



CHALMERS
UNIVERSITY OF TECHNOLOGY



Impact of Training Data Volume on Neural Network Training and Accuracy

Master thesis in Engineering Physics

ALICIA REY ALONSO

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

MASTER THESIS PROJECT REPORT 2023

Impact of Training Data Volume on
Neural Network Training and Accuracy

ALICIA REY ALONSO



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023

Impact of Training Data Volume on Neural Network Training and Accuracy

ALICIA REY ALONSO

© ALICIA REY ALONSO, 2023.

Supervisor: Tomas Björklund, Principal Engineer within Deep Learning, Volvo Cars
Examiner: Giovanni Volpe, Department of Physics

Master thesis report 2023
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2023

Abstract

This master thesis explores the impact of data volume on model training and accuracy in the context of neural networks. The study focuses on conducting experiments on two image-based networks performing classification tasks, namely ResNet50 and MobileNetV2. The objective is to investigate the behaviour and accuracy of these networks as they are trained on progressively smaller subsets of the original dataset.

With this study, we aim to gain some insight into how neural networks perform under different data availability scenarios. This type of information can become key in decision making processes regarding data collection, model development, and output handling, particularly in situations where data volume is limited.

The research begins by establishing a baseline performance of the networks when trained on the entire dataset. Subsequently, various subsets of the original dataset are created by progressively reducing the volume of training data. The performance of the networks is then evaluated using these reduced datasets. This process allows for a comprehensive analysis of the effect of data volume on model training and accuracy.

Throughout all of this process, statistical studies will be carried out to verify the robustness of our results, as well as the possible influence the different subsets have on the results.

More specifically, the experiments involve training ResNet50 and MobileNetV2 models on subsets of the ImageNet-1K dataset, containing over 1.2 million training images across 1000 categories. The study examines how the reduction in training data volume affects the convergence of the models, as well as their accuracy in classifying images. Furthermore, the evolution of the network's confidence in its predictions evolves through training.

Keywords: data volume, model training, accuracy, neural networks, image-based networks, ResNet50, MobileNetV2.

Acknowledgements

I would like to express my deepest gratitude to my Supervisor Tomas Björklund, for his guidance, expertise and unwavering support, which have been instrumental in shaping this thesis. I would also like to extend my thanks to Volvo Cars, in particular to the R&D Analytics & AI group, for providing an accesible learning environment. Furthermore, I am grateful to Giovanni Volpe for taking the role of Examiner and to Hannes Johansson and Daniel Olander for taking the role of opponents and for teaching me Swedish.

Finally, I would like to express my deepest appreciation to my parents, my first mentors, whose unrelating faith, sacrifices and constant support through my academic pursuits have been the driving force behind my accomplishments. Thank you for teaching me to grow beyond.

Alicia Rey Alonso, Gothenburg, June 2023

Contents

1	Introduction	1
2	Aim and limitations	1
3	Background	2
3.1	Introduction to neural networks	2
3.1.1	Neurons and activation functions	2
3.1.2	Forward pass and backpropagation	5
3.1.3	Loss functions and optimisation techniques	6
3.2	Image-based networks	7
3.2.1	Convolutional neural networks	7
3.2.2	Residual neural networks	8
3.2.3	The architectures	9
4	Methodology	11
4.1	Statistical study	12
4.2	The dataset: ImageNet-1K	12
4.2.1	Creating subsets of ImageNet	14
4.3	The training script	14
4.3.1	Structure of the code	16
4.4	Metrics	17
5	Results	18
5.1	Statistical study	19
5.2	Selection of number of epochs to benchmark against.	22
5.3	Training runs for the different subsets of the dataset	24
5.4	Training and validation accuracies	28
5.5	Top accuracies versus data volume	31
5.6	Evolution of network's confidence through training	32
6	Conclusions	34
7	Future work	35
	Bibliography	37
A	Appendix	I
A.1	Subset maker code	I
A.2	Command prompts	II
A.2.1	MobileNetV2	II
A.2.2	ResNet50	II

1 Introduction

In many fields, obtaining labelled data for machine learning is a challenging and resource-intensive task. Be it medical imaging [1], natural language processing [2], or autonomous driving [3], the scarcity and expensiveness of quality labelled data poses a significant barrier in the development of accurate and robust machine learning architectures. In such contexts, it is paramount to understand the impact of training data volume on neural network training and performance, as this knowledge can be crucial in informed decision making over data collection, resource allocation, and model development.

Conventionally, these domains often follow a meticulous annotation phase followed by intensive model tuning and optimisation to obtain the best possible performance. However, recent research has highlighted the substantial influence of data volume on model generalisation and accuracy [4]. If data is limited, models are more prone to overfitting and become unable to generalise to unseen examples, regardless of how the architecture is curated and improved[5]. Consequently, it is essential to explore how different data volumes affect neural network training, aiming to identify the point at which additional data no longer significantly improves model accuracy.

This master's thesis aims to perform a comprehensive study on the impact of training data volume on neural network training and accuracy. Specifically, we will explore the relationship between different data volumes and the resulting top 1 and top 5 accuracy metrics, as well as analysing the confidence of the network on its predictions. By utilising the widely established ImageNet dataset and training two popular neural network architectures, ResNet50 and MobileNet, we will investigate how differently sized subsets of the ImageNet dataset influence model performance.

2 Aim and limitations

The main aim of the study is to be able to determine how much data is needed to perform machine learning. Practically, we aim to answer questions pertaining the effect data volume has on our chosen metrics: Does having more training data lead to better generalisation and improved accuracy? Is there a point at which increasing the training data size stops improving performance? Does the choice of architecture influence how data volume will impact performance?

In order to tackle these questions within the scope of a master's thesis, considerations need to be taken, namely restricting the domain of the study. The following limitations have been considered in order to conduct the study:

- Computer task and architectures: the study focuses on two specific model architectures (MobileNetV2 and ResNet50), both performing the task of image classification.
- Dataset: All experiments performed will use subsets or the full ImageNet1K dataset as training set.
- To evaluate performance, Top 1 and Top 5 accuracy metrics will be used in the study.

It is also important to note that the study focuses on the effect data has on the performance metric, rather than the tuning of a model. For this reason, we will not perform experiments designed to tune

parameters of the model, but rather select configurations we know can produce a good benchmark result.

With the scope of the study delineated, we dive into the necessary background needed to understand and contextualise the results.

3 Background

The theoretical background of this study is composed of machine learning architecture and training techniques. Therefore, an introduction on artificial neural networks will be given in section 3.1, providing an overview of its structure and components required for training. Subsequently, image based networks will be addressed in section 3.2, where convolutional neural networks and residual neural networks will be defined and an outline of both the architectures used in the study will be provided.

3.1 Introduction to neural networks

Neural networks have been one of the most groundbreaking developments in the field of artificial intelligence. Taking inspiration from the human brain, neural networks are computer models capable of analysing complex patterns, extracting key features and producing an output based on what it has learnt [6]. This set of abilities shines through when networks are set to different computational tasks such as image recognition, natural language processing, regression, etc. where the ability to learn from new data and adapt to new information is key. This has resulted in significant advances across multiple fields, such as the autonomous vehicle industry, medical diagnosis, recommendation systems, among others.

In the following sections we will go through the key components of a neural network, as well as their functioning.

3.1.1 Neurons and activation functions

A neuron is the computational unit of the neural network[7]. Neurons are organised into layers inside of the network, known as the input layer, a number of hidden layers (if any), and the output layer.

In order to explain the basic functioning of a neuron, we will use neurons found in a fully connected neural network (FCNN) The structure of a neuron can be found in figure 1. Originally developed by McCulloch and Pitts[8], the neural network aims to mimic the behaviour of a human neuron by collecting information from multiple incoming signals and, once the necessary threshold has been reached, produce an output. Mathematically, the model looks as follows:

$$y = f \left(\sum_{i=1}^n x_i w_i + b \right), \quad (1)$$

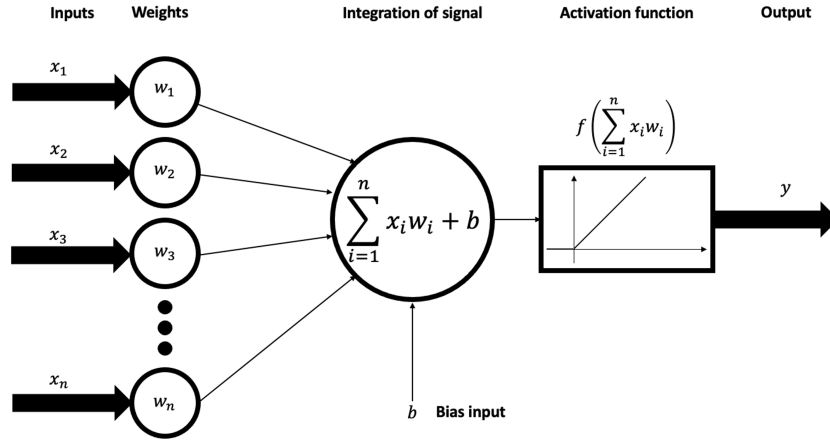


Figure 1: Diagram depicting the inner workings of a neuron in a FCNN. All inputs enter the neuron with a learnable weight, where the output is computed through an activation function. Image taken from [9] under a CC BY 4.0 license.

where y is the output of the neuron, f is an activation function, which we will discuss in below, and for each input i , w_i is the trainable weight of the neuron connected to the network input x_i . Finally, the neuron also has a trainable bias term b .

As we mentioned before, neurons are arranged in layers, so the expression above can be generalised to define the output of an entire layer, where j refers to a specific neuron:

$$y(j) = f \left(\sum_{i=1}^n x_i w_{ij} + b \right) = f(z_j), \quad (2)$$

where we have now also defined z_j as the activation of the neuron, the input to the activation function. Activation functions play a key role in introducing non-linearity into the network. Without them, our model only consists of linear transformations, which only yields a linear relationship between input and output. This poses a restricting limit into what the network is able to portray, whereas the introduction of non-linear activation functions allow to capture more complex relationships. Furthermore, activation functions introduce a threshold that is essential in defining decision boundaries[10]. By mapping the output of a neuron to a specific range, complex decision boundaries can be created, which is essential for a multitude of computer vision tasks.

There is a multitude of activation functions, however several have become the most commonly used, due to their suitability for different tasks:

- Sigmoid: Also known as the logistic function and widely used in earlier neural network models, the function maps the input to a range between 0 and 1, which means the input, under the right circumstances, can be interpreted as a probability. It is worth noting that the choice of the sigmoid might have unwanted consequences for inputs with extremely high absolute values, as the function saturates leading to vanishing gradients [11]. The function is defined

as:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

- Hyperbolic Tangent (tanh): This activation function works similarly to the sigmoid, but instead mapping the input to the range $[-1, 1]$. While it suffers from the same saturation problems as the sigmoid, tanh is symmetrical with respect to the origin, which allows it to handle negative inputs with more success [12] and reach convergence faster. The expression for the tanh function is as follows:

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (4)$$

- Rectified Linear Unit (ReLU): one of the most present in nowadays' architectures, the ReLU function sets negative inputs to 0, while keeping positive inputs unchanged. This function offers several advantages, including alleviating the vanishing gradient problem for positive inputs as well as computational efficiency [13]. However, it can suffer from the "dying ReLU" problem, where neurons can become inactive if they consistently receive negative values. The equation for the function is:

$$f(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

A variant of this function is the leaky ReLU[14], which addresses the "dying ReLU" problem by assigning a small non-zero slope to negative values in the inputs. This allows negative values to transmit through the network and avoids dead neurons.

In figure 2, a representation of the three aforementioned activation functions:

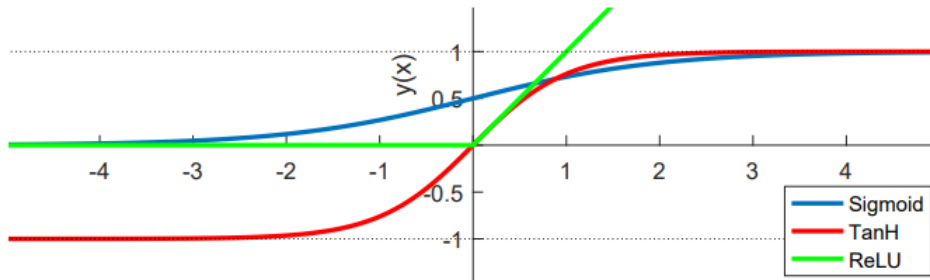


Figure 2: Plot of the non-linear activation functions sigmoid (blue), tanH (red) and ReLU (green). Image taken from [15] with permission.

