UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TRANSFER LEARNING IN TINY DEEP LEARNING ENVIRONMENTS

Diploma Thesis

VASILEIOS LYGNOS

Supervisor: DIMITRIOS KATSAROS

Volos 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TRANSFER LEARNING IN TINY DEEP LEARNING ENVIRONMENTS

Diploma Thesis

VASILEIOS LYGNOS

Supervisor: DIMITRIOS KATSAROS

Volos 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΕΤΑΦΕΡΟΜΕΝΗ ΜΑΘΗΣΗ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΜΙΚΡΟΣΚΟΠΙΚΗΣ ΒΑΘΕΙΑΣ ΜΑΘΗΣΗΣ

Διπλωματική Εργασία

ΒΑΣΙΛΕΙΟΣ ΛΥΓΝΟΣ

Επιβλέπων: ΔΗΜΗΤΡΙΟΣ ΚΑΤΣΑΡΟΣ

Βόλος 2022

Approved by the Examination Committee:

Supervisor	**DIMITRIOS KATSAROS**
	Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**GEORGIOS THANOS**
	Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly
Member	**DIMITRIOS RAFAILIDIS**
	Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to thank my supervisor D. Katsaros for his dedicated support and guidance. He was willing to help me throughout the whole semester. I would also like to thank Evangelia Fragkou for providing advice regarding analysis and my family who supported me and helped me achieve this goal.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Diploma Thesis

TRANSFER LEARNING IN TINY DEEP LEARNING ENVIRONMENTS

VASILEIOS LYGNOS

Abstract

Significant improvements have been made in machine learning and deep learning over the last years in all their fields. This improvement is a result of the continuous development of hardware and the increasing amount of available data. However, building a deep learning model is quite demanding and costly in terms of resources. The purpose of this thesis is to propose a transfer learning method that can train a model with the use of limited data and the minimum required resources, whilst being very precise. To demonstrate that, we will use convolution neural networks for classification problems and we will see how this method performs. More specifically, we will use a pre-trained CNN model and test it on a new dataset: initially, we will train the model in different ways aiming to decide which one performs better. For instance, selecting to train just the fully connected and the last 2 convolution layers, is one of the cases that we will check in all of our tests. Thereafter, we will collect our data and compare the cases in terms of their accuracy, loss, and energy consumption. Lastly, we will evaluate the different implementation options and analyze the results.

Διπλωματική Εργασία

ΜΕΤΑΦΕΡΟΜΕΝΗ ΜΑΘΗΣΗ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΜΙΚΡΟΣΚΟΠΙΚΗΣ ΒΑΘΕΙΑΣ ΜΑΘΗΣΗΣ

ΒΑΣΙΛΕΙΟΣ ΛΥΓΝΟΣ

Περίληψη

Σημαντικές βελτιώσεις έχουν γίνει στη μηχανική και τη βαθιά μάθηση τα τελευταία χρόνια σε όλους τους τομείς τους. Αυτή η βελτίωση είναι αποτέλεσμα της συνεχούς ανάπτυξης του υλικού και του αυξανόμενου όγκου των διαθέσιμων δεδομένων. Ωστόσο, η δημιουργία ενός μοντέλου βαθιάς μάθησης είναι αρκετά απαιτητική και δαπανηρή από πλευράς πόρων. Ο σκοπός αυτής της διπλωματικής εργασίας είναι να προτείνει μια μέθοδο transfer learning που μπορεί να εκπαιδεύσει ένα μοντέλο με τη χρήση περιορισμένων δεδομένων και των ελάχιστων απαιτούμενων πόρων, χωρίς να μειωθεί η ακρίβεια των αποτελεσματων. Για να το αποδείξουμε αυτό, θα χρησιμοποιήσουμε νευρωνικά δίκτυα συνέλιξης για προβλήματα ταξινόμησης και θα δούμε πώς αποδίδει αυτή η μέθοδος. Πιο συγκεκριμένα, θα χρησιμοποιήσουμε ένα προεκπαιδευμένο μοντέλο CNN και θα το δοκιμάσουμε σε ένα νέο σύνολο δεδομένων: αρχικά, θα εκπαιδεύσουμε το μοντέλο με διαφορετικούς τρόπους, με στόχο να αποφασίσουμε ποιο αποδίδει καλύτερα. Για παράδειγμα, η επιλογή για εκπαίδευση μόνο των πλήρως συνδεδεμένων και των τελευταίων 2 επιπέδων συνέλιξης, είναι μία από τις περιπτώσεις που θα ελέγξουμε σε όλες τις δοκιμές μας. Στη συνέχεια, θα συλλέξουμε τα δεδομένα μας και θα συγκρίνουμε τις περιπτώσεις ως προς την ακρίβεια των αποτελεσματων, την απώλεια και την κατανάλωση ενέργειας. Τέλος, θα αξιολογήσουμε τις διαφορετικές επιλογές υλοποίησης και θα αναλύσουμε τα αποτελέσματα.

Table of contents

List of figures

Abbreviations

CNN	Convolution Neural Network
IoT	Internet of Things
RGB	Red-Green-Blue
NJ	Nano joule
FLOPS	Floating-point operations
ANN	Artificial neural network
RNN	Recurrent Neural Networks

Chapter 1

Introduction

With the rise of technology, the number of devices that are connected to the internet is increasing and so a lot of new data are being produced. The problem is that it is not so easy to have a server where that data can be stored and processed as it is expensive and sometimes it can be time-consuming, therefore we need a way to process data as they are being created. For instance, a farmer can have a system on his crop with devices such as cameras and sensors which help him identify quickly plant diseases, the need for water or pesticides, and so on. All these devices produce data that must be sent to a server, processed, and then used to make a decision. This procedure is not ideal because it can be slow and not keep up with the produced data, so we need to find a way to work with real-time data. Numerous examples show the significance of fast processing of real-time data, such as self-driving cars, medical devices, face recognition from security cameras, traffic cameras, and many more. So, is mandatory to find ways of using real-time data quickly. Having the ability to process the data on the device that is created is the key to solving this problem. This method though has 2 main drawbacks, energy consumption, and computational power. Machine learning is very costly in both energy and computational power so in order to process the data on the device itself we need to find a way to make a machine learning lighter without sacrificing any accuracy and also reduce the computational power. This idea is not new as a lot of research has been done on this problem and a lot of methods have been proposed to make machine learning work on small devices (tiny machine learning). Most of these methods can be combined for a good result. The simple train-load method is the most usual way machine learning is used from small edge devices. This means that first, they collect the data, they create a model for the needs of the problem, train it and then load the weights to the edge devices. This technique solves the problem of the time and storage as this is done on the devices but with this approach, models are not being updated as the new data are coming. So, it is important for the new methods that are being proposed to try to solve both the energy-computational power problem and keeping up to date the model as the data come.

