride value is 1 in the remaining repetitions.

Building unit of Mobilenetv2, has several properties that make it particularly suitable for mobile applications. It allows very memory-efficient inference and relies on utilize standard operations present in all neural frameworks. In this study, PyTorch framework is used to implement this model.

3.2. Fast Walsh Hadamard Transform

WHTs are used in many different applications, such as power spectrum analysis, filtering, processing speech and medical signals, multiplexing and coding in communications, characterizing non-linear signals, solving non-linear differential equations, and logical design and analysis. The WHT is a suboptimal, non-sinusoidal, orthogonal transformation that decomposes a signal into a set of orthogonal, rectangular waveforms called Walsh functions. The transformation has no multipliers and is real because the amplitude of Walsh (or Hadamard) functions has only two values, +1 or -1.

In the Figure 3.3, +1 values are shown in yellow, and -1 values are shown in purple. As can be seen from the graphs, the Walsh matrix is the frequency ordered version of the Hadamard matrix. Gray code permutation and bit-reversal permutation operations are used to reduce the frequency sequence operation from $O(n^2)$ time complexity to O(nlogn) time complexity. In order to perform the WTS operation, the Walsh matrix must be subtracted. The dimension of the Walsh matrix is always 2^n $n \in N$. These expressions are used to describe the orthogonal Walsh matrix $W_k \in R^{(m \text{ x } m).}$ Therefore, the size of the image or attribute matrix to be transformed should be padded with 0 in the same way.

The fast Walsh-Hadamard transform is based on the butterfly process, which is the basis of the fast Fourier transform. The basic building block of the HW transformation is the following binary Hadamard matrix [15].

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.2.1}$$

The 2x2 WH transformation of vector **Y** can be explained for $\mathbf{X} \in \mathbb{R}^2$ as follows.

$$\mathbf{Y} = \mathbf{WX} = \mathbf{HX} \tag{3.2.2}$$

In general, the WH transformation of the **Y** vector can be described as $\mathbf{Y} = \mathbf{W}_k \mathbf{X}$, $\mathbf{X} \in \mathbb{R}^m$. Here m = 2k, k ∈ ℕ. These expressions are used to describe the orthogonal Walsh matrix $\mathbf{W}_k \in \mathbb{R}^{m\,x\,m}$. To find the Walsh matrix, the corresponding Hadamard matrix can be frequency sorted.

➢ First, the Hadamard matrix is created **H**k:

$$\mathbf{H}_k = \begin{cases} \mathbf{1}, & k = 0, \\ \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{H}_{k-1} \\ \mathbf{H}_{k-1} & -\mathbf{H}_{k-1} \end{bmatrix}, & k > 0, \end{cases} \tag{3.2.3}$$

Alternatively, the Kronecker product can be used to calculate the Hadamard matrix more easily. The definition of this process is given as follows. This sign "⊗" stands for the Kronecker product.

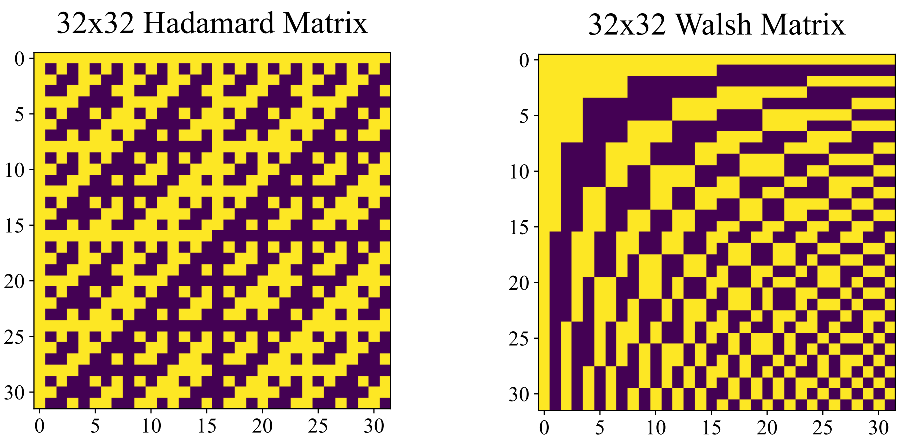

Figure 3.3: Graphical Representation of Hadamard and Walsh Matrices

$$\mathbf{H}_k = \frac{1}{\sqrt{2}} \mathbf{H}_1 \otimes \mathbf{H}_{k-1} \tag{3.2.4}$$