A fourth activation function is worth mentioning, however it is singled out from the others due to its difference in output and overall use. The softmax activation function is most commonly used in the output layer of multi-class classification tasks. It takes a vector as input and normalises the values into a probability distribution, such that the sum of all values equals to 1. The expression for this activation function is the following:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (6)$$

It is worth noting, however, that although the function produces a probability distribution, when used in the output layer of a neural network it is not reporting on the probability of the input belonging to each of the classes, but rather on the *confidence level* the network predicts each class with.

3.1.2 Forward pass and backpropagation

Once we have the model created, the neural network needs to learn. This process occurs by iteratively adjusting the weights and biases of the network based on the output errors of the predicted labels of the network and the ground truth of a batch of training data for each iteration (explained further into this section). This process can be divided in two key sections:

- Forward pass: During a forward pass the input data gets propagated through the network and a prediction or output is produced. Following the neuron structure pictured in 1, the input data is fed into the neurons of the input layer, where the weighted sum is computed for each of the neurons to be then fed into the activation function. This produces the output of the neuron layer, which is then fed into the next neuron layer as input. This process continues until the output layer is reached and a prediction is made based on the calculated activations. Depending of the problem, the format of the output can vary, with a single value for binary classification, or a confidence vector for multi-class classification.
- Backpropagation: This is the part of the learning algorithm in charge of implementing the actual learning. To do this, the network performs an error calculation between the predicted label of the network versus the true label of the data through a chosen loss function, which will be discussed in the following section. A backward pass is then performed, where the error is propagated from the output layer all the way to the input layer. Next, the gradient of the error (the loss function) is computed via the chain rule. This is a way to quantify how much each neuron contributed to the overall error, which then allows the final update of the weights. This last step is performed by an optimisation algorithm, based on the previously calculated gradients and a learning rate, in control of the “step size” of the updates. The former is selected from a range of suitable functions, and will also be discussed in the following section.

When both a forward pass and backpropagation have been computed for the entirety of the training set, an epoch has elapsed. During each epoch, the model processes the entire dataset, divides it into smaller subsets called batches, and updates its parameters based on the computed gradients from each batch sequentially. Batches, on the other hand, are subsets of the training data that are processed together in parallel to perform forward passes and backpropagation for parameter updates. Dividing the dataset into batches allows for more efficient computation and optimisation, especially when dealing with large datasets that may not fit entirely into memory. Training is then performed until a specified number of epochs or until some form of convergence (usually determined by the loss function) is reached.

This constitutes the learning experience of the network, and allows it to adjust its parameters to learn key features and gain a deeper understanding of the data set. It is also worth noting that learning can be enhanced by adding extra measures to the backpropagation algorithm, such as momentum, regularisation, dropout, etc.

3.1.3 Loss functions and optimisation techniques

The choice of appropriate loss functions and optimisers plays a crucial role in the network's performance, as they are the defining features of the backpropagation algorithm discussed in the previous section. The loss function quantifies the discrepancy between the output of the label and the ground truth, while the optimiser is in charge of updating the weights and biases to minimise the loss function. Below we list some of the more relevant functions used for the two:

Loss functions

- Mean Squared Error (MSE): Most suitable for regression tasks where the output is a series of continuous values [16], the function measures the average squared difference between the prediction made by the network y_{pred} and the true label of the input data y_{true} . The equation is as follows:

$$\text{MSE} = \frac{1}{2} \sum (y_{pred} - y_{true})^2 \quad (7)$$

- Binary cross-entropy loss (BCE): As its name indicates, this function is used in binary classification problems. It quantifies the dissimilarity between predicted label and true predicted label. Mathematically, the expression to calculate BCE is shown below:

$$\text{BCE} = -\frac{1}{n} \sum (y_{true} \log(y_{pred}) + (1 - y_{true}) \log(1 - y_{pred})) \quad (8)$$

- Categorical cross-entropy loss (CCE): Often referred to just as cross-entropy loss, it is a generalisation of the BCE to suit for a multi-class classification problem[10]. Its expression is as follows:

$$\text{CCE} = -\frac{1}{n} \sum y_{true} \log(y_{pred}) \quad (9)$$

Optimisation techniques

- Stochastic gradient descent (SGD): This is one of the most widely used optimisation algorithms. Its key feature is introducing stochasticity into the model by computing the gradients on mini-batches and updating the weights instead of computing the gradients for the entire training set. This also improves efficiency and memory usage, and can increase convergence time[17]. Furthermore, SGD typically features a fixed learning rate, in charge of determining the step size taken when updating the weights. Adjusting the learning rate becomes a balancing game between reaching convergence fast enough without inducing overshooting due to unstable updates.
- Adaptive learning rate methods: These type of techniques improve on the fixed learning rate of SGD by dynamically updating the hyperparameter to improve convergence and accelerate training. Some of the most common are adaptive gradient (AdGrad), root mean square propagation (RMSprop), and adaptive moment estimation (Adam)[18].

This concludes the general introduction into the inner workings of a neural network. Our experiments will be run in convolutional neural networks (CNNs) and residual neural networks (ResNets), both networks which take images as an input. Therefore, we will now delve into the relevant characteristics of these two types of architectures.

3.2 Image-based networks

Image based networks are referred to those that take an image as input and through deep learning techniques analyse and extract meaningful features within the images. This allows to perform a wide variety of tasks, like image classification, object detection, image segmentation, and so on. We will discuss convolutional neural networks, one of the most prominent architectures in the field of computer vision, and residual neural networks, a modification of the traditional convolutional network structure to allow for deeper network training.

3.2.1 Convolutional neural networks

Popularised in 1998 by LeCun et al. [19], CNNs are a type of neural network that excels at analysing grid like data, particularly images. Their main characteristic is the use of convolutions as a way to extract key features from the input image. Convolutional blocks often have the following components:

- Convolutional layers. These layers apply filters (also known as kernels) to the input image. Convolution operations occur as the filter slides across the input image, using the network's weights to generate a set of output feature maps, as shown in figure 3. This is the part of the network in charge of feature extraction [15], where the network extracts and learns relevant features from the input data. These types of layers are stacked together to create deep networks capable of detecting complex features and connections.
- Activation function. The output feature maps then go through an activation function (commonly ReLUs in the case of CNNs) which introduces non-linearity and enhances the expressiveness of the function, allowing it to draw more complex relations between the input and the output.

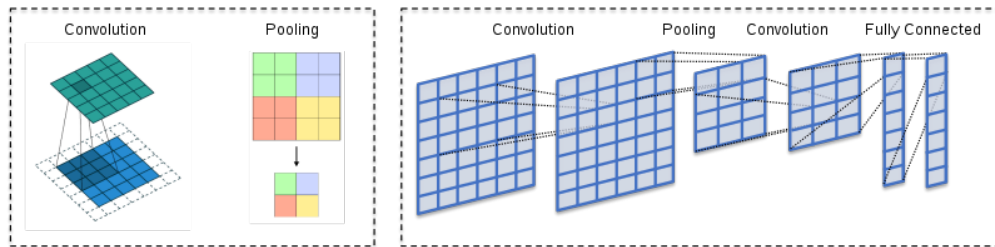


Figure 3: Diagram of different elements of a CNN. On the left, the application of convolutional filters and a pooling layer. On the right, a simple structure of a CNN alternating convolutions and pooling layers ending in fully connected layers. Image taken from [20] under a CC BY 4.0 license.

- Pooling layers. While not a necessary element, convolutional blocks often have pooling layers which downsample the feature maps while retaining the most relevant information. The network thus becomes less vulnerable to small translations in an image [21] and is able to capture higher-level features. This reduction in dimensions implies a reduction in number of parameters in subsequent layers. A common technique used is MaxPool, where the criteria to select which information is passed down into the downsampling is selecting the maximum value within defined sections across the input. An illustration of this technique can be found in figure 3

- Classification block. Typically, fully connected layers (FCNs) are stacked together to extract semantic information from extracted features. As mentioned before, these layers connect every neuron in one layer to every neuron in the subsequent layer, enabling the network to learn global relationships. Typically, the fully connected layers are followed by activation functions and a softmax layer for final output classification.

In figure 4, a diagram of a simple CNN architecture is depicted, showing how all the components are traditionally stacked together to create a network. In CNNs forward- and backpropagation works as described for the fully connected network described above, although each convolution weight is exposed to multiple connections in each sample, which essentially means the weight update becomes averaged based on all backpropagation feedback.

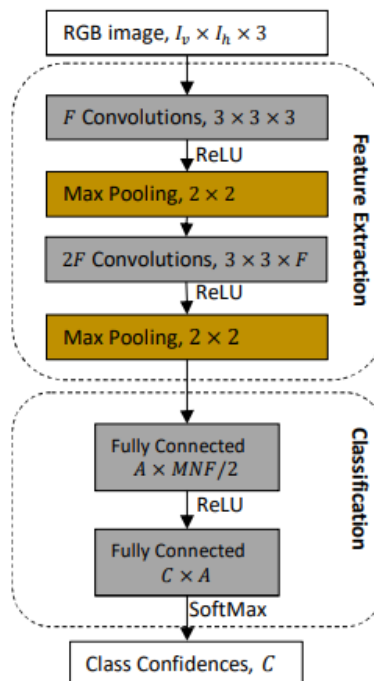


Figure 4: Diagram of a simple CNN. Image reproduced from [15] with permission.

3.2.2 Residual neural networks

ResNets address the challenge of training very deep convolutional networks. CNNs reached a limit in terms of how many hidden layers a network could have before the vanishing gradient problem arose and accuracy dropped [22]. ResNets were first proposed by Kaiming He et al. in 2016 [23] and provided a solution to this problem through the use of residual connections. The essence of this technique is that by adding identity mappings, the network learns the residual to these identity mappings rather than the actual desired mappings. At the beginning of the training, this significantly improves the flow of information through the network compared to starting with only random filters as traditional CNNs do.

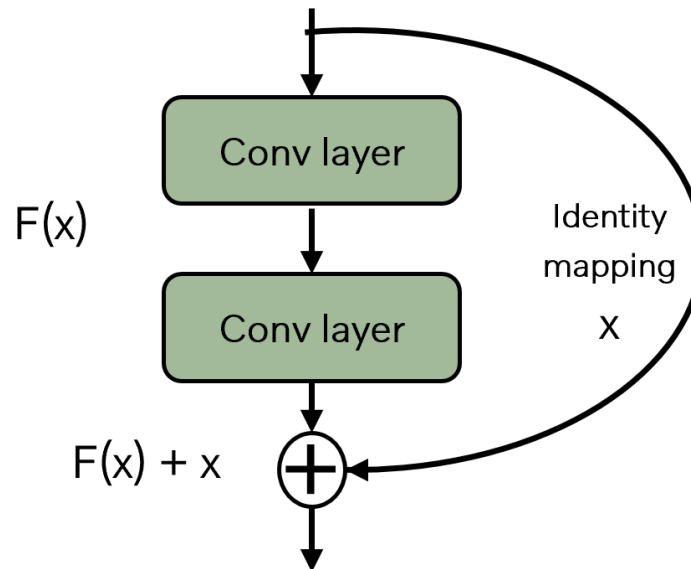


Figure 5: Diagram of a residual block. In the straight arrow connections, a ReLU function is being applied to the convolutional layer output.

The key element that was introduced to ResNet that differed from previous CNN implementations was the introduction of residual blocks with an identity mapping shortcut through the block. A diagram of this new feature can be seen in figure 5. It consists of a series of convolutional layers, followed by a non-linear activation function, traditionally a ReLU function. The network also contains a shortcut, also referred to as skip connection, which is in charge of performing the *identity mapping*. These connections allow the flow of information directly to later layers, without passing through any activation functions. Thus, the network can use the input from these connections to model the residual mapping, which can be understood as the difference between the desired output and the input [24].

The use of residual blocks and skip connections provides shortcuts for the gradients when performing backpropagation, addressing the issue of the vanishing gradient problem and allowing for training of very deep networks. Furthermore, skip connections allow information to be preserved across the layers, mitigating the risk of important features getting lost in the deeper layers of the network.