1.1 Motivation

Deep learning has demanding requirements in data as they need thousands or even millions of data to make good predictions. Training them is expensive in both time and resources but the biggest problem is that the models are trained for a specific task to do. Transfer learning is the technique that has been proposed to help with these problems. This technique can help reduce time, and energy consumption and take advantage of both the previous dataset in which the model has been trained and the new one. Tiny deep learning problems are the ones that transfer learning is solving, time, energy consumption, and memory consumption. There is a lot of research done in this field combing transfer learning and tiny deep learning environments. For instance, research[1] has been done to combine federated and transfer learning in small edge devices for deep learning networks, so we can see transfer learning can be used in various ways. We wanted to utilize transfer learning to train deep learning models with few resources and without sacrificing accuracy.

1.2 Related work

Transfer learning due to its rise in fame in recent years has been used in numerous researches. This paper [2] has used deep convolutional neural networks for Computer-Aided detection with different techniques. They found out that when they trained a CNN from scratch and with transfer learning from the ImageNet dataset or Pascal dataset, the second performed much better than the first. When they took the pre-trained model, they lowered the learning rate to 10 times less than it was originally trained for the convolutional layers when it was trained for the new datasets. With this technique, they manage to keep the learning feature of the previous dataset but also slightly update them to perform better. For the fully connected layers they randomly initialize them and then freshly trained them to the new dataset with the normal learning rate. A different approach to transfer learning has followed this paper [3]. When you take a pre-trained model for a new dataset it is inevitable that you will also get features from the previous dataset that are not useful and will make your model more complicated. So, they tried to prune the less useful layers so the model can be more precise. From their results, the pruned models achieve better accuracy and F-score and also reduced FLOPS needed to complete the training. This is an incredible finding because they increase accuracy while they reduce the flops. They also did 3 separate pieces of training, just the last dense layer, the fully connected, and the whole model which all was pretty close.

1.3 Thesis statement

In this thesis, we will try to implement transfer learning in tiny deep learning environments without training all the neural network so we reduce the need for energy and memory. We will specifically try to do that on CNN networks. The method we will follow is to get the learning features we have from a dataset and test a new dataset without training all the model just some parts of it. After we get our results and decide whether this method is possible and which part of the model is it worth training most. With training just, a part of the model we make sure we maintain a high accuracy compared to training it all and reducing the energy and memory usage.

1.4 Organization of the thesis

The following thesis has been structured in 3 chapters. Chapter 2 has information about machine learning and some of its fields that I will be using and also an explanation of the datasets and the model we will be using. Chapter 3 includes the methodology we followed for our thesis project and has all the tests we did with an explanation for each one. Chapter 4 has the summary of our findings and our final thoughts.

Chapter 2

Background

2.1 Introduction

With the rise of technology, the number of small edge devices that are being used has increased exponentially and subsequently the data produced by those devices. This has created an increasing need of finding ways to manage all these data. This is where tiny machine learning comes to solve this problem.

2.1.1 What is machine learning

Machine Learning is a field of computer science developed from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from data and make predictions about it. Such algorithms work by constructing models from experimental data to make predictions based on data or to make decisions that are expressed as the result. Machine learning is applied in a series of computational tasks, where both the design and the explicit programming of the algorithms are impossible. Examples of applications are spam filters, optical character recognition (OCR), search engines, and computational vision. Machine learning is sometimes confused with data mining, where the latter focuses more on exploratory data analysis, also known as unsupervised learning. Machine learning is distributed in 3 categories depending on the way the machine is learning from data

1. Supervised learning: The computer program receives exemplary inputs as well as the desired results from a ”teacher”, and the aim is to learn a general rule in order to match

the inputs with the Results.

2. Unsupervised learning (otherwise supervised learning or unsupervised learning): Without providing any experience in the learning algorithm, it has to find the structure of the input data. Unsupervised learning can be an end itself (discovering hidden patterns in data) or means for an end (characteristic of learning).

3. Reinforcement learning: A computer program interacts with a dynamic environment in which a specific goal must be achieved (such as driving a vehicle), without a teacher explicitly telling him or her if he or she has come close to his or her goal. Another example is learning to play a game against an opponent

2.1.2 What is tiny machine learning

When we use the term tiny machine learning we define the field of machine learning technologies as algorithms, hardware, and software that can perform these technologies on small devices. A small device can be a sensor or a traffic camera which usually has limited energy and memory to use. The hard part of this field is the limited number of resources you have available to use. In the last few years, tiny machine learning has become popular due to the data these edge devices produce as most of them are connected to the internet; these devices are also called IoT devices.

2.1.3 What is deep learning

Deep learning is a subset of machine learning that uses a network of interconnected layers to simulate the behavior of a human brain. Although it's not able to match the human brain's capabilities yet, these networks can still learn from large datasets. Besides being able to make accurate predictions, these networks also have additional hidden layers that can help improve their accuracy. Forward propagation refers to the progression of calculations via the network. The visible layers of a deep neural network are the input and output layers. The deep learning model ingests the data for processing in the input layer, and the final prediction or classification is performed in the output layer.