The process for finding a sample Walsh matrix is given below.

$$\mathbf{H}_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{3.2.5}$$

➢ To obtain the Walsh matrix from the Hadamard matrix above, frequency ordering is applied along the row or column.

$$\mathbf{W}_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{3.2.6}$$

Leaving aside the coefficient that scales the entire matrix, the time complexity of the fast Walsh Hadamard transform can be expressed as $O(m \log_2 m)$. FWHT As mentioned in Eq. (3.2.1), it is completely butterfly based. In this way, only addition and subtraction operations can be used while getting the transformation. If the $1/\sqrt{2}$ coefficients are not used during the transformation, the inverse WH transform can be represented as $\mathbf{X} = \frac{1}{m}\ \mathbf{W}_k \mathbf{Y}$, in which case the coefficient m should be the normalization coefficient.

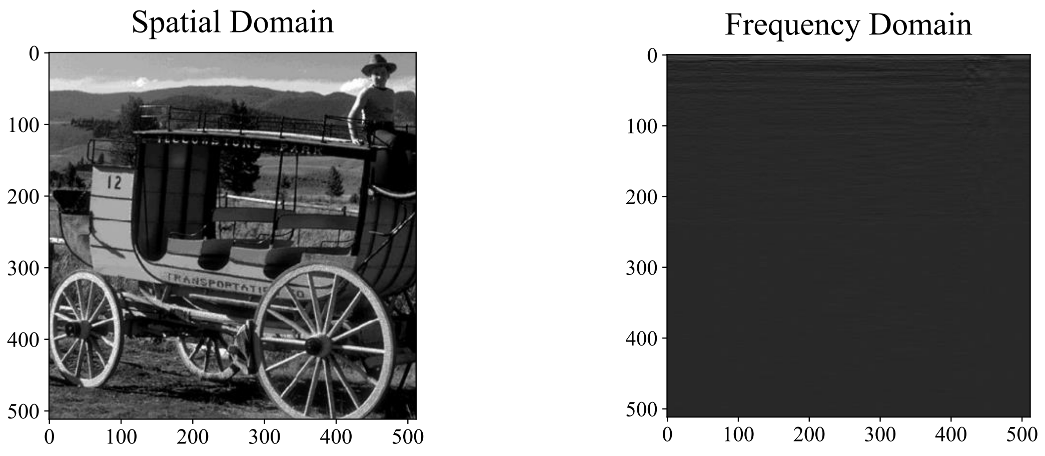

Figure 3.4: Result of Forward WHT of a Sample Image

Algorithm 1 Walsh Matrix Generation

```
function GetGrayCode($n$)
    $gray \leftarrow emptyArray()$

    for i in 0 to n do
        $gray.add(i \; xor \; (i >> 1))$
    end for

    return $gray$
end function

function GetBitReversal($gray$)
    $reverse \leftarrow emptyArray()$

    for i in 0 to gray.length do
        $reverse[i] \leftarrow bin(gray[i])[:: -1]$
    end for

    return $reverse$
end function

function GetHadamardMatrix($k$)
    if $k < 1$ then
        return $Array(1)$
    end if

    $r \leftarrow onesArray(2^k, 2^k)$
    $r[2^{k-1}:, 2^{k-1}:] *= -1$

    $p \leftarrow GetHadamardMatrix(k-1)$

    $h \leftarrow emptyArray(2^k, 2^k)$

    $h[0:2^{k-1}, 0:2^{k-1}] \leftarrow r[0:2^{k-1}, 0:2^{k-1}] * p$
    $h[2^{k-1}:, 0:2^{k-1}] \leftarrow r[2^{k-1}:, 0:2^{k-1}] * p$
    $h[0:2^{k-1}, 2^{k-1}:] \leftarrow r[0:2^{k-1}, 2^{k-1}:] * p$
    $h[2^{k-1}:, 2^{k-1}:] \leftarrow r[2^{k-1}:, 2^{k-1}:] * p$

    return $h$
end function
```

Figure 3.5: Pseudocode for the Walsh Matrix Generation

In the algorithms above, pseudo-codes on how to construct Hadamard matrix, gray code permutation and bit-reversal permutation are shared.