The two discussed architectures are a blue print for a myriad of models that employ state-of-the-art techniques to improve performances. In the following section we will discuss the specific architecture of the two networks used in the study.

3.2.3 The architectures

For this study, the architectures chosen were MobileNetV2 and ResNet50. These two networks are widely recognised and extensively used deep neural network architectures for image classification tasks. They represent different architectural design choices, such as depth-wise separable convolutions in MobileNetV2 and skip connections in ResNet50. By selecting these two representative models, the study can provide insights into the generalisability and robustness of training outcomes across different architectural structures.

MobileNetV2 is a variant of CNNs created with the objective of making a network suitable for mobile phones[25]. This resulted in a computationally efficient and lightweight network, with quicker training times than some of the widely studied convolutional models. In order to achieve this, some modifications to the traditional CNN architecture had to be added to reduce model size and computational complexity while still providing a competitive accuracy:

- Depthwise separable convolutions. In a regular convolutional block a separate filter is applied to each input channel and the output is produced by combining the results of the channels. In a depthwise separable convolution the process occurs in two stages. The first remains unchanged as with a regular convolution block, and is referred to as the depth-wise convolution. However, when it comes to obtaining the output, a pointwise convolution is applied, by which a 1x1 convolution is used to combine the feature maps from the previous stage. This significantly reduces the number of parameters and calculations, resulting in a lighter model. An overview of the structure of the depthwise separable convolutions can be seen on figure 6.

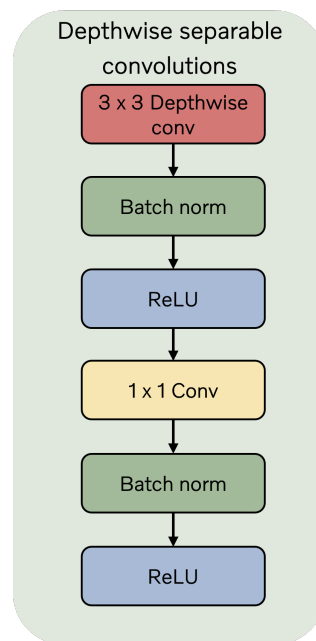


Figure 6: Diagram of Depthwise separable convolutions introduced in MobilenetV2. Batch normalization is performed after each convolution, and a ReLU activation function is applied to the outputs.

- Inverted residuals with linear bottlenecks. Similar to residual connections seen in residual networks, a skip connection is employed to connect the beginning and the end of a convolutional block. In this particular architecture, the use of point-wise and depth-wise convolutions to create a bottleneck structure which allows the dimensions to remain as they were.
- Width multiplier. The addition of a hyperparameter that controls the number of channel in each layer allows for the scaling of the network size at each layer. This causes a slight decrease in accuracy, and thus a trade-off between lightness and speed of the model, and accuracy

appears.

ResNet50 is a specific implementation of the ResNet architecture, firstly introduced in the original ResNet paper [23]. The number "50" refers to the total number of layers found in the network, accounting for convolutional layers, pooling layers, activation layers and fully connected layers. It was specifically designed with the tasks of image recognition and object classification in mind. The architecture has achieved noteworthy performance in various computer vision tasks, and is widely used as a backbone architecture for many state-of-the-art models such as Faster R-CNN[26] and Mask R-CNN [27].

The main structure of ResNet50 can be seen in figure 7. As mentioned earlier, residual blocks and skip connections are its key components, along with other features such as bottlenecks intended to reduce computational complexity while maintaining the overall performance of the network.

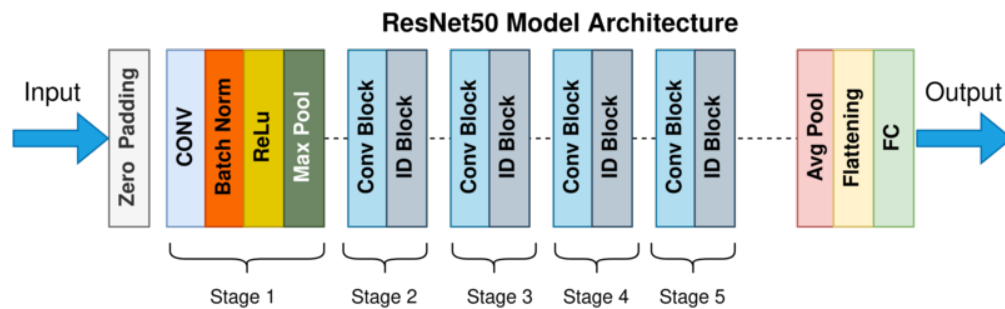


Figure 7: Diagram of ResNet50 architecture. A convolutional block is followed by an identity block, which performs the identity mapping and propagates the residuals. Image reproduced from [28] under a CC BY 4.0 license.

Furthermore, due to its sheer popularity amongst researchers, ResNet50 has a wide availability of pre-trained weights across large-scale datasets, which makes the network a prime choice to be used in transfer learning tasks.

Overall, ResNet50 played a crucial role in pushing the boundaries of accuracy and performance in computer vision tasks, and due to its popularity, it proves a good network to benchmark with.

With the theoretical knowledge for the study explained, we now proceed to the chosen methodology to conduct the experiments.

4 Methodology

We aim to study the effect data quantity has on the accuracy and overall training of the aforementioned architectures. In order to do this, we will have to perform iterative trainings of the networks with different quantities of data, and observe the effects on different parameters on the network.

4.1 Statistical study

To ensure statistical accuracy and the reliability of the results, we will perform several instances of the same training. This will be carried out in different ways. Firstly, we will compare training runs on the same training set done from scratch using the decided upon metrics to evaluate differences in performance. Secondly, while still maintaining the same training set, we will select a specific epoch in one of our completed train runs and re-train the network from said epoch.

Due to inherent randomness in the training of neural networks, we expect to see slightly different results. However, we aim to determine whether these differences are quantifiable enough to be significant or insignificant enough to consider repeated statistical experiments redundant.

Another point to consider is whether the subset of the dataset we select has an effect on the overall training results. Therefore, a similar study must be conducted for different subsets of the same size to study whether the subsets will produce relevant results.

With this in mind, we have performed two full runs of training on ResNet50, and a third run where we re-started training at a specific epoch. In terms of the influence of the data distribution effect on the results, we have performed three training runs on MobileNetV2, where on each run we used a different subset containing 5 % of the ImageNet training data set.

Based on the results of this study, the number of iterations each training is performed will be determined. Experiments have a high computational and memory cost, given all models must be stored for analysis. While ideally we would like to perform each experiment several times and average the results, this would set an unreasonable time frame for the obtention of results, therefore determining when enough experiments will give a robust result is crucial for the completion of the study.

We now turn to look at the main components of the experiment: the dataset, and the training script.

4.2 The dataset: ImageNet-1K

ImageNet-1K [29], often referred to simply as ImageNet, is one of the most commonly used benchmark datasets in the field of computer vision. It has been instrumental in advancing the state-of-the-art classification models and remains prominent in tasks requiring pre-training of weights such as transfer learning.

ImageNet1K has its origins in the superset ImageNet22K, containing close to 15 million images classified across roughly 22,000 synsets, synonym sets, which represent groups or collections of synonymous words or terms that represent a specific concept or object category. The classes are assigned using WordNet's database [30], which follows a hierarchical structure where more general concepts are represented at higher levels, with more specific labels appearing at lower levels. This allows for easier navigation of the dataset.

The original 22K dataset was created for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [31], which aimed to evaluate image recognition models. However, its sheer size and substantial amount of classes proved hard to work with and a smaller dataset was required. ImageNet-1K was created by carefully choosing 1,000 object categories from the original 22,000 with

the objective to preserve the wide variety of objects and visual complexity the superset provided.

With respect to data volume, the training set contains approximately 1.3 million images distributed across 1,000 classes, with 1,300 images on average per class. Figure 8 shows how images are distributed across each class, with most classes containing approximately 1300 images while few classes have as little as 750.

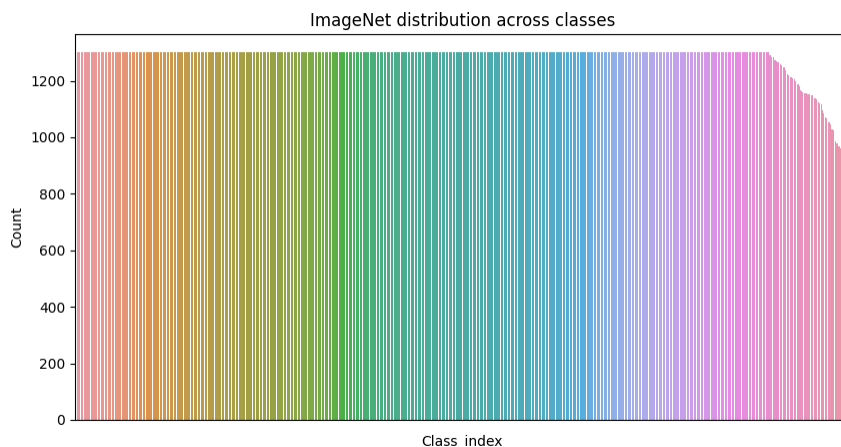


Figure 8: Distribution of images across classes for ImageNet. The classes have been sorted by number of images in descending order.

The validation set contains 50,000 images distributed evenly across all classes, with 50 images in each class. Note that the proportions of images per class are not the same as those portrayed in the training set, but rather we have a fully balanced dataset. This permits the evaluation process to become more balanced and avoids biases that could arise from unequal class representation. It also allows for a more objective assessment of the model's performance across different classes and ensures that no particular class has a disproportionate influence on the evaluation results.

All of the images vary in resolution, with the biggest being 4288 x 2848 pixels and the smallest being 75x76 pixels. Pre-processing will be applied prior to the training loop to prepare the data for the network, including resizing the images to 224 x 224 pixels.

As is commonly used in image classification tasks, the study will be focusing in the top-1 and top-5 metrics, which will be discussed in section 4.4. The use of these two metrics is particularly relevant for datasets with a large number of classes, where networks often predict with lower confidence levels across all categories.

On another note, ImageNet, like any large-scale dataset, poses several ethical considerations that must be kept in mind. These mostly stem from the vast amount of personal and/or sensitive data and possible biases present in the dataset. Some of these are listed below:

- ImageNet contains a vast collection of images obtained from multiple sources, including the internet. This means that obtaining explicit consent from all the individuals depicted in the

dataset is not always feasible. This raises concerns regarding privacy and informed consent of individuals whose images were used without their knowledge.

- Large datasets are prone to contain sensitive information, such as personally identifiable information (PII), confidential documents or private scenes. Data anonymisation should be used in order to address these concerns and safeguard individuals' privacy as well as prevent any misuse of the data.
- The dataset may contain biases in terms of cultural representation, perspectives and under-represented communities. This is a well known concern across machine learning models that ImageNet is not excluded from. It is an important issue to be aware of, as biased datasets can produce biased results, reinforcing existing social prejudice and discriminatory practices[32].

We recognise the importance of addressing privacy concerns associated with the use of the ImageNet dataset, therefore to ensure the protection of individual privacy, only the neural network models will be exposed to the images in the dataset, nor will the report display any specific images from the ImageNet dataset. Instead, we focus on the analysis and discussion of the overall trends, performance metrics, and observations derived from the trained models. Our emphasis lies in understanding the impact of data volume on the model's accuracy and confidence, rather than individual images or their content. Furthermore, the trained networks will only be used for benchmarking purposes, the individual classifications taken by these networks will never be used for anything else than contributing to the average performance calculations.

In order to study the influence of data volume on our chosen architectures, subsets of our dataset must be created, which will be tackled in the following section.

4.2.1 Creating subsets of ImageNet

When creating subsets of ImageNet, important considerations need to be taken. Firstly, the amount of classes must remain the same across all subsets. Given that the objective is to analyse the same problem, but with less amount of data, reducing the number of categories would simplify the problem. Nonetheless, the ratio of images across classes will be preserved, to avoid introducing any bias that wasn't already present in the original problem.

It is also relevant to use random sampling to create the subsets, while preserving the constraints we have mentioned above. This helps guarantee the subset is representative of the original dataset and mitigates the introduction of bias that could arise from manual selection.

For the practical creation of the subset, Kaggle's "Create ImageNet Train Subset 100k" [33] has been used as base code and modified to fit our specifications. The code is provided in Appendix A.1.