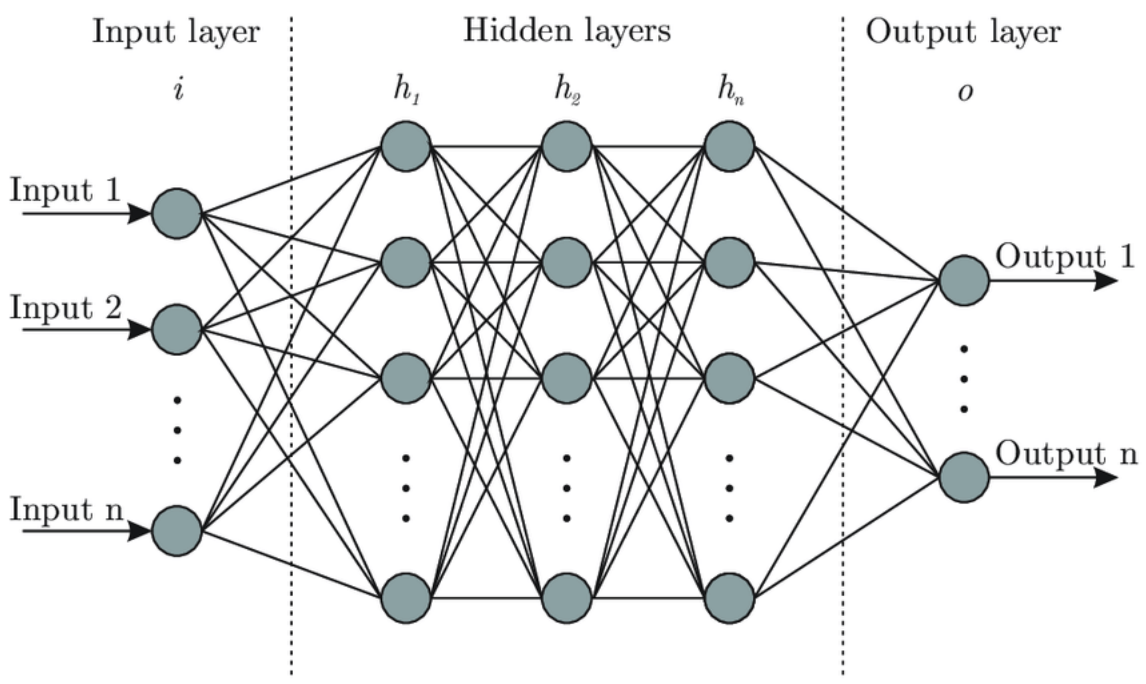


Figure 2.1: Graphical representation of a neyral network

Backpropagation is a method of training a model that uses methods such as gradient descent to calculate prediction errors and then modifies the weights and biases of the function by traveling backward through the layers. Deep learning is a major driver of artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention, and is behind many everyday products and services and emerging technologies.

2.1.4 What is convolution neural network

CNN is a special type of neural network, which is widely used for computer vision. This category of ANN is the best for image processing tasks such as object detection and classification. The architecture of CNN contains always convolutional layers, pooling layers, and fully connected layers. Usually, the first layer of a CNN is a convolution layer that has a three-dimension input, width, height, and depth. For an image, the height and width are the dimensions of it and the depth is the color, 3 if the image is RGB, and 1 if is non-RGB. A typical model has multiple convolution layers, the first layer extracts basic features from the input image such as lines. The output passes to the next layer which detects more complex features and as we move deeper into the network the features are getting more and more complex. The next layer CNN models have is the pooling layer, this layer is responsible for decreasing the size of the output a convolution layer has. Reducing the dimensions of the feature maps is useful for reducing the computational power needed. Lastly, the fully connected layers are at the end of the network and they are compiling the information extracted from the previous layers and making the classification. They are named “fully connected” because every neuron from the layer is connected to every neuron from the next layer.

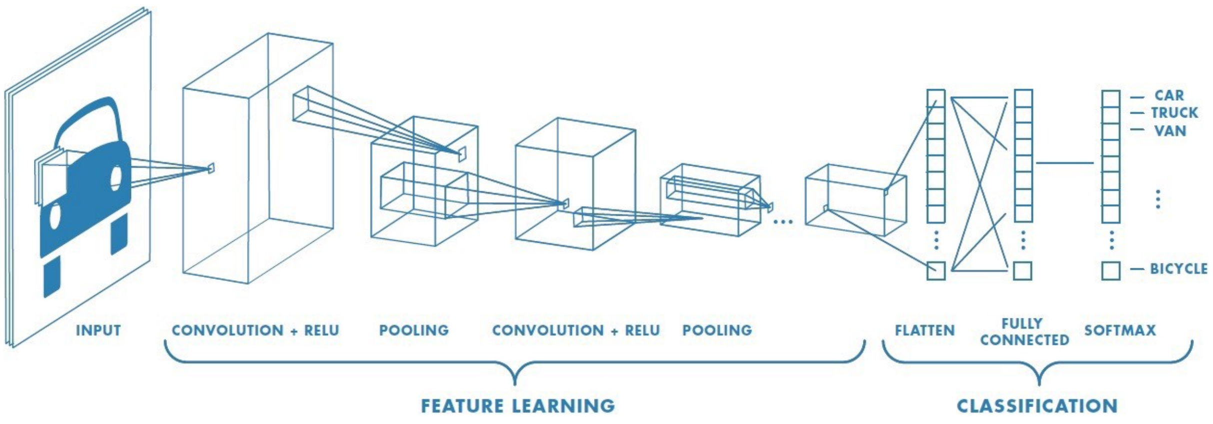


Figure 2.2: Graphical representation of a neyral network

2.1.5 What is transfer learning

Transfer learning is a method of training a model where you take a pre-trained model for a task and reuse it as a starting point for a different, but similar, task. The way the method works is, that you take the features learned on one problem and transfer them to a new similar problem. Transfer learning has a lot of benefits that can provide and these are the reasons why it is becoming so popular over the years. First of all, it saves time because when you are using the weights from another dataset the training time decreases a lot. It also can help to reduce the resources the training consumes, energy, and computational power. For example, when a large network is been trained it can take multiple days to finish even if it is trained on servers with high-end hardware. This process consumes power and produces CO2. Research [4] has shown that a network with 200 million parameters produces more CO2 than 5 average cars in their whole lifetime. Transfer learning has multiple ways it can be used, for example, you can use a pre-trained model for a new dataset just to save time or you can freeze the model and save the features and just train the fully connected layers. Transfer learning can be also combined with other methods to reduce the resources needed. In this research, we will use only transfer learning but we will test which layers are the best choice to train and which are the best to freeze.

2.2 Tools

2.2.1 Environment

The code for the tests is written in Python 3.8 and the integrated development environment (IDE) we have decided to use is Pycharm. Pycharm provides a wide range of tools for developers both for data science and web. The API for deep learning we are using is Keras[5], an open-source library for artificial neural networks and is developed by Google and written in python.

2.2.2 Datasets

The datasets we used to test our method of transfer learning were all for image classification. We wanted to have a small similarity between them so we could test how the learning features from one dataset apply to the other and how we can take advantage of that.

1. The first dataset was CIFAR-10[6]. This dataset contains small images with a size of 32x32 in 10 classes. It contains 50.000 train images and 10.000 test images. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

2. The second dataset we used was CIFAR-100[7]. This dataset is the same as CIFAR-10, except it has 100 classes and contains 600 images per class. This dataset also is divided into 20 superclasses and each of them has 5 classes, so each image comes with 2 labels, the class, and the superclass.

3. The next dataset we used was ImageNet[8]. This dataset consists of more than 14 million images and it has 1000 classes. The variety of images it contains makes it perfect for transfer learning. The average size of each image is 469x387.

4. The fourth is the Intel image classification dataset. It contains 25.000 images of size 150x150 distributed under 6 classes: buildings, forest, glacier, mountain, sea, and street.

5. MNIST[9] handwritten digit database was the fifth dataset. It is a small dataset for digit classification with 60.000 images with a size of 28x28. The classes are 10 for the digits 0 to 9.