3.3. Fast Walsh Hadamard Layer

As it is known, 1x1 convolutions have as many parameters as the channel size of the filter. This number of parameters is very time consuming both during back propagation and forward propagation. Thanks to the FWHT layer [15] some of the 1x1 convolutions used in MobilenetV2 can be replaced with this transformation-based layer. Walsh-Hadamard transform is applied to the input tensor in the FWHT layer. With this transformation, the tensor is passed to the Hadamard domain. Soft-Thresholding operation is applied to this tensor in the Hadamard domain as an activation function. After these operations, the inverse transform is taken and the input tensor is passed back to the feature-map domain.

In Figure 3.6, a comparison of the Soft-Thresholding and Smooth-Thresholding activation functions can be seen. The FWH transform coefficients can take both positive and negative values. Large positive and negative transform domain coefficients are equally important. Therefore, we cannot use the ReLU function in the transform domain. Soft-Thresholding operation is applied to all channels except the first channel of the transformed tensor. The first channel was excluded because it contains important information about features.

$$y = \mathrm{S}_T(x) = \mathrm{sign}(x)(|x| - T)_+ = \begin{cases} x + T, & x < -T \\ 0, & |x| \le T \\ x - T, & x > T \end{cases} \tag{3.3.1}$$

In Eq. 3.3.1, functional representation of Soft-Thresholding can be seen. Soft-thresholding (ST) is commonly used in wavelet domain denoising algorithms. T is the thresholding parameter and is trainable in FWHT layer . The denoising parameter T can be learned using the back-propagation algorithm during training

The trainable number of parameters in FWHT layers is no more than the (2d − 1) where d is the minimum integer such that 2d is no less than the number of input channels. The trainable parameters are only the threshold values of the smooth-thresholding. Therefore, it is clear that FWHT layer requires significantly fewer parameters than the regular 1 × 1 layer. In this implementation only last 1x1 convolution layer is replaced. In Figure 3.7 the general flow of the model with FWHT layer can be seen.

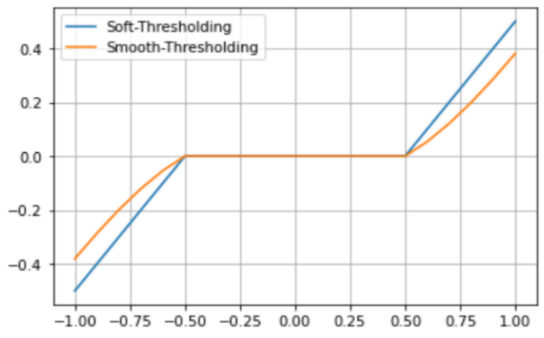

Figure 3.6: Soft-Thresholding vs Smooth- Thresholding [15]

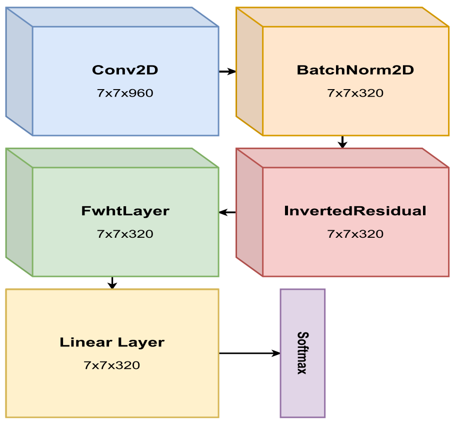

Figure 3.7: Flowchart of MobilenetV2 with FWHT Layer

3.4. Cosine Annealing Learning Rate with SGDR

Gradient descent is an iterative first-order optimization algorithm used to find a local minimum and maximum of a given function. This method is commonly used in deep learning to minimize a loss function. Gradient descent is a way to minimize an objective function J(θ) parameterized by a model's parameters $\theta \in R^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ respectively to the parameters [17].