4.3 The training script

To perform the experiments we chose to follow PyTorch's "How to Train State-Of-The-Art Models Using TorchVision's Latest Primitives" [34]. PyTorch offers pre-trained models in their libraries which often serve as benchmark for new models or as a building block for future transfer learning.

In the article, Pytorch provides detailed instructions on how to achieve the accuracy benchmarks, which served as a starting point to test whether we could achieve the same results with our set up, and as a first data point for what training on 100 % of the data set results in.

The training recipe provided includes several steps and settings aiming to achieve the best accuracy results possible for the models existing models, amongst which the two chosen can be found. We will now look into some of the important features of the recipe:

- **Automatic augmentation:** Data augmentation consists of a series of specific transformations designed to artificially increase the size of a dataset, such as cropping, rotations, random erasing, colour distortions, etc. Traditionally, these transformations would be selected manually, however this can be quite time consuming and not produce satisfactory results. Automatic augmentation, often referred to as AutoAugment, uses a search algorithm to explore possible augmentation practices and select those that optimise model performance. In the recipe provided, TrivialAugment is used [35], which is parameter free and performs only a single augmentation to each image.
- **Learning rate (LR)**
 - Learning rate scheduler: The technique consists in adjusting the learning rate throughout the training process rather than leaving it constant throughout the entirety of the experiment. Typically, a learning rate is decreased over time to stabilise training as the model gains understanding of the dataset. LR schedulers can take different forms, at times dependent on a fixed number of epochs (step) or more complex functions which can perform adjustments on individual learning parameters.
 - Learning rate warm-up: Before applying the scheduled learning rate, training begins with a reduced learning rate which gradually increases according to some specific function until the designated end of the warm-up stage is reached. The goal behind this is to mitigate common problems that arise during the early stages of training such as overshooting (occurs when a too high learning rate at the beginning of training causes the optimiser to take large steps in the weight space, which in turn can cause strong fluctuations in the loss function). The added stability provided by the learning rate warm-up helps the model explore the loss function space more suitably, improving the chances of finding better solutions during training.
- **Label smoothing:** Regularisation technique which aims to address networks becoming overconfident or the appearance of overfitting. Classification tasks typically use one-hot encoding, where an image is assigned a single correct class label (typically being 1), and the rest of the classes will be assigned as the "wrong" class (indicated with a 0). Label smoothing introduces a degree of uncertainty into these predictions, erasing their binary nature and making the model more robust against unseen data.
- **Mixup and CutMix:** These two data augmentation techniques focus on making use of several images present in the training set in order to produce new samples, which preserve characteristics from all images used in creating them.
 - Mixup combines pairs of samples from the training data set to create new synthetic samples. The motivation behind this is to create new images with interpolated characteristics. Mathematically, this is done as follows:

$$\begin{aligned}x &= \lambda x_1 + (1 - \lambda)x_2 \\y &= \lambda y_1 + (1 - \lambda)y_2\end{aligned}\tag{10}$$

In equation 10, we have a pair of training samples (x_1, y_1) and (x_2, y_2) , where x refers to the input and y to the true output, and λ is a mixing coefficient randomly sampled from a beta distribution [36], which controls the interpolation between the two samples. By blending the features and labels of different samples, mixup encourages the model to learn more robust and generalised representations.

- CutMix combines image patches from different training samples, using a cut-and-paste strategy [37]. A random bounding box is fixed on one of the input images. A crop of this box is performed, and it is then pasted on another image on the same position as the bounding box was in the original image, thus creating a new sample. As for the label, a weighted average of the original labels is computed, accounting for the size of the swap. This technique encourages the network to focus on understanding and classifying partial object regions while considering the context of the entire image [37].
- **Weight decay:** This technique, also known as L2 regularisation, introduces a penalty term to the loss function which penalises large weight values and encourages the model to learn smaller weights [10]. This term is controlled by a weight decay hyperparameter, which should be tuned to improve performance. Higher values of the parameter lead to stronger regularisation, whereas lower values minimise the impact of the decay.

4.3.1 Structure of the code

Pytorch’s repository [38] has been chosen to conduct our training experiments, mainly because of the classification models already provided in the aforementioned repository. The repository contains the main script `train.py`, as well as four auxiliary scripts that contain utilities necessary to perform the training¹. We now provide a brief structure of the main training script:

1. The script begins importing the necessary dependencies from the auxiliary scripts which will perform upcoming tasks such as data manipulation and evaluation.
2. The datasets are loaded and prepared for training. The training-validation split has already been performed upon loading the datasets, so the script creates the dataloaders needed for training and applies the chosen transformations to the data.
3. The model is defined. The script will load a saved architecture from Pytorch’s library. The choice to load pre-trained weights, either from Pytorch’s library or from a custom stored model is also available.
4. Relevant parameters for the training are set, such as the learning rate, learning rate schedulers, choice of optimiser and loss function.

¹There is a fifth script available in the repository, `train_quantization.py`, used to train quantised models. That type of model has not been used in this study and therefore neither has this script.

5. The training loop is the core part of the script, where the model trains by performing forward passes and backpropagation to update its weights. The script tracks the desired metrics, as well as informing of any updates in the dynamical parameters, such as the learning rate scheduling.
6. After an epoch of the training pass is performed, an evaluation pass on the validation set is performed, where the metrics for the epoch are computed.
7. The script stores each epoch's architecture and weights as a .pth file for future use.

To launch the script, Pytorch's tool TorchRun is used. TorchRun is a utility tool that streamlines the workflow by allowing easy definition of the aforementioned tasks described before [39]. The desired settings for the training are provided as arguments to the function, which allows the user to evade the complexity of setting up the training loop, allowing researchers to focus on model development experimentation.

Finally, in terms of hardware, all calculations have been performed in 4 NVIDIA T4 GPUs through VoVo Cars' cloud environment.

After the description of the experiments have been conducted, we delve into an explanation of the metrics to be used.

4.4 Metrics

In this study, the evaluation of the multi-class classification task will be based on two primary metrics: accuracy Top-1 and accuracy Top-5. These metrics provide information on how the model performs in terms of the most confident prediction (Top-1) and the top five predictions (Top-5). The choice of this metrics is driven by their relevance in assessing the models ability to correctly classify input across multiple classes.

Accuracy Top-1 represents the proportion of correctly predicted instances where the true class label matches the model's highest-confidence prediction. It provides a measure of how well the model performs when making single-class predictions. A higher Top-1 accuracy indicates better performance in accurately assigning the most probable class label to the input samples.

Similarly, accuracy Top-5 takes into account whether the true label of the sample can be found amongst the 5 highest-confidence predictions the model has made. This metric is particularly useful when dealing with tasks where we can be more lenient with our results, especially when in classifications with such an amount of classes.

In image classification tasks other metrics are often evaluated as well, such as precision, recall and F1-score, particularly in binary classification. Recall, which measures the ability to correctly identify positive instances, and precision, which assesses the accuracy of positive predictions, are commonly used in imbalanced datasets or when certain classes require higher emphasis. F1 score combines both precision and recall into a single metric, providing a harmonic mean that balances the two[40].

Given that the multiclass classification task performed in this study aims to evaluate overall prediction accuracy across all classes, we have deemed to focus on the two aforementioned metrics, as the dataset of choice doesn't favour any one class, and our specific classification problem has no preference over minimising true negatives or false positives.

Furthermore, we are also interested in analysing how the confidence of the network evolves as training progresses. As previously stated, the confidence of a network refers to its level of certainty in the correctness of its predictions. Observing how the network's confidence evolves when making correct and incorrect predictions can give us insight on how well calibrated the network is, essential information in tasks with strong emphasis on reliable probability estimates, such as medical diagnosis.

Instead of focusing solely on absolute confidence values, we chose to examine the confident margin as a measure of confidence. The confident margin is defined as the difference between the highest and second-highest confidence scores for a given prediction[41]. The reasoning behind this choice is that, since the convergence of the ResNet50 model was not yet complete, the absolute confidence values might not accurately reflect the model's certainty. By focusing on how the confidence margin evolves through training, we could analyse the relative confidence of the top prediction compared to the second-highest prediction, which provides insights into the model's decision-making process.

We will choose to evaluate a normalised confidence margin following the formula:

$$C_{12,norm} = \frac{C_1 - C_2}{C_1}, \quad (11)$$

where $C_{12,norm}$ is the normalised confidence margin, C_1 is the confidence of the Top-1 prediction and C_2 is the confidence of the Top-2 prediction.

In this methodology section, we have presented a comprehensive overview of the experimental setup and procedures employed in this study. The selection of the dataset, choice of neural network architectures, and training procedure have been described in detail. Moving forward to the results section, we will now present the outcomes of our experiments, including the accuracy scores, and confidence analyses.

5 Results

In this section the findings of the conducted study are presented. We will begin by looking at the results of the statistical study in section 5.1, which will determine how many runs of each training setup are needed. Subsequently, section 5.2 will focus on the full long trainings of both networks used to determine what the number of epochs to train for all networks would be, followed by the results of training across different subsets of data in section 5.5. Lastly, we will study how the confidence of the network evolves throughout training in section ??.

All of these results were obtained by launching the training through a command instruction, containing all the relevant information in terms of parameters, data augmentations, regularisation techniques, etc. In the report, only the relevant parameters will be mentioned, i.e. those that differ

between runs, however the full instruction (and therefore training configuration) can be found in appendix A.2.

5.1 Statistical study

We will first focus on the statistical study, since its results will determine the amount of samples we must take to consider our results robust, that is, how many training runs under the same condition must be repeated to obtain an acceptably stable result. This is an essential step in our study, given that all experiments are time consuming, which means that we cannot afford to perform as many calculations as would be desired, but rather those that will provide a clear overview as to what the results are.

We begin by studying whether equal training runs produce noticeably different results. We have done this by performing two instances of training on the full ImageNet data set in ResNet50. In order to save computational time, the training has been performed not focusing on convergence, but rather on performing a long enough training to observe whether the behaviour of both iterations mimicked each other. With this in mind, training on both iterations was performed for 40 epochs.

A third run of training was started from one of our previous training runs' halfway point. This aims to reproduce the possibility of a stopped training experiment that we wish to restart. The results can be seen on figure 9.

Looking at figure 9, we can observe that all three runs closely follow each other, which leads us to conclude that runs of the same training set up using the same training set only need to be run once to obtain veritable results, unless unexpected variations are detected.

Next, the study on the influence of the data subset used is performed. Using MobileNetV2 on this occasion, we generated three different subsets of ImageNet, each containing 5 % of the data. As we have mentioned before, these subsets were created respecting the distribution of the classes, and the samples were drawn at random, allowing for potential overlap between the samples in different subsets. The objective of the study is to observe the effect data variability and the stability of the model, rather than studying the effect each section of the data has towards overall performance. We aim to reproduce real life situations in which redundant or similar samples are an occurrence, therefore avoiding full overlap is unnecessary. Nonetheless, the overlap should not be a considerable amount of the dataset, lest we will perform the same training twice. In table 1, we represent the overlap between the three data sets:

	Subset 1	Subset 2	Subset 3
Subset 1	-	4.95 %	4.91 %
Subset 2	4.95 %	-	5.11 %
Subset 3	4.91 %	5.11 %	-

Table 1: Overlap of ImageNet 5% subsets for statistical study

The overlap consistently is close to 5 % (which was to be expected, as any two random 5% subsets of a superset should expect approximately 5 % overlap between them), and considering that the