6. The last dataset was the Street view house number dataset[10]. This dataset is similar to MNIST with the only difference the digits are not handwritten but images from house

numbers. It has 73.257 images with a size of 32x32 and comes with 10 classes for the digits 0-9.

2.2.3 Models

The models we decided to test are 3. For the purpose of this research, we want to have a variety of sizes so we used a small, a medium, and a big size model. The small model is a CNN model with an input of 32x32x1 and 550,000 total number of parameters. It has 6 convolution layers and a fully connected part with 2 dense layers.

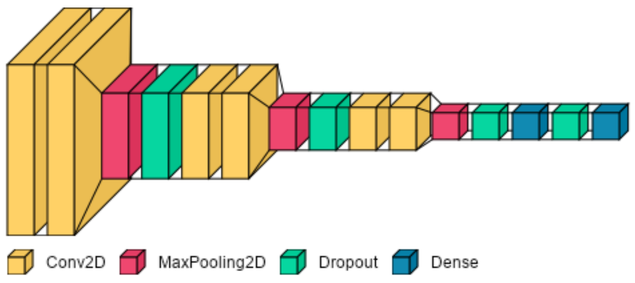


Figure 2.3: Visualisation of the small model (without Flatten and Batch Normalization layers)

The other two models are from the family of EfficientNet[11]. EfficientNet is a series of CNN models proposed by Google that compared to other architectures they provide better accuracy and efficiency (reduced flops). We picked the 2 smaller networks EfficientNetB0 and EfficientNetB2 with 4,049,571 and 7,768,569 parameters respectively. The biggest of the family, EfficientNetB7, has around 65,000,000 parameters.

2.2.4 Energy consumption

In the course of our work, we cannot review our results just from the accuracy and the loss of the model for each test. We want this method to be used for small devices which have limited energy and memory so we will also make for each test an estimation of the energy consumption. The compute cost is usually measured as the total number of floating-point operations (FLOPS) which is roughly equivalent to the number of multiply-and-accumulate (MAC) divided by two. So we have that:

$$ENERGY = \frac{FLOPS}{2} * Emac$$

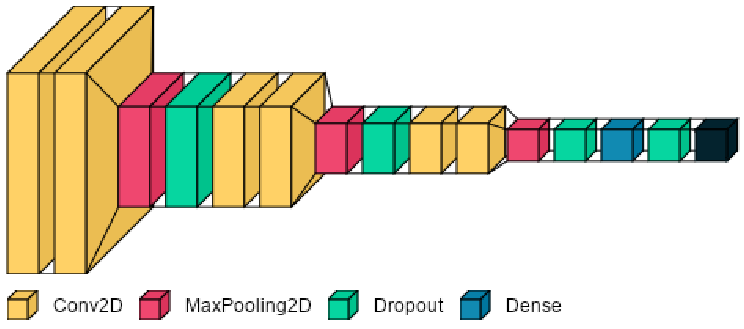
For this test, we will consider each operation is done at a 45nm CMOS technology processor which we know that every MAC operation needs 3.2 picojoule[12]. We will not take into consideration the energy needed for memory access.

Chapter 3

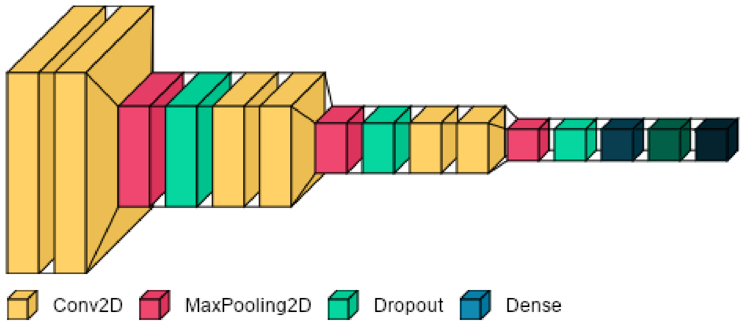
Experiment

3.1 Experimental Evaluation

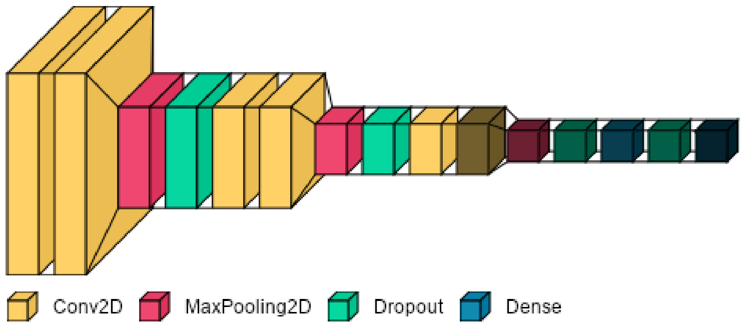
Starting with the practical part of this thesis our initial goal is to see whether is worthy to unfreeze more than the fully connected layers when you train your model and use transfer learning. In order to check if the results we have are correct, and support our method we did various tests with different datasets and models. The methodology we followed is the same for all the tests. First, we take a model which is trained to a dataset and a second dataset which has some similarity with the first, and then make the tests we want. We start with freezing all the model except the fully connected part, after training the model again we take the results and freeze the fully connected and the last convolution layer. Next, we freeze the fully connected and the last 2 convolutions and so on. To make it clear we will visualize the layers we are freezing and the layers we keep trainable for the small model.



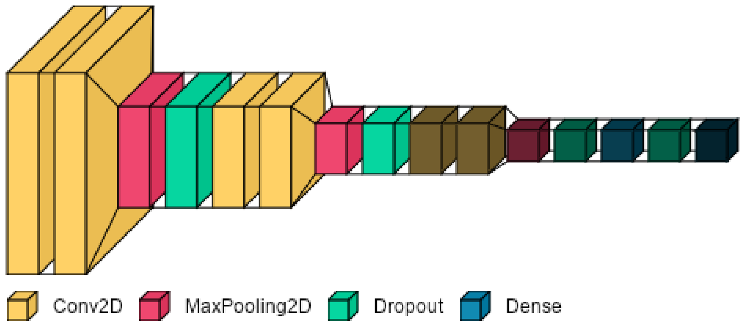
Last Dense layer (1)