SGDR means Stochastic Gradient Descent with Restart. The idea behind SGDR is, instead of trying to add various forms of learning rate decay, reset our learning rate every so many iterations so that we may be able to more easily pop out of a local minimum if we appear stuck. This has seemed to be quite an improvement in various situations as compared to the normal SGD using mini batches. SGDs are using cosine learning rate so that it initially starts with a relatively high learning rates for several iterations in the beginning to quickly approach a local minimum, then gradually decrease the learning rate as we get closer to the minimum, ending with several small learning rate iterations.

Cosine annealing learning rate's graph can be seen in Figure 3.8. Small restarts can be seen after some epoch intervals. With the restart asserted the learning rate is set to 0.5.

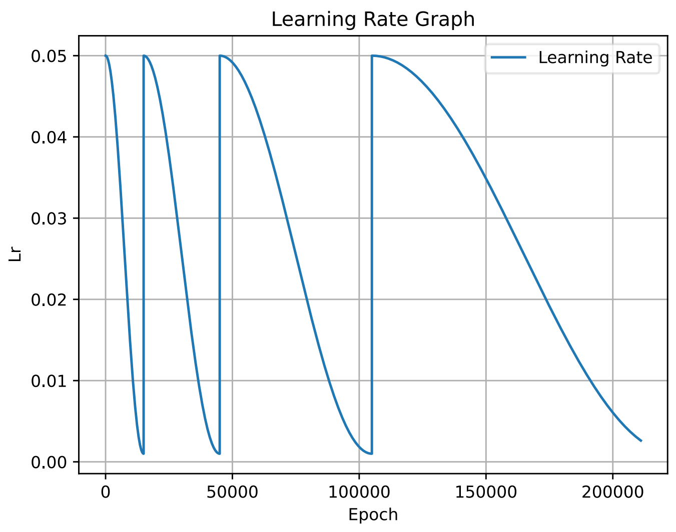

Figure 3.8: Cosine Annealing Learning Rate Graph of MobileNetv2

4.EXPERIMENTAL RESULTS

The Mobilenetv2 model and its modified version were built using Python 3.8 with the Pytorch library. The test results obtained are the training results on the CUDA supported GTX 1080 graphics card.

4.1. Dataset

CIFAR-10 is commonly used dataset for benchmarking CNN models. This dataset consist of 10 classes and each class has 6000 images. Images in the dataset are 32x32 colored RGB images. The classes in the dataset are like in Figure 4.1 which, are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Image distribution used for benchmarking the network is given in the Table 4.1 For each class we used 5000 images for training. Remaining part of the images are used for the test and validation. CIFAR-10 dataset used to compare the performance of the developed model with the vanilla MobilenetV2 model [19].

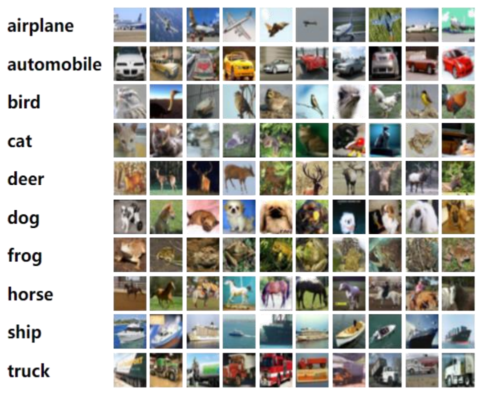

Figure 4.1: Image Samples and Classes of CIFAR10 [19]