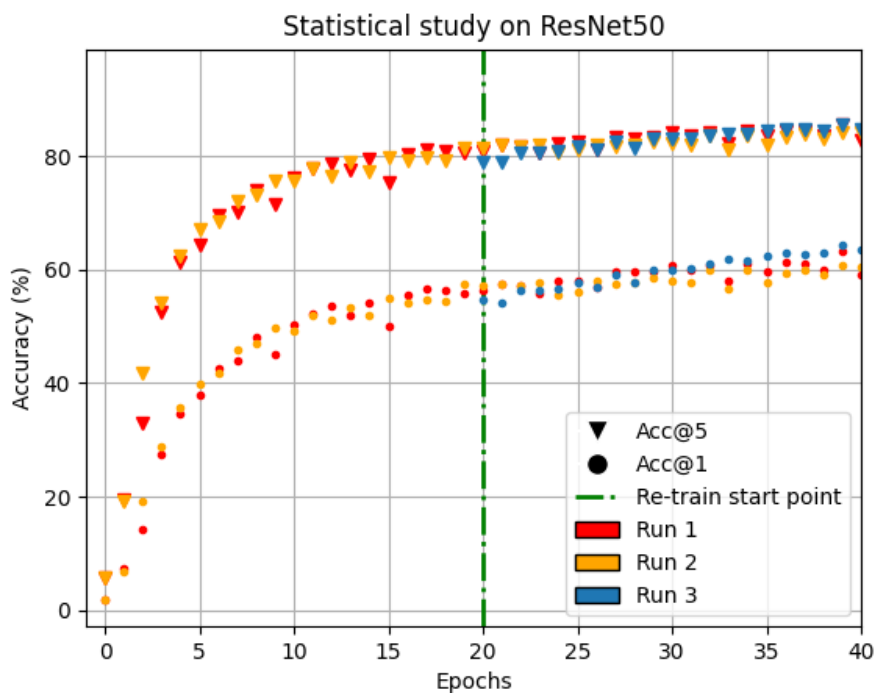
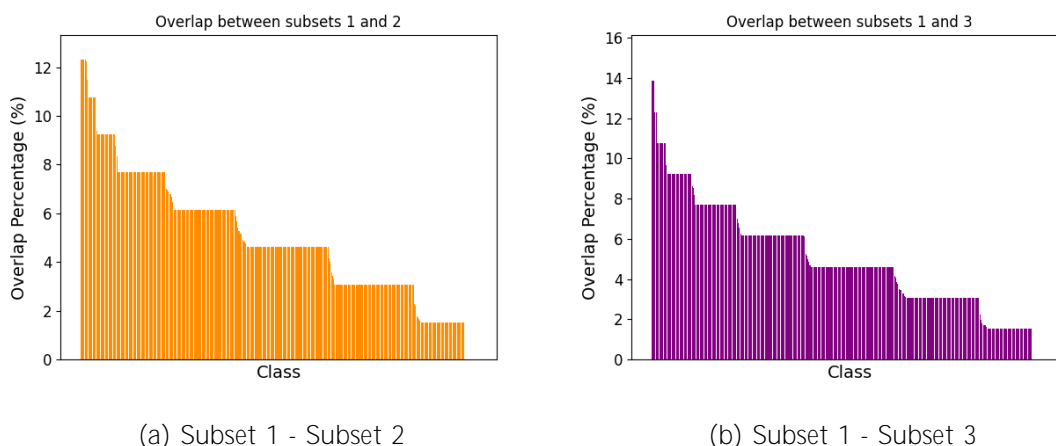
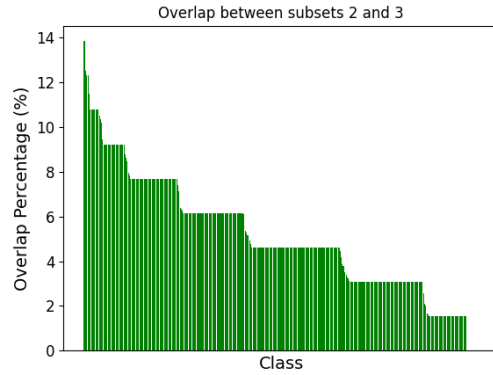


Figure 9: Top-1 (Acc@1) and Top-5 (Acc@5) accuracies of the validation set for the same training regime. The green dashdotted line indicates the epoch at which the weights from run 1 were used to start run 3.

size of the 5 % subsets is of 64062 images, this means approximately 3200 images can be found in both of the subsets compared in each case. These percentages only pertain to the overall dataset. However, it is also interesting to visualise what the overlap looks like for each class. Even though from figure 8 we can see most of our categories exceed the 1000 images, some are as low as 750, so it is important to see whether the overlapping images have concentrated on only some categories, meaning a high percentage of overlap. The graphs on figure 10 show that, at most, the classes experience 14 % overlap, which we can accept and work with.





(c) Subset 2 - Subset 3

Figure 10: Percentage of overlap in classes (relative to class size). All three subsets overlap graphs have been sorted by amount of overlap per class in descending order

We now move into the actual training results. In figure 11 we see how the Top-1 and Top-5 accuracies on the validation set evolve as the network trains. The three subsets produce similar enough results, therefore we can conclude that the subset of ImageNet we choose has no significant effect on the overall results of our training. Using these results, further experiments will only be run once for each size of the data set in order to save considerable computational time.

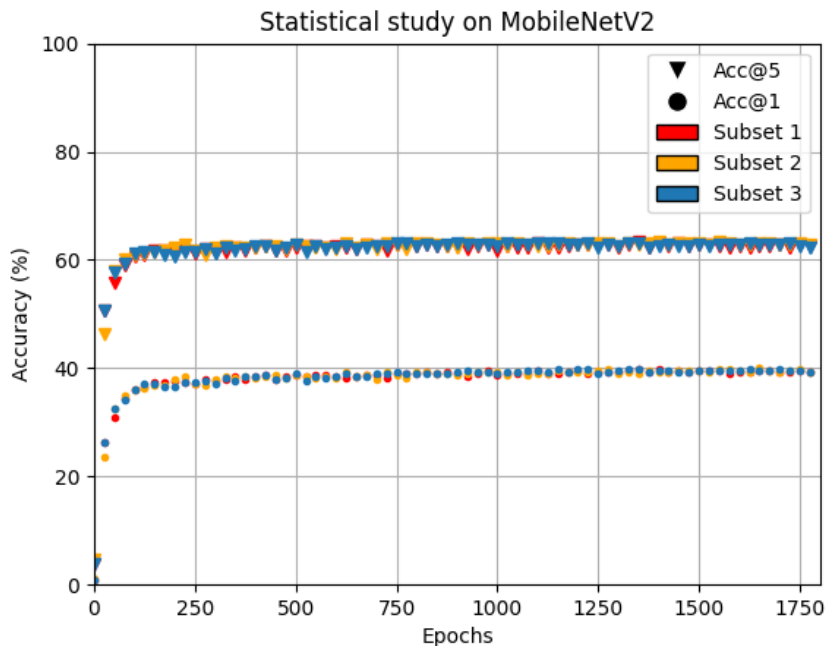


Figure 11: Top-1 and Top-5 accuracies for the 3 different 5% subsets on the validation check. Note that, for visualisation purposes, the figure is representing a data point every 20 epochs, replicating the amount of points 100% of the data would produce. The original 1800 epochs trend showed slightly more noise, however nothing significant to skew the proposed results.

When it comes to training time, shown in table 2, we see very little difference among the 3 subsets, 0.52 % between the lengthiest and fastest training, which can be reasoned to be due to rounding of the timer used to count, overall occupation of the GPUs, and other experimental set up reasons. Therefore, we also conclude the subset choice had no effect on training time.

	Training time
Subset 1	1 day, 14:28:32
Subset 2	1 day, 14:16:56
Subset 3	1 day, 14:19:28

Table 2: Training time of the three subsets on MobileNetV2

Due to the results of the statistical study, we proceed only running each specific training setting once, confident in the fact that the results will be robust for both networks.

5.2 Selection of number of epochs to benchmark against.

To begin with, we performed a long training, following the training recipe in [34] on the totality of the dataset. This training aimed to match the benchmark provided by Pytorch’s pre-trained models, as well as serve as this study’s benchmark for the subsets to compare against. Starting with MobileNetV2, the results of said training can be seen on figure 12.

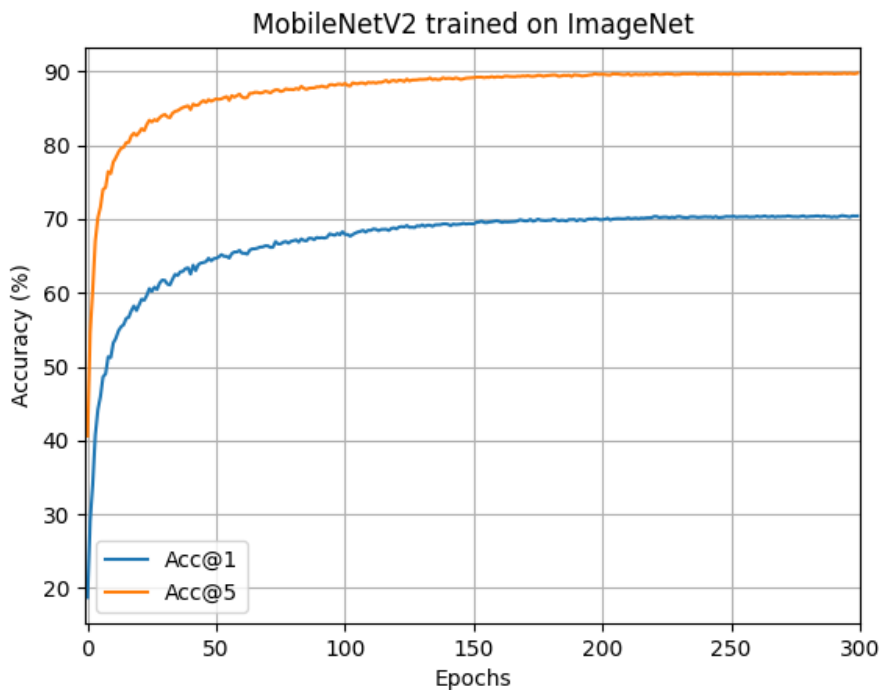


Figure 12: Top-1 and Top-5 accuracies for the long training performed on MobileNetV2.

Both accuracies skyrocket early on in the training before growing stagnant after slightly over one

third of the training. The full training took approximately 4 days and 4 hours, and managed to reach the Top 1 and Top 5 accuracy values shown in table 3. Our training was able to get considerably close, less than 2 percentage points below the provided benchmark. This could be due to multiple reasons, e.g. hardware. Pytorch’s training was performed on 8 GPUs, which allowed for bigger batch sizes. Ours was performed in 4, therefore, the batch size suggested had to be adjusted for the script to be able to run. Nonetheless, we still consider the training results satisfactory and proceed with the analogous experiment for ResNet50.

	Acc@1	Acc@5
Model after 300 epochs	70.428	89.784
Pytorch’s benchmark	72.154	90.822

Table 3: Accuracies of the validation set for our MobileNetV2 training compared to those provided by Pytorch’s model library.

ResNet50’s long training results can be seen in figure 13. We see the very same sharp increase at the beginning of training observed in MobileNetV2 in figure 12. However, unlike MobileNetV2, ResNet50 does not converge, instead it keeps increasing at a slower rate throughout the entire performed training. It is worth noting that in the suggested implementation [38], a long training of 600 epochs was used to meet the provided benchmark. Given that each epoch of ResNet50 took approximately 52 minutes to train, the full training of ResNet50 would have taken almost 22 days, which in turn would have made the completion of this study impossible. The benchmark provided by Pytorch, compared to the one obtained in the training is provided in table 4.

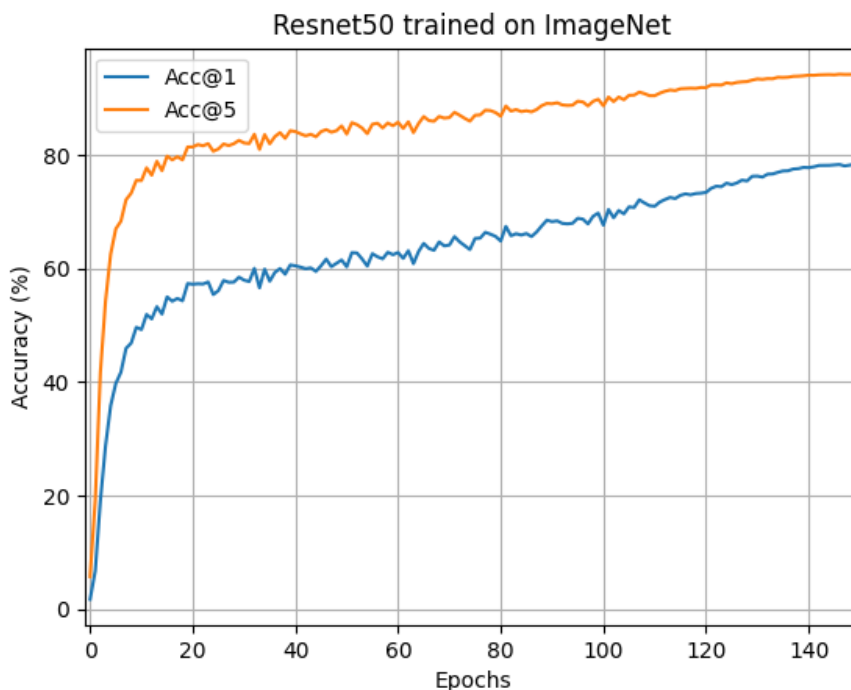


Figure 13: Top 1 and Top 5 accuracies for the long training performed on MobileNetV2.