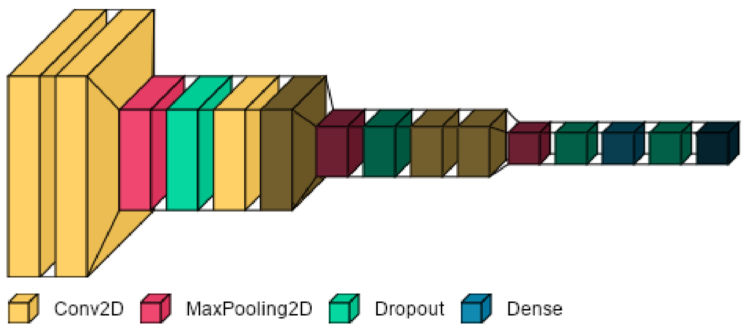
Fully connected layers (4)



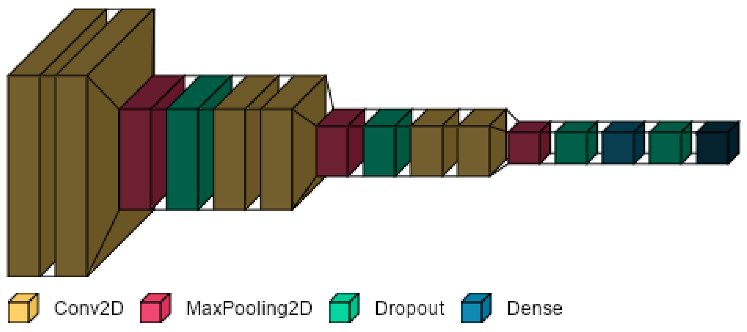
Fully connected layers and last convolution layer (9)



Fully connected layers and last 2 convolution layer (11)



Fully connected layers and last 3 convolution layer (15)



All model (all)

Figure 3.1: The 6 cases we test on the small model every time we use it. The dark layers are the ones which are trainable, the others are frozen

The numbers in the figure above represent the number of layers that are trainable and they are also used in the graphs. The same applies to all the models but we will not visualize the EfficientNet models as they are much bigger. The matching for the EfficicentNet models is as the table 3.1 shows. Lastly, after we collected all the results, we compare them based on accuracy, loss, and energy consumed.

Trainable layers	Convolution layers trained
3	1
5	2
8	4
16	5
20	6
23	8
31	10

EfficientNetB0

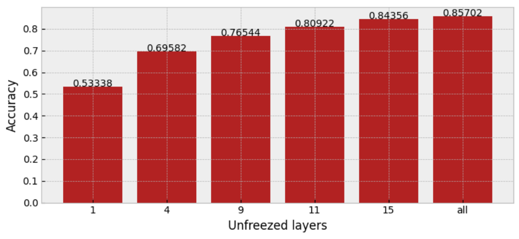
Trainable layers	Convolution layers trained
3	1
7	2
10	4
15	5
23	8
31	9
38	12

EfficientNetB2

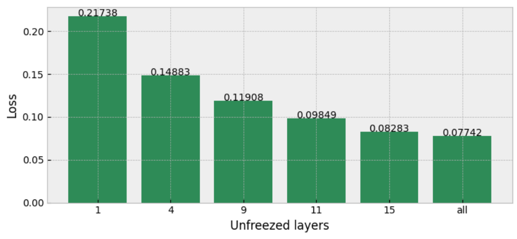
Table 3.1: The number of layers we train each time on the left side of each table and the number of convolution layers we have on the right

3.1.1 First test

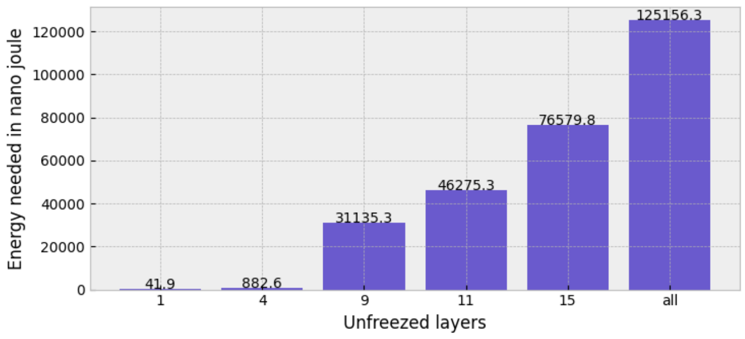
The first test we did was with the CIFAR-10 and CIFAR-100 datasets. We trained the small model with CIFAR-100 and then tested CIFAR-10 on it. From the charts 3.2, we can easily understand that the bigger the part of the model you train the better accuracy (or lower loss) you have. This is a result we were waiting to see because the last convolution layers are the ones that have filters for complex features and so by training them the model can adjust with more ease to the new dataset. The worth mentioning in the accuracy and loss charts is the big difference between the fully connected and the last convolution layer as it is the biggest improvement compared to the other bars. Looking at the energy graph we can see that training only the fully connected layers we need 850 NJ but after we start adding convolution layers to the training the consumption is increasing a lot.



Accuracy



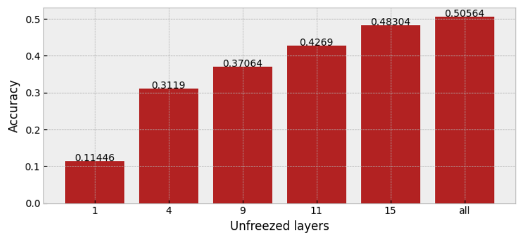
Loss



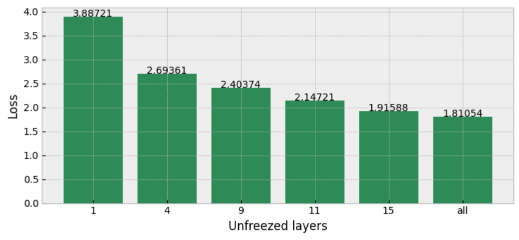
Energy consumption

Figure 3.2: The results when testing CIFAR-10 on the small model pre-trained on CIFAR-100

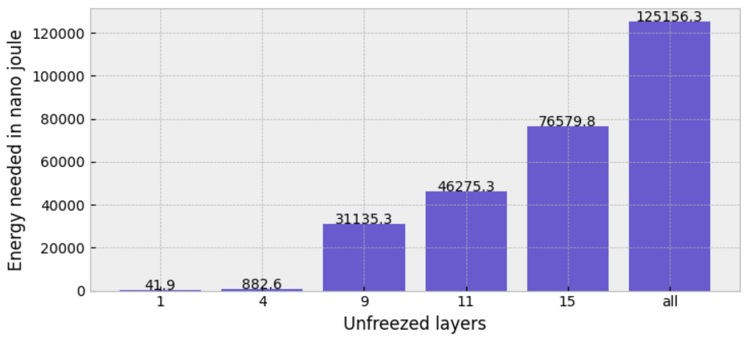
After the above test, we did the same but we trained the small model with CIFFAR-10 and tested CIFAR-100 on it. With a fast look at 3.3, we can that the accuracy is much lower than before but is a normal thing because this dataset contains 100 classes and the model is not big enough to give good results. Apart from that the findings we have here are very similar to the previous test, the biggest improvement is between fully connected and the last convolution layer, and the bigger the part of the model we train the bigger accuracy we have. Also, the energy is almost the same but a little higher because in the last dense layer we have 100 neurons compared to the 10 we had before. So from this experiment, the 2 most important things we keep are, the 13% improvement from fully connected layers to the last convolution layer and the increase in the energy we have when we start adding convolution layers to the training.