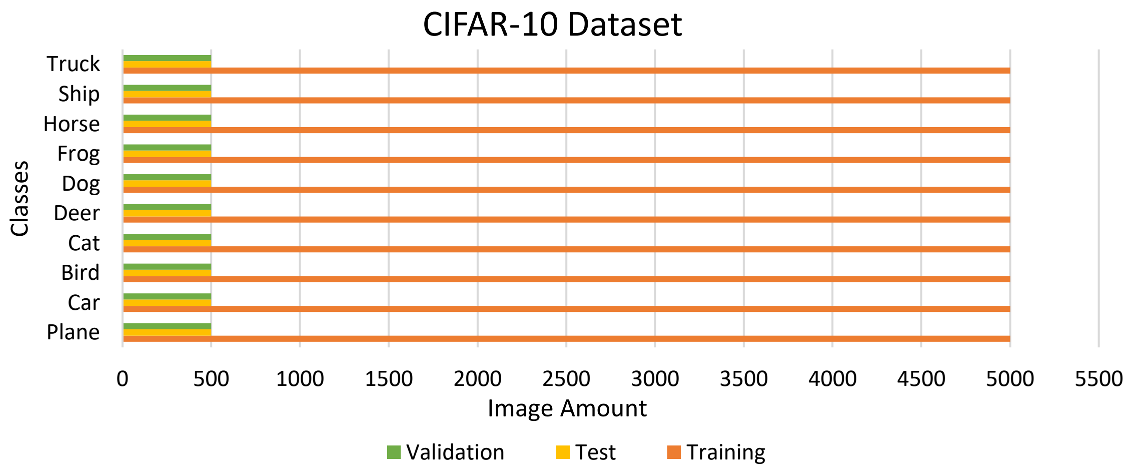

Table 4.1: CIFAR-10 Dataset Image Distribution

4.2. Test Result for Vanilla MobilenetV2

To compare the performance of the modified model in this thesis, the results of the vanilla version of the MobilenetV2 model must be obtained in the same test environment. In this context, the MobilenetV2 model, which was created using the exact version in the literature, was trained with the CIFAR-10 dataset using the cosine annealing learning rate. The accuracy, recall and precision results obtained from the test are as in Figure 4.2.

As can be seen from the graphs, it can be concluded that the model generalizes after the 80th epoch. In the loss graph in Figure 4.3, it can be understood that the vanilla MobilenetV2 model learned without loss. With these two inferences, it can be said that the model is ready and suitable for comparison.