	Acc@1	Acc@5
Model after 150 epochs	78.378	94.226
Pytorch's benchmark	80.858	95.434

Table 4: Accuracies of the validation set for our ResNet50 training compared to those provided by Pytorch's model library

There are less than 3 percentage points separating our Top-1 accuracy from the one provided by the benchmark, so we judge the training performed was enough to determine that our experimental set up is correct.

In order to be more efficient and save computational resources, we will cut down the training of further subsets, given that the majority of the accuracy increase happens in the early stages of training. Furthermore, we hypothesise that smaller amounts of data have less information to learn from, therefore, training should converge and grow stagnant quicker. This means that, in the case of MobileNetV2, we will benchmark the rest of our subsets to those obtained after the network reached 90 epochs. The choice of 90 epochs can be explained by observing that the accuracy curves in figure 12 have begun stabilising by this epoch and that, in order to achieve some of the desired percentage of subsets, a number of epochs divisible by 3 was required. For ResNet50, due to the extensive amount of time required to train each epoch (nearly three times as much time as MobileNetV2, which has an average time of 18 minutes per epoch), a more conservative number of epochs had to be selected. In this case, we will choose to study how the fast growth seen in the first epochs in figure 13 changes depending on the amount of data, and will use the 60 epoch metrics as our benchmark. The Top-1 and Top-5 accuracies for both of the benchmarks are reflected in table 5, which will be the accuracies that will henceforth be referred as accuracies using 100 % of the dataset.

	Acc@1	Acc@5
MobileNetV2 after 90 epochs	67.512	87.916
ResNet50 after 60 epochs	62.464	85.184

Table 5: Accuracies of the validation set for 100 % of the dataset

With these parameters settled, we move to the training using the subsets of ImageNet.

5.3 Training runs for the different subsets of the dataset

The selected percentages of data are 5 %, 12.5 %, 25 %, 50 %, and 75 %. It is worth noting that the training script runs by specifying number of epochs, and contains dynamic methods that adapt the learning rate. Therefore, when we launch our trainings, all these parameters must be adjusted to ensure that the same amount of adaptations happen at the same time in all trainings. In table 6, the relevant parameters that require adjustment are presented.

For MobileNetV2, all runs were performed as described in table 6. Furthermore, no learning rate warm up was used. In ResNet50's case, we omitted performances on data subsets bigger than 12.5 % (apart from the 100 % run), given that the 12.5 % subset already matched performance with the full dataset run, and this saved computational time and resources for further tests.

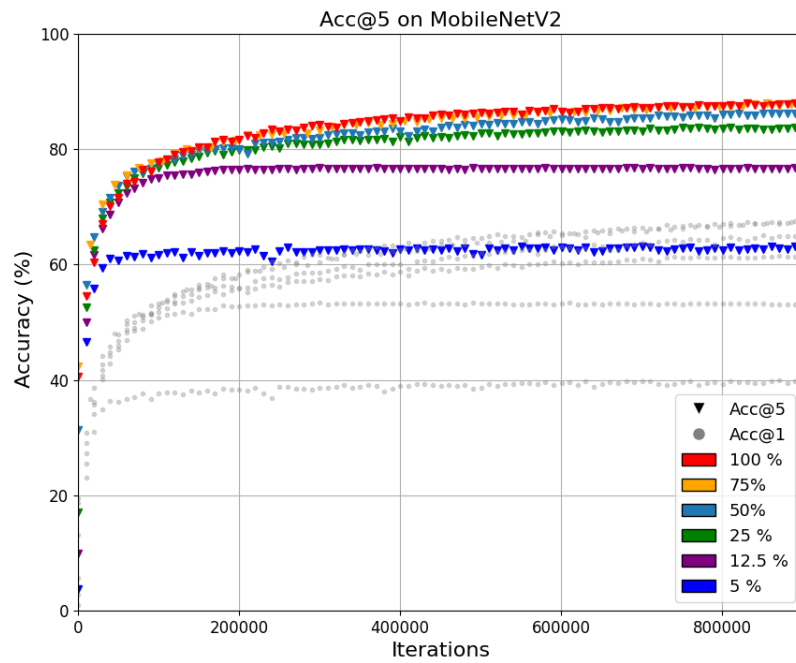
Additionally, during the analysis of the results, it came to our attention that ResNet50's 12.5 %

Percentage of original dataset (%)	MobileNetV2		ResNet50		
	epochs	LR step size	epochs	LR warm up	LR step size
5	1800	20	1200	100	600
12.5	720	8	480	40 5	240 30
25	360	4	240	20	120
50	180	2	120	10	60
75	120	1	80	7	40
100	90	1	60	5	30

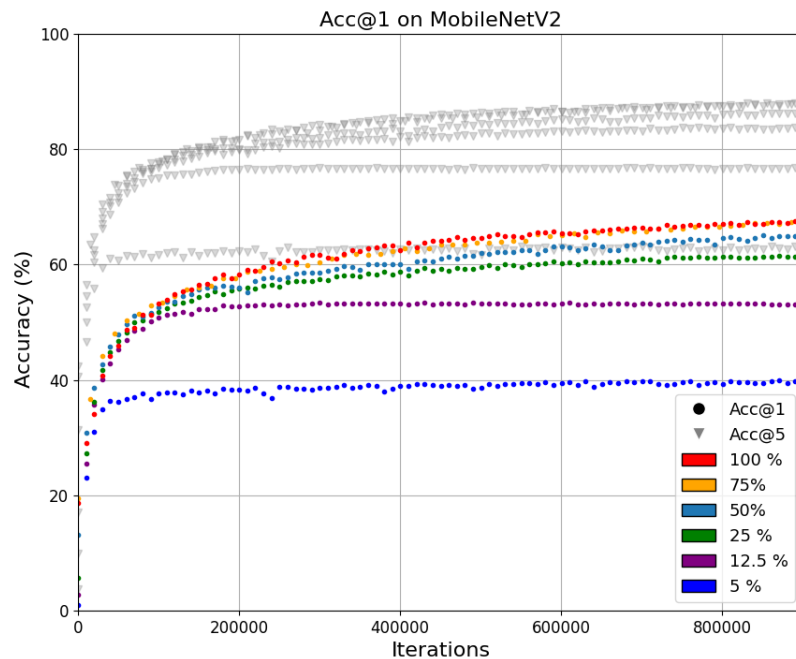
Table 6: Proposed runs and relevant parameters for MobileNetV2 and ResNet50. The strikethrough values in ResNet50 correspond to runs that were not carried out. The red values for the 12.5 % subset run are the values that the training was eventually performed in, rather than those the training should have been performed in.

run was inadvertently performed using the learning rate parameters of the 100 %, which may have influenced the results of this training run. Therefore, it is crucial to interpret and analyse the findings with the awareness that the learning rate had a smaller warm up than it should have and a faster learning rate decay.

We move towards the results of the training performed with the stipulated parameters in table 6. Starting with MobileNetV2, in figure 14 the validation accuracies of the subsets are represented with respect to the number of iterations (i.e., the number of backpropagations that occur). It becomes clear that the lower the amount of data, the lower performance our network obtains. Furthermore, there appears to be a considerable leap between the 5 % of the data values and the 12.5 % ones, and a noticeable one between the 12.5 % subset and the 25 % subset. Increasing the subset size forward does yield better results, but the improvement is minimal, until 75 % of the dataset converges with the full dataset results.



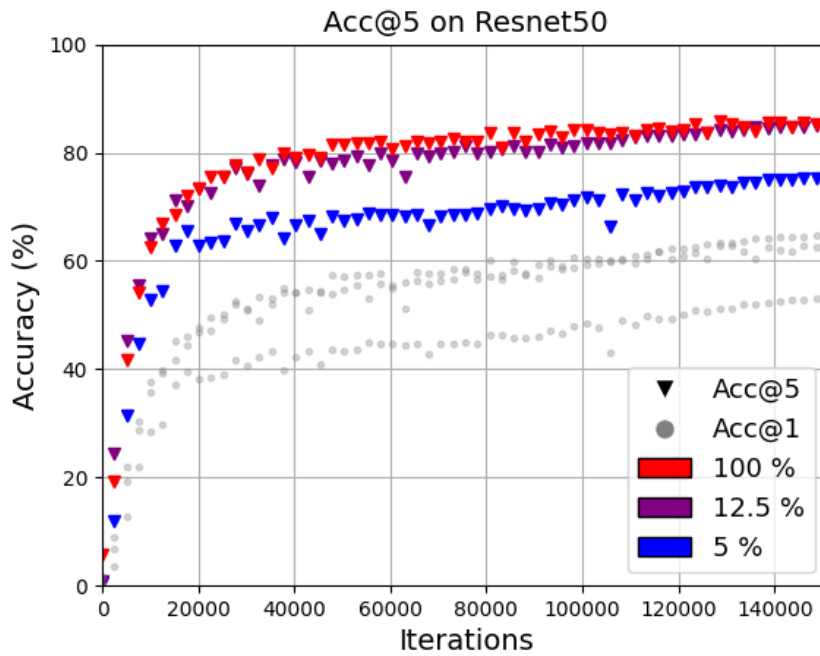
(a) Top-5 accuracy



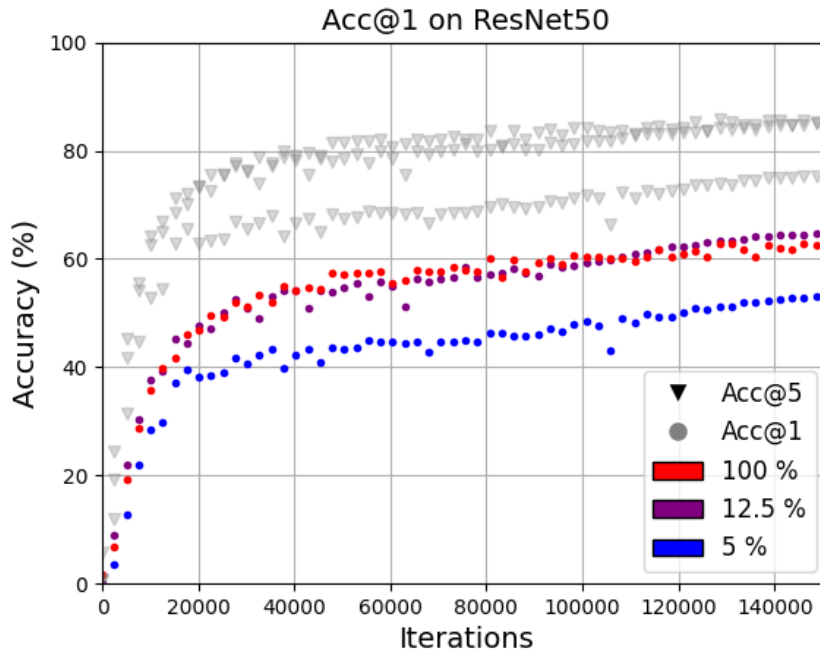
(b) Top-1 accuracy

Figure 14: Top-1 and Top-5 validation accuracies for the different subsets of data, trained on MobileNetV2. The greyed out data points on both graphs show the non-represented accuracy, for better visualisation of the results.

Moving on to ResNet50, the results of the study can be seen on figure 15. As was done for MobileNetV2, iterations is used rather than epochs to analyse the validation accuracies. Comparing these results to those reflected in figure 14, we can observe ResNet50 is much more robust against training volume reduction, with the 12.5 % run reproducing the Top-1 and Top-5 accuracy of the full data set. While we must account for the possibility the use of incorrect parameters has produced a better performance for the specific subset, a test run with 25 % of the data was started, which also closely reproduced the performance of the full dataset. This still confirms the statement that ResNet50 can achieve the desired accuracy with smaller amounts of training data. Moreover, ResNet50's 5 % subset run does not appear to converge earlier than the full dataset training, as was the case for MobileNetV2, but rather mimics the slower increase in accuracy that the training on the full dataset displays, albeit with a smaller overall accuracy. We must take into account that we have not allowed either run to converge in the experiment, which doesn't allow us to speculate on whether smaller subsets would have reached convergence earlier than the full dataset.