Accuracy



Loss

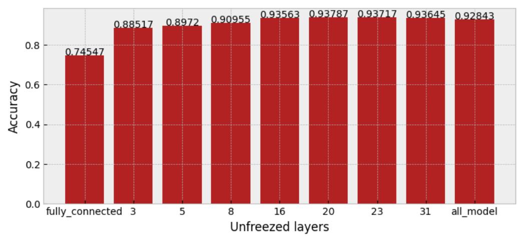


Energy consumption

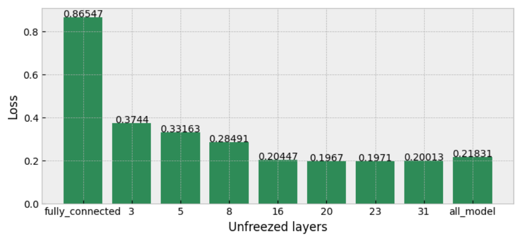
Figure 3.3: The results when testing CIFAR-100 on the small model pre-trained on CIFAR-10

3.1.2 Second test

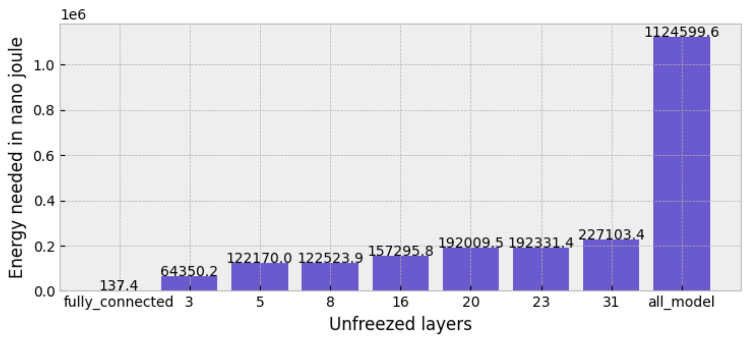
Moving to the next test we took CIFAR-100 and ImageNet datasets. The model we used for this one is EfficientNetB0 which has 4,049,571 parameters. We took the model trained for ImageNet and tested CIFAR-100 on it. Looking at the figure 3.4 we can see that accuracy is a lot higher compared to the first test we did with CIFAR-100 and this happens for two reasons. The first one is that this model is a lot better than the first one and much bigger (8 times bigger) and the second reason is that the ImageNet has 1000 classes while CIFAR has 100 so is very likely that some classes match together or need the same learning features (filters). This though is just from comparing to the other test. From the bars, we can again see that by training the fully connected layers and the last convolution layer we can get a big boost in accuracy (or much lower loss) compared to just training the fully connected layers only. Training the last two to seven convolution layers we see that we do not get a big improvement just around 4%. Taking into consideration that this model is bigger than the first model, we have bigger energy consumption and the energy for the last convolution layer and fully connected (122537 NJ) is 220 times bigger compared to just the fully connected layer (561 NJ).



Accuracy



Loss

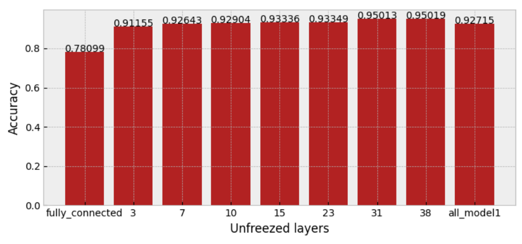


Energy consumption

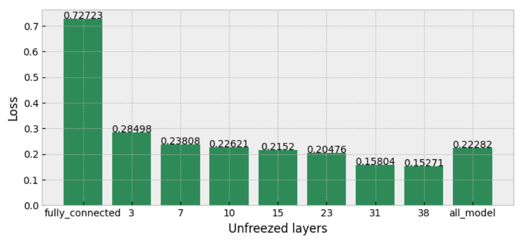
Figure 3.4: The results when testing CIFAR-100 on the EfficientNetB0 pre-trained on ImageNet

3.1.3 Third test

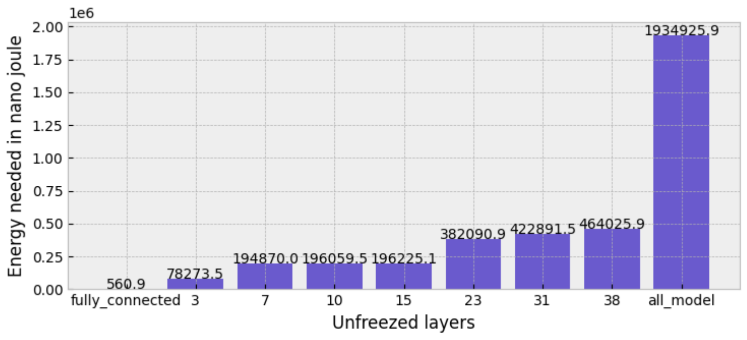
For the next test, we used again both CIFAR-100 and ImageNet but a different model. This time we used EfficientNetB2 which is a model bigger than EfficientNetB0 with 7,768,569 parameters. The findings we have here are very similar to the ones with EfficientNetB0. As it is observed from the figure 3.5 the accuracy and loss, especially in the first bars, are higher and lower respectively compared to EfficientNetB0. This only happens because of the size of the model as the dataset it was trained and the dataset was tested are the same. Again, the best improvement was between the fully connected layers and the last convolution layer with an increase of 16% in accuracy while the other steps at the bars are very small (lower than 3 %). Taking into consideration that this is a big model, the energy for the last convolution layer and fully connected (78273 NJ) is 140 times bigger compared to just the fully connected layer (561 NJ).