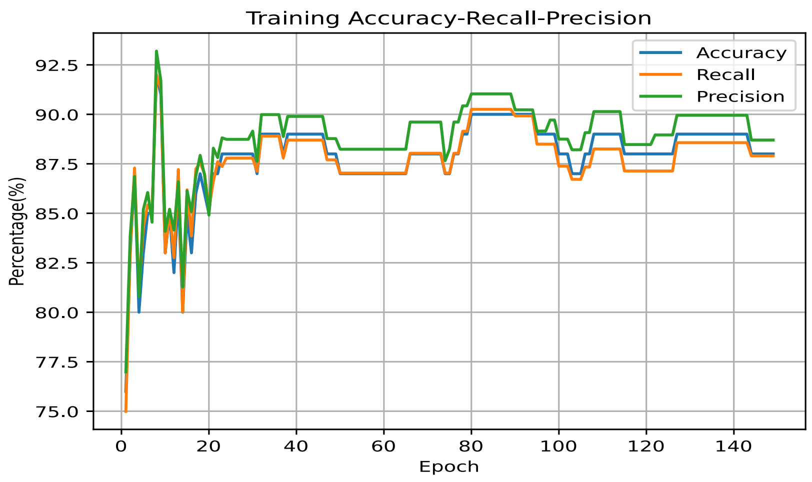

Figure 4.2: MobilenetV2 Accuracy, Recall, Precision Results

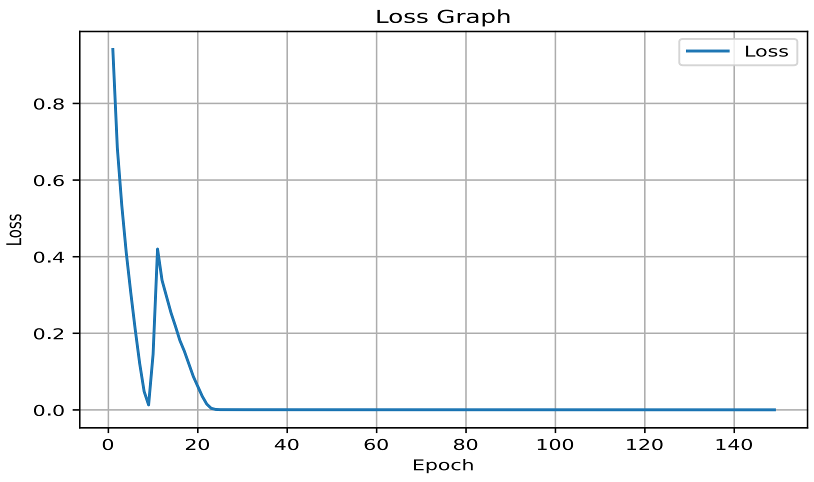

Figure 4.3: MobilenetV2 Loss Result

4.2. Test Result for Modified MobilenetV2

This test includes the result and inferences obtained by replacing the Walsh Hadamard transform layer of MobilenetV2, which is mentioned in the methodology section, instead of the last 1x1 convolution layer in the main architecture. This modified architecture was also trained with the same dataset to make a healthy comparison with the vanilla MobilenetV2 training.

Firstly, as can be seen in Table 4.2, a decrease of 18.86% was achieved in the total number of parameters. But this parameter reduction comes with a huge loss in general architecture. In Figure 4.4 the loss is not converging to 0. This means that the model is not learning.