(a) Top-5 accuracy



(b) Top-1 accuracy

Figure 15: Top-1 and Top-5 validation accuracies for the different subsets of data, trained on ResNet50. The greyed out data points on both graphs show the non-represented accuracy, for better visualisation of the results.

It is worth noting that, for both networks, regardless of the size of the subset, we don't see the accuracy in the validation set decrease at any point, but rather, its growth slows down for ResNet50 and remains stagnant for MobileNetV2. We might have expected to see a drop in performance for lower amounts of data after a certain number of iterations, as an indicator that the network was overfitting, however that has not been the case. To investigate further, we proceed to look at the training curves compared to the validation curves for each of the datasets.

5.4 Training and validation accuracies

In figure 16 we can see all the training and validation curves for MobileNetV2. These curves show that for the subsets above 50 % of the data set, figures 16a, 16b, and 16c, both the training sets and validation sets accuracies were increasing when we stopped training. For 25 % of the data, figure 16d, the training accuracies were still seeing some growth, whereas the validation accuracies had converged. In the case of the 12.5 % subset, convergence is seen for both the validation set and the training set.

Once more, we note that the training and validation curves in figure 16 do not reflect any overfitting, but rather stagnant accuracies on the training and validation sets. This can be attributed to the complexity of the training script use, which uses methods like dropout, data augmentation and batch normalisation, known to mitigate overfitting [42].

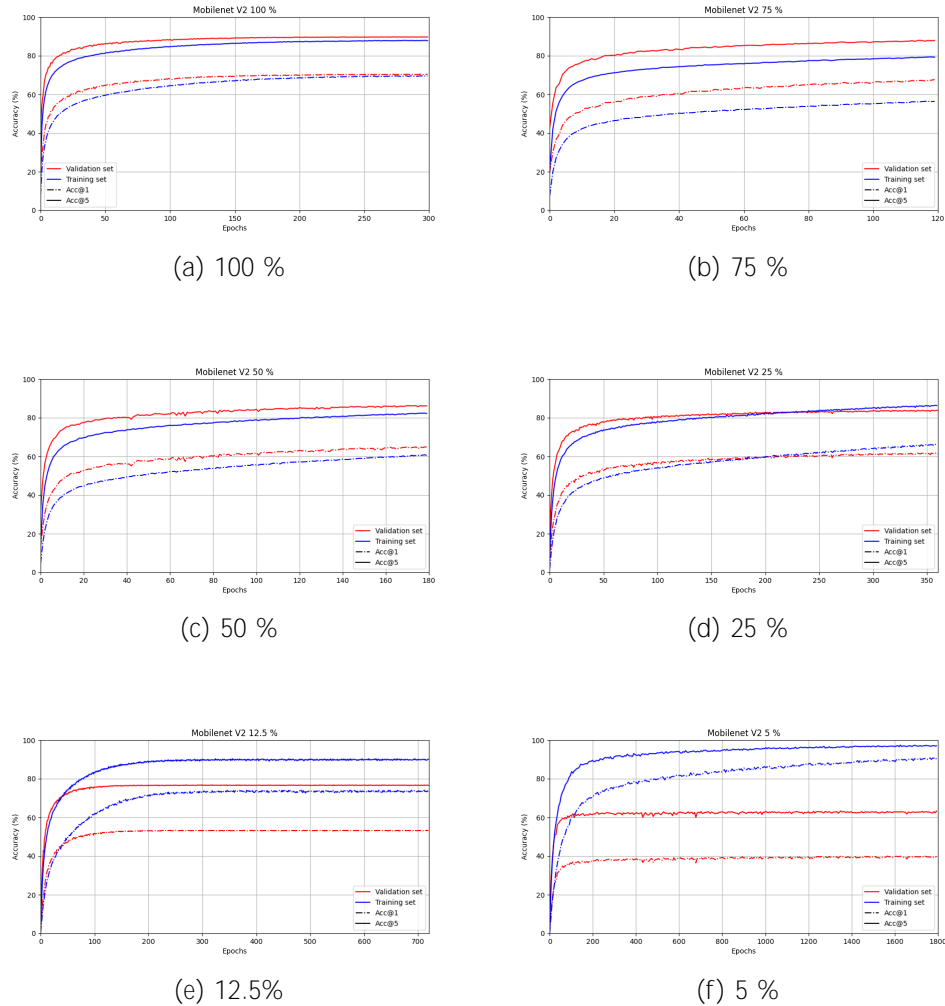


Figure 16: Training and validation curves for the different subsets for MobileNetV2.

It is also worth mentioning how the validation set has higher accuracies than those present in the training set for the bigger subsets of the data, while the smaller subsets reflect the commonly seen trend of training accuracy over validation accuracy. This can also be explained by the presence of regularisation techniques in the training script. Dropout and weight decay, both of which are applied during training, introduce a form of controlled noise that aims to prevent overfitting [42]. When a training set is small, these techniques become particularly important, as the model will be more prone to overfitting. Therefore, the smaller subsets do not display the characteristic drop in validation accuracy after extended training, given that the techniques used give the training set enough diversity to still yield acceptable results.

Conversely, the bigger datasets, which would not induce overfitting, are training on a set of data that has been made more complex to mitigate that very same phenomenon. This shows a better performance in the validation set, given that it is an easier set to classify with respect to the training set, which displays lower accuracy values.

This conclusion is backed by figure 16, where we see the difference between training and validation curves diminish as the data increases, until eventually the shift occurs in the 25 % subset in figure

16d. The training accuracy overtakes the test accuracy as the model is trained for too long and the regularisation techniques shift from improving performance to mitigating overfitting.

The training and validation curves for the Top-1 and Top-5 accuracies training on ResNet50 are displayed on figure 17. As occurred with MobileNetV2 in figure 16, we see no clear signs of overfitting, given that there is no drop in accuracy in any of the validation accuracies.

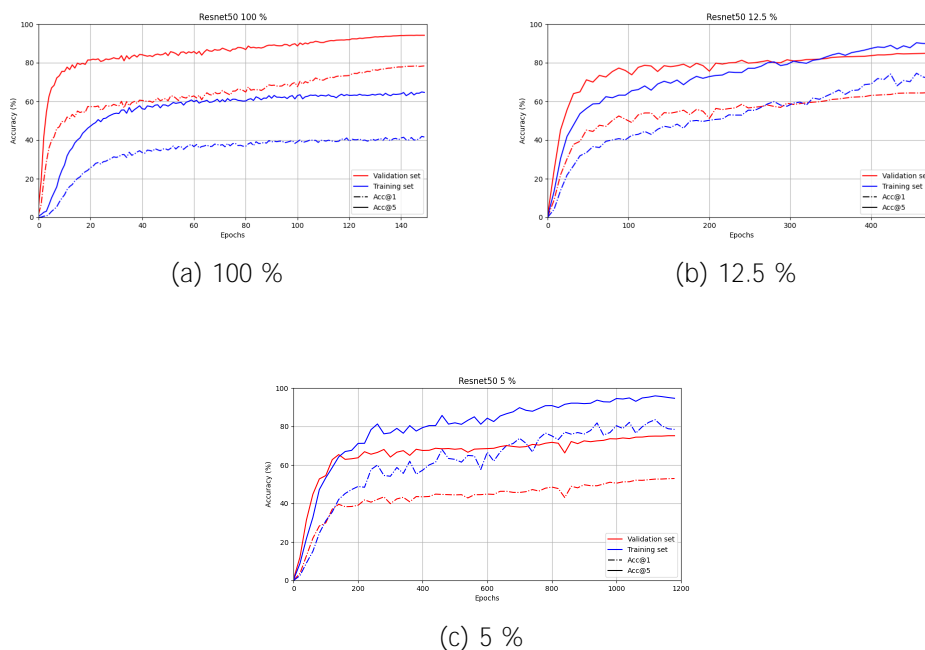


Figure 17: Training and validation curves for the different subsets for ResNet50.

However, we have selected to train ResNet50 for a comparatively shorter training time than MobileNetV2, that is, the training finishes before the network reaches convergence. Therefore, we might have encountered overfitting in later stages of training.

ResNet50's experiments show a similar behaviour between the training and test accuracy curves than that observed for MobileNetV2. In the subset with the full dataset, depicted in figure 17a, the validation set has higher Top 1 and Top 5 accuracies than the training set, whereas the 5 % subset sees the opposite, with the training accuracies higher than the test accuracies. ResNet50's residual connections allow for the easier flow of gradients during training, enabling the network to learn intricate representations through the use of residual mappings. The residual connections allow for training of very deep networks, and makes them less prone to overfitting. As the dataset size increases, ResNet50 benefits from this architectural advantage and the mitigated possibility of overfitting allows for the regularisation to encourage the network to learn more robust and transferable features. This yields a network that is able to better generalise to unseen examples, leading to higher testing accuracy.

Having examined the convergence patterns of the training and validation curves, we now delve into exploring the relationship between data volume and the best accuracy achieved by our models. This investigation seeks to shed light on the impact of dataset size on the model's ability to attain its

highest level of accuracy.

5.5 Top accuracies versus data volume

In figure 18 the best and last values for Top 1 and Top 5 accuracy achieved on MobileNetV2 for different subsets of ImageNet. It becomes clear that for this network we see a continued increase in performance as data volume increases, until the accuracy values saturate at 75 % of the dataset. Furthermore, we can further observe that there appears to be a period of rapid growth at the beginning of the curve, before the increase in both Top-1 and Top-5 accuracies slows down.

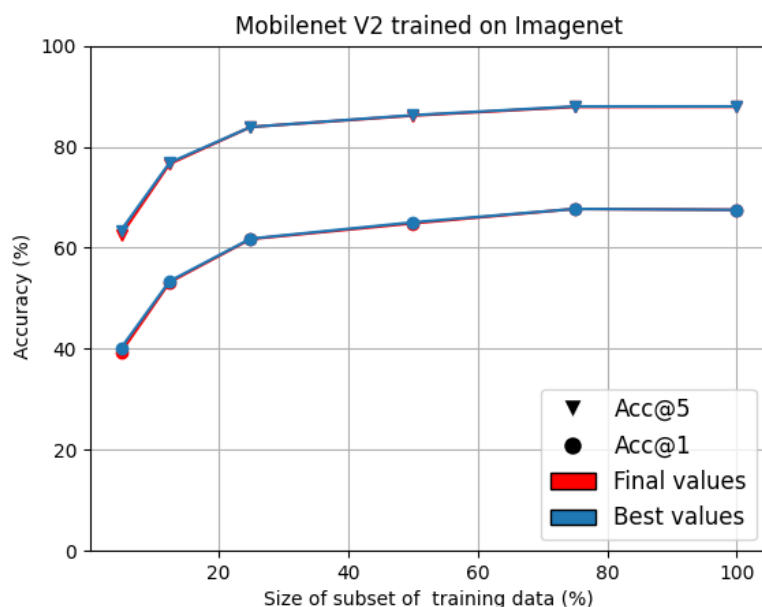


Figure 18: Best and last values for the Top-1 and Top-5 accuracies for the different subsets. Training performed on MobileNetV2

It is also worth noting that in most cases, the best values and the final values overlap fully, with slightly more deviation in the smaller subsets. This is consistent with the graphs presented in figure 16, where the smaller subsets accuracy metrics grew stagnant early on, and given that the curve has stochastic oscillations, the last iteration does not necessarily need to have been the best one.

ResNet50's best accuracies for the validation sets of different subsets of ImageNet are displayed in figure 19. As in MobileNetV2, we see there is a sharp increase in accuracy for the smaller subsets, after which the Top-1 and Top-5 accuracy values saturate. The 12.5 % subset performs slightly better in Top-1 accuracy than the full ImageNet training set, however this presumably due to the smaller learning rate step size resulting in quicker updates of the learning rate.