Accuracy



Loss



Energy consumption

Figure 3.5: The results when testing CIFAR-100 on the EfficientNetB2 pre-trained on ImageNet

3.1.4 Fourth test

Following the aforementioned tests with CIFAR datasets, we wanted to try a dataset that does not have high similarity. For this test, we will use the EfficientNetB0 model trained on ImageNet and use its learning features to see how the Intel image classification dataset performs. This dataset has 6 classes and none of them is an object only sceneries from a city, mountain, sea, glacier, and forest. Before we even see the results, we would expect a high accuracy for 2 reasons. First, this dataset contains 6 classes so it is much easier to predict the correct answer, and secondly, the learning features we want from the model are in the first convolution layers which we will freeze. When a model is training for a dataset the first layers are usually general features from the image and as you move to the next convolution layers the features are becoming more complex. For instance, an image that belongs to the class Mountain tent from ImageNet 3.6 is just a tent in front of a mountain so it will help to identify images that belong to the mountain class from the Intel dataset.

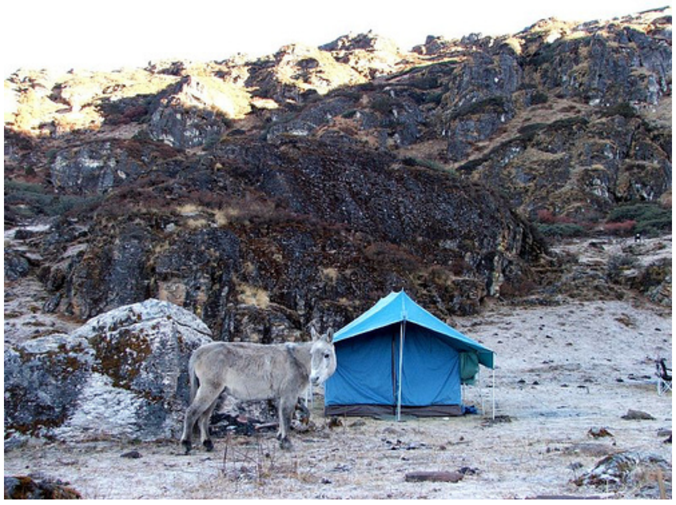
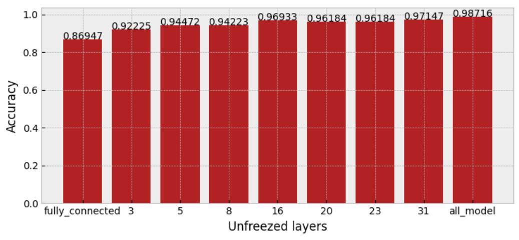
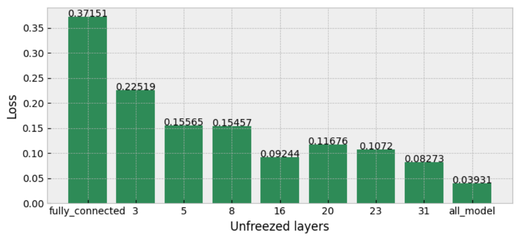


Figure 3.6: Image from ImageNet with label Mountain tent

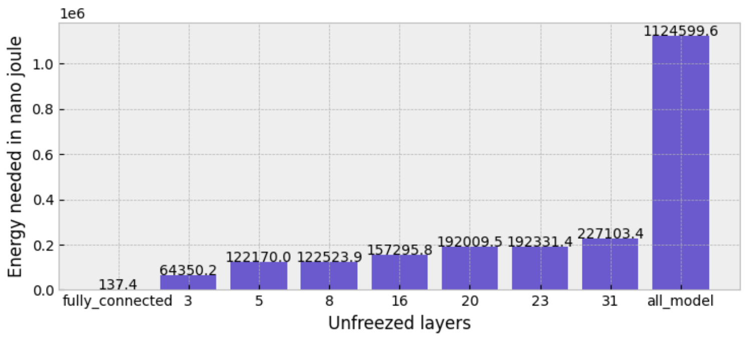
Moving to the graphs 3.7 from the tests we can notice that the accuracy of all training tests is high. Again, the best improvement we see is going from training the fully connected layers to training both fully connected layers and the last convolution layer at about a 7% increase in accuracy. The energy is almost the same for all the EfficientNet models with some differences at the fully connected part depending on the number of classes the dataset has.



Accuracy



Loss

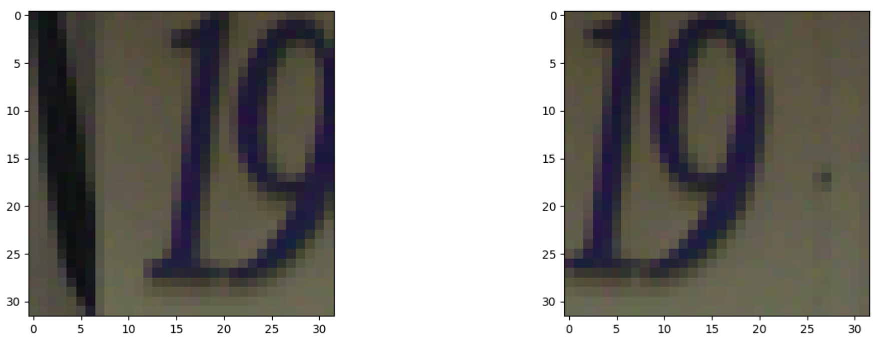


Energy consumption

Figure 3.7: The results when testing Intel clasification dataset on the EfficientNetB0 pre-trained on ImageNet

3.1.5 Fifth test

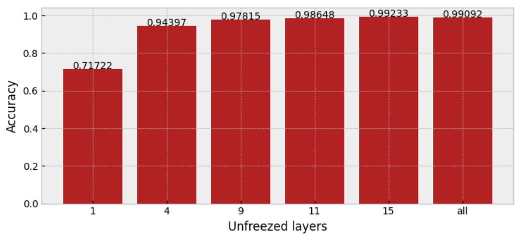
For the last test, we will use the two digits datasets with the small model. These two datasets even though they have the same classes (digits from 0 to 9) they are quite different. MNIST is quite a common dataset for beginners and easy to achieve high accuracy as the images are only the digits. On the other hand, SVHN is pictures from house numbers from Streetview, so they can be blurry or not centered and the hardest part is having more than one digit in the same image. For instance, in the following figure 3.8 are 2 different images in the dataset, with the label of the first being 1 and the second being 9. First, we will see how