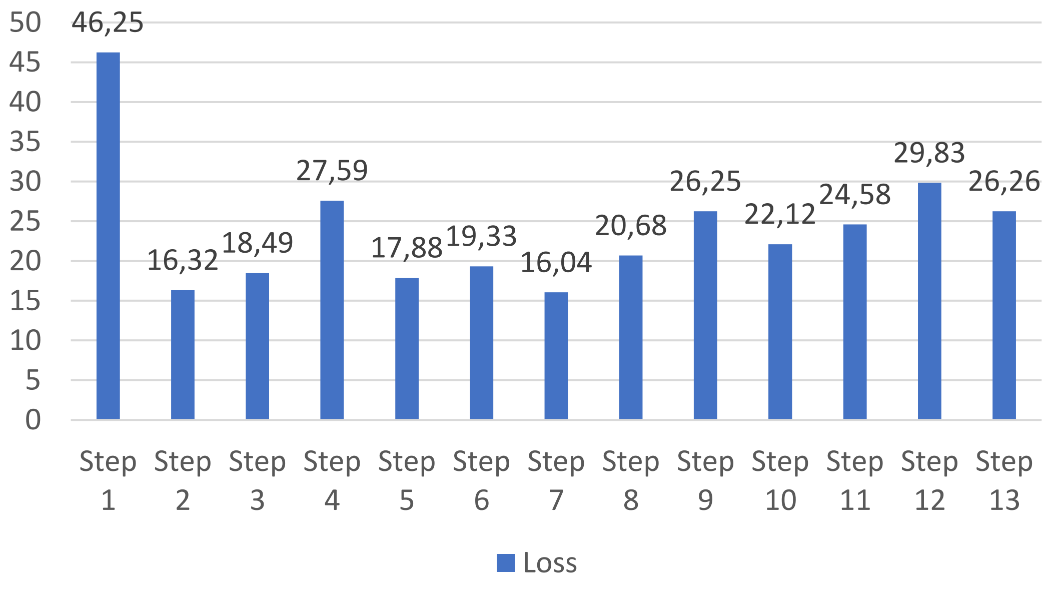

Figure 4.4: Modified MobilenetV2 Loss Result

Architecture	Parameters
Mobilenetv2	2,236,682
Mobilenetv2 +FWHT	1,814,922

Table 4.2: Modified Architecture Comparison

5.CONCLUISON

In this study, Walsh Hadamard Layer is used instead of 1x1 convolutions in MobilenetV2. In this layer, Walsh Hadamard transform, and soft threshold activation function are used. In the FWHT layer, threshold values are selected as trainable parameters. In this way, the parameters are updated because of back propagation. Also, the Walsh Hadamard transform is implemented in O(nlogn) complexity, resulting in less processing requirements compared to the 1x1 convolution. In addition to these, Cosine Annealing Learning Rate was used during the training and the global minimum point was found with warm restarts at certain intervals. After the training with the CIFAR-10 dataset, a decrease of 18.86% was observed in the number of parameters. With this decrease, when the graph of the loss function is examined, it is seen that it did not converge to 0 and there was no decrease. To conclude, it has been concluded that the Walsh-Hadamard layer is more beneficial for systems with low hardware capabilities in terms of reducing the number of parameters, but the model does not learn based on the graph of the loss function due to a reason that may arise from the position where the layer is added.

REFERENCES

[1] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding “Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S. Awwal, Vijayan K. Asari: “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches”, 2018; [http://arxiv.org/abs/1803.01164 arXiv:1803.01164].

[2] Karen Simonyan, Andrew Zisserman: “Very Deep Convolutional Networks for Large-Scale Image Recognition”, 2014; [http://arxiv.org/abs/1409.1556 arXiv:1409.1556].

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: “Deep Residual Learning for Image Recognition”, 2015; [http://arxiv.org/abs/1512.03385 arXiv:1512.03385].

[4] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le Google Brain,” AutoAugment: Learning Augmentation Strategies from Data” arXiv preprint arXiv:1805.09501v3 (2019)

[5] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with Fourier transforms. arXiv preprint arXiv:2105.03824,2021.

[6] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4820– 4828, 2016.

[7] Usama Muneeb, Erdem Koyuncu, Yasaman Keshtkarjahromd, Hulya Seferoglu, Mehmet Fatih Erden, and A Enis Cetin. Robust and computationally efficient anomaly detection using powers-of-two networks. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2992–2996. IEEE, 2020.

[8] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In International conference on machine learning, pages 2285–2294. PMLR, 2015.

[9] Hongyi Pan, Diaa Badawi, and Ahmet Enis Cetin. Computationally efficient wildfire detection method using a deep convolutional network pruned via fourier analysis. Sensors, 20(10):2891, 2020.

[10] Mingchao Yu, Zhifeng Lin, Krishna Narra, Songze Li, Youjie Li, Nam Sung Kim, Alexander Schwing, Murali Annavaram, and Salman Avestimehr. Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training. arXiv preprint arXiv:1811.03617, 2018.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arxiv 2015. arXiv preprint arXiv:1510.00149, 2019.

[12] Maneesh Ayi and Mohamed El-Sharkawy. Rmnv2: Reduced mobilenet v2 for cifar10. In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pages 0287–0292. IEEE, 2020.

[13] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.

[14] Adrian Bulat and Georgios Tzimiropoulos. Hierarchical binary cnns for landmark localization with limited resources. IEEE transactions on pattern analysis and machine intelligence, 42(2):343–356, 2018.

[15] Hongyi Pan, Diaa Badawi, Ahmet Enis Cetin. Fast Walsh-Hadamard Transform and Smooth-Thresholding Based Binary Layers in Deep Neural Networks. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), arXiv preprint arXiv:2104.07085v4, 2021.

[16] Cem Emre Akbas¸ Alican Bozkurt, A Enis Cetin, Rengul Cetin-Atalay, and Aysegul¨ Uner. Multiplication-free neural networks. In 2015 23rd Signal Processing and Communications Applications Conference (SIU), pages 2416–2418. IEEE, 2015.

[17] Ruder, S. (2017, June 15). An overview of gradient descent optimization algorithms. arXiv.org. Retrieved June 14, 2022, from https://arxiv.org/abs/1609.04747

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520, 2018

[19] CIFAR-10 and CIFAR-100 datasets. (n.d.). Retrieved June 23, 2022, from http://www.cs.toronto.edu/~kriz/cifar.html