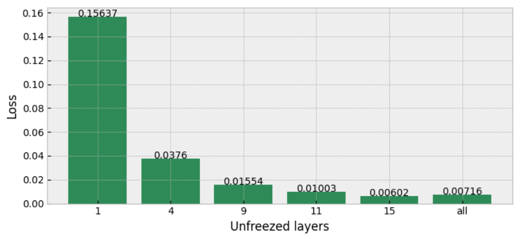
Label is 1 Label is 9

Figure 3.8: Two images from the SVHN dataset

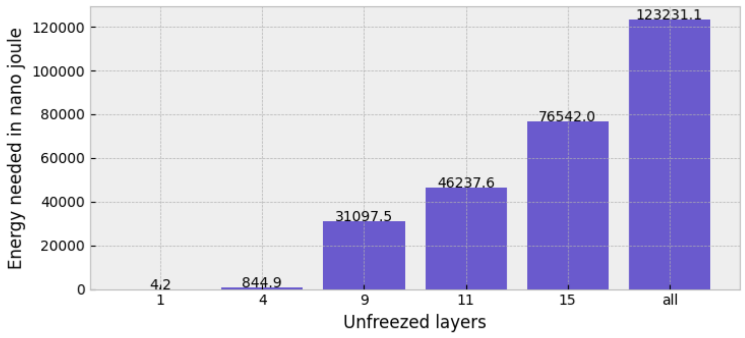
MNIST performs on the small model pre-trained with SVHN. Having said the above and looking at the graphs we can see that this test has some different patterns than the previous tests. We can see from 3.9 that the biggest improvement is between training only the last dense layer and training all the fully connected layers. We see that training and the last convolution layer does not improve our accuracy (nor reduce the loss) of the model by a lot as it is already good enough.



Accuracy

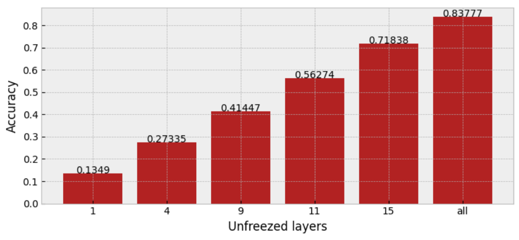


Loss

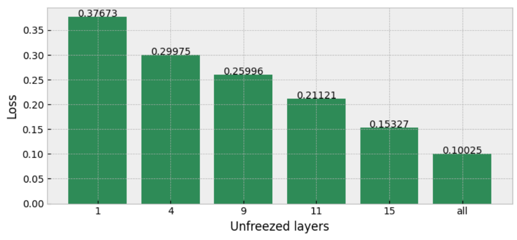


Energy consumption

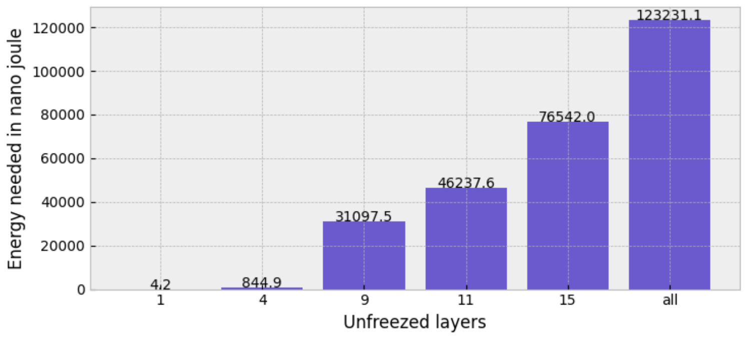
Figure 3.9: The results when testing MNIST on the small model pretrained on SVHN



Accuracy



Loss



Energy consumption

Figure 3.10: The results when testing SVHN on the small model pretrained on MNIST

If we check how SVHN performs on the pre-trained model on MNIST (figure 3.10) we see a totally different pattern. First of all, accuracy starts at a very low point training only the last dense layer at 13% while MNIST had 71%. As we start adding layers to the training part of the model, accuracy is slowly rising adding around 14% accuracy per step, and only when we train all the model we achieve 80% accuracy. Energy consumption is the same as above.

Chapter 4

Conclusions

This research aimed to assess whether we can leverage transfer learning and alternative approaches of not training the entire model of a convolution network but still achieve good accuracy and low computational cost. The outcomes of the collected results through the different approaches tested are very promising and indicate that that selective training with transfer learning of a convolution network is feasible under certain suitable conditions.

4.1 Summary and conclusions

Looking at the experiment results, we can conclude that transfer learning can be used in many different ways. As expected, the bigger and more completed the part of the model we train, the better the accuracy we get is, but by training just the fully connected layers and only the last convolution layer we achieve more than 10% of improvement for all the tests than just training the FC layers. By comparing the results we had for the CIFAR-100 model in the first (3.1.1), second (3.1.2), and third (3.1.3) tests, we realize that the architecture of the model is critical for its accuracy. The small model had an accuracy of 41% when training the fully connected part and the last convolution layer (3.3), while the EfficientNetB0 and EfficientNetB2 had 89% (3.4) and 91% (3.5) respectively. The dataset used also plays a crucial role, in terms of its size and available attributes. ImageNet is a massive dataset and a model trained on it has a lot of learning features that can be transferred to another dataset with success. On the other hand, in the last test we saw that datasets such as MNIST cannot provide good learning features to other datasets because of being too simple. SVHN did not perform well on the small model pretrained on MNIST, since training the fully connected and the last convolution layer had a 41% accuracy while training the complete model had an 83% accuracy (3.10). This is a huge gap in the performed accuracy between the two compared approaches. In addition, in our first testCIFAR-100 had a 41% accuracy with the small model by training the fully connected and the last convolution layer, and a 55% by training the complete model (3.3). This difference of a 34% improvement does not indicate that transfer learning is not effective. To conclude, transfer learning is an effective method to use under certain conditions; firstly, you need a well-designed model, and secondly, and more critically, the initial dataset the model has been trained should contain plenty of useful learning features. Taking also energy consumption into consideration our method seems more attractive and productive because whilst we train more than just the fully connected layers, we still manage to maintain a low energy consumption and still a well-accepted accuracy. To summarize, our method of applying transfer learning by training the fully connected layers and only the last convolutional layer while freezing the rest of the model is effective and with very good results. In our tests, this has been an over 10% improvement, while always remaining very close to the performed accuracy compared to the training of the complete model.

4.2 Future work

As the results of the findings being good enough and promising for implementation on an edge device the future work is split into 2 categories. Firstly, further research on this method for various neural networks such as RNN and ANN and for different problems apart from classification. Secondly, implement this method for real-world problems and devices to evaluate its performance. This method could also be combined with other techniques for real-world problems such as federated learning.

Bibliography

[1] Kavya Kopparapu and Eric Lin. Tinyfedtl: Federated transfer learning on tiny devices. *CoRR*, abs/2110.01107, 2021.

[2] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.

[3] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A transfer learning with structured filter pruning approach for improved breast cancer classification on point-of-care devices. *Computers in Biology and Medicine*, 134:104432, 2021.

[4] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *CoRR*, abs/1906.02243, 2019.

[5] Francois Chollet et al. Keras, 2015.

[6] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[7] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[10] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[11] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[12] Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14, 2020.