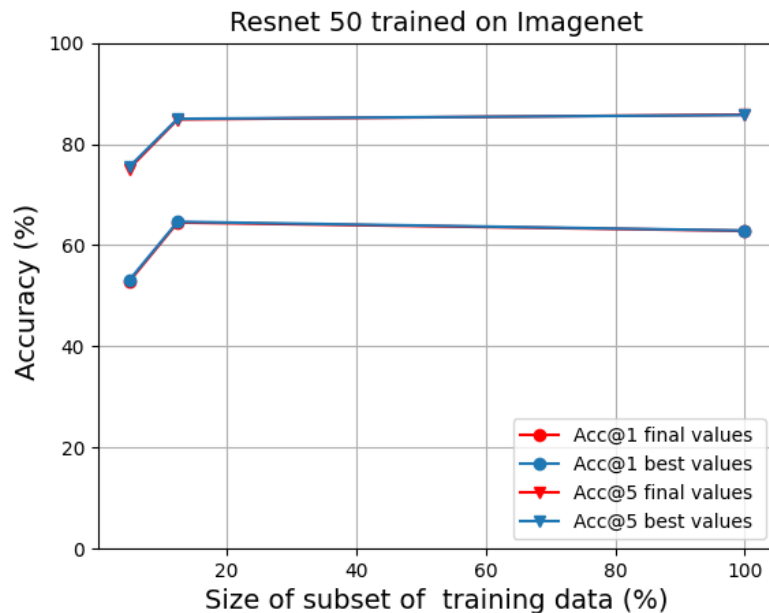


Figure 19: Best and last values for the top 1 and top 5 accuracies for the different subsets. Training performed on ResNet50

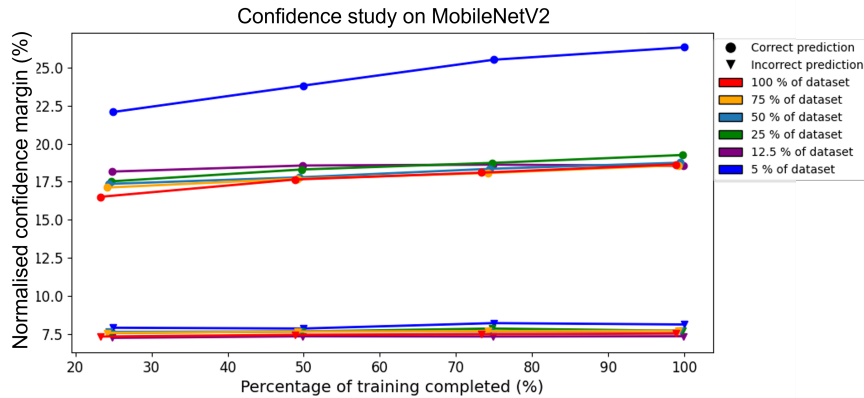
Both networks appear to indicate that there is a certain data volume after which adding more samples to the training set will not yield an improvement in performance. Furthermore, the relationship between data volume and accuracy of the network is not a linear one. The same increase in samples for smaller datasets has a considerably bigger impact than those performed for bigger datasets.

With a comprehensive understanding of the impact of data volume on achieving the best accuracy, we now shift our focus to another crucial aspect of model performance: the evolution of the network’s confidence in its predictions.

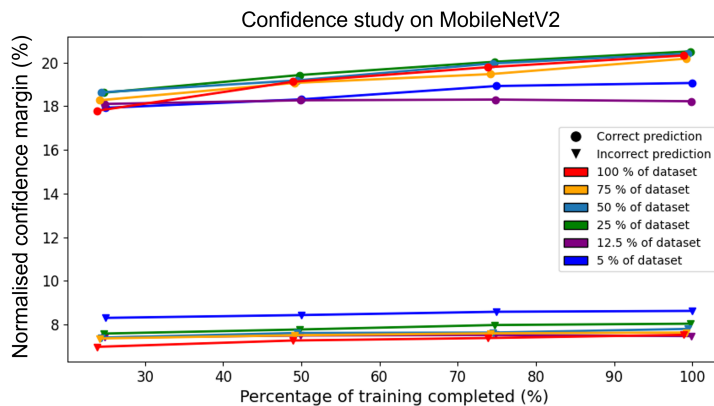
5.6 Evolution of network’s confidence through training

Understanding how the confidence in predictions evolves through network training is crucial to assess the model’s learning process and its level of certainty in making accurate predictions. In this section, we investigate the evolution of confidence margins across different training datasets of varying sizes.

Starting with MobileNetV2, the normalised confidence margins on the training and validation set can be found on figure 20. It is noteworthy that the network demonstrates good calibration early on, as indicated by higher confidence margins for correct predictions compared to incorrect predictions, regardless of the subset size.



(a) Training set.



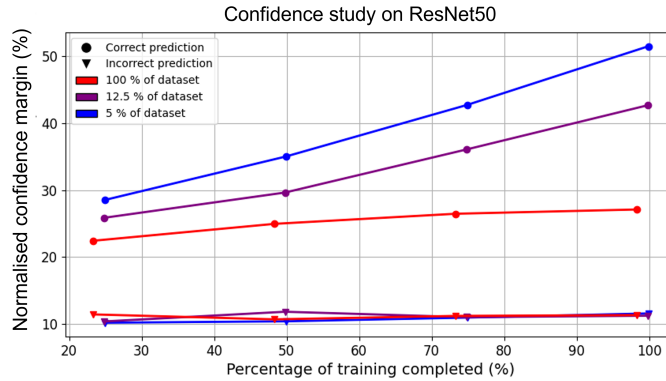
(b) Validation set.

Figure 20: MobileNetV2 normalised confidence margins for different sizes of subset of ImageNet

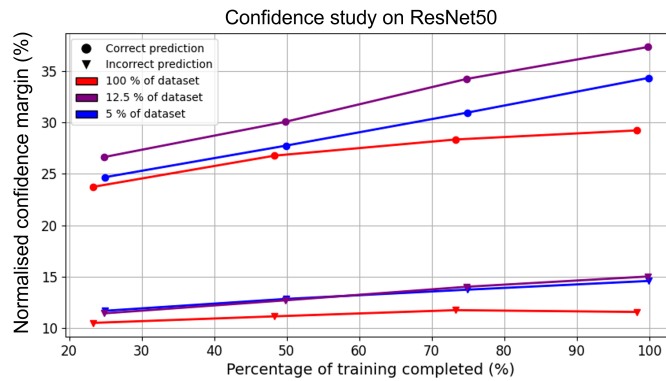
On the validation set, depicted in figure 20b, all differently sized subsets exhibit a similar pattern, with the confidence margins for correct predictions generally following a consistent trend throughout training. Additionally, it is observed that smaller data sets tend to trail slightly behind in terms of confidence margin for correct predictions as training progresses. However, on the training set, in figure 20a, the 5% subset stands out with a notably larger normalised confidence margin than any other subset for correct predictions.

In the case of ResNet50, we find the graphs presenting the normalised confidence margin in figure 21. A similar trend to that found in MobileNetV2 can be observed in the analysis of confidence margins for both correct and incorrect predictions, indicating good calibration of the network and that the network is generally more confident in its correct predictions than its incorrect ones.

On the training set in figure 21a, smaller subsets of ImageNet display a higher confidence margin for correct predictions, while the full dataset shows a comparatively lower confidence margin. Moreover, this difference in confidence margin between smaller subsets and the full dataset becomes more pronounced as training progresses. This suggests that the network finds the smaller subsets easier to learn from, leading to higher confidence in correct predictions during training.



(a) Training set.



(b) Validation set.

Figure 21: ResNet50 normalised confidence margins for different sizes of subset of ImageNet

In contrast, on the validation set depicted in figure 21b, the 12.5% subset performs better than the other subsets, with the 5% subset trailing behind. We recognise there might be hidden complexities in the architectures chosen or in the specific composition of the subset, as there appears to be no direct relation between increase of data resulting in higher confidence margins for ResNet50.

6 Conclusions

To conclude, our study on the impact of data volume on image classification using ResNet50 and MobileNetV2 on the ImageNet dataset has yielded several important findings. First, we observed that both networks exhibited a positive correlation between training data volume and accuracy. As the amount of training data increased, both Top-1 and Top-5 accuracies improved. However, we also identified a saturation point, beyond which increasing the data amount has no effect in terms of accuracy improvement. This suggests that there is an optimal lower threshold of data volume that is necessary to achieve satisfactory performance. In addition, observing that the accuracy improvement after doubling the amount of data once is always smaller than when you double it a second time, the possibility to half the current dataset to get an upper bound of the accuracy improvement would allow for more informed decision making in terms of data collection.

Furthermore, our comparison between ResNet50 and MobileNetV2 revealed differences in their behaviour with respect to data volume. ResNet50 demonstrated a higher efficiency in utilising training data, as it required a smaller data amount (12.5%) to approximate the accuracy of the full dataset. In contrast, MobileNetV2 needed a larger proportion of data (75%) to approach the accuracy achieved with the full dataset. This discrepancy can be attributed to the architectural differences between the two networks, indicating that the network design and complexity can influence the amount of data needed for convergence.

Although we anticipated that overfitting might occur when using smaller subsets of data, we found that the regularisation techniques employed in our training script effectively mitigated overfitting, even in the smaller datasets. Interestingly, we observed that regularisation techniques had a more pronounced effect on the performance of MobileNetV2, suggesting that its architecture may be more susceptible to overfitting without proper regularisation. Additionally, we observed that the smaller subsets of data reached convergence earlier than the larger subsets for MobileNetV2, but at lower accuracy values.

Examining the confidence margins in both networks, we consistently found that correct predictions exhibited higher confidence values than incorrect predictions across all subsets, indicating that the confidence can be a useful indication of prediction correctness, although some calibration would be necessary. Furthermore, the confidence margins for correct predictions exhibited a consistent upward trend throughout training, while incorrect predictions experienced slower or stagnant growth. Notably, smaller data subsets showed increased confidence values for correct predictions in the training set, with ResNet50 displaying a more significant effect compared to MobileNetV2. However, no definitive conclusions can be drawn regarding the influence of data volume on confidence margins, without further comprehensive testing and analysis.

In summary, our study highlights the importance of data volume in training neural networks for image classification tasks. We have demonstrated that increasing the amount of training data can lead to significant improvement in accuracy, but there is a point of diminishing returns which allows the definition of a threshold for how much data is enough for a given network architecture.

7 Future work

The accomplished experiments have focused on the study of image based networks and the impact data volume has on prediction accuracy. However, there are several avenues for future research that can build upon these findings and further enhance our understanding.

Multiple regularisation techniques kept our results from showing overfitting for the smaller subsets of data. It would be interesting to eliminate these features from the training script and analyse whether overfitting truly appeared as predicted, and whether its effects were also quantifiable.

Further exploration could be conducted on other architectures, such as analysing the effect of data volume on networks such as DenseNet, InceptionNet or attention based architectures such as the Transformer Vision architecture, which would provide a deeper understanding on how different architectures respond to varying data volume.

In this study, a single dataset has been used. Experimentation with other datasets, with smaller amounts of classes and smaller amounts of data per class would also be a relevant area of research, given that real-world applications often involve diverse datasets with varying characteristics. By examining how the performance of neural networks is affected by datasets with different class sizes and distributions, we can gain insights into the generalisability and robustness of the results. Analysing different data distributions across classes and whether an imbalance can be mitigated so long as there is enough data, is also a possible avenue of research.

Expanding the study to tasks beyond image classification is also very relevant for future work. Investigating how the volume and characteristics of training data impact the accuracy and generalisability of neural networks such as natural language processing, or one of the many machine learning algorithms used in autonomous driving (object recognition, segmentation, regression, etc.) would provide valuable information on data volume requirements in fields where data acquisition is costly.

By broadening the scope of investigation, future research can contribute to a more comprehensive understanding of the influence of data volume and characteristics on the performance of neural networks across diverse domains and applications. This will enable researchers and practitioners to make informed decisions and develop effective strategies for training neural networks in various real-world scenarios.

References

- [1] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: Review, opportunities and challenges, Nov 2018.
- [2] Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. A survey on recent approaches for natural language processing in low-resource scenarios, 2021.
- [3] Tamás Matuszka and Dániel Kozma. A novel neural network training method for autonomous driving using semi-pseudo-labels and 3d data augmentations. In *Intelligent Information and Database Systems*, pages 216–229. Springer Nature Switzerland, 2022.
- [4] Andrew Ng and DeepLearningAI. A chat with andrew on mlops: From model-centric to data-centric ai, 2021.
- [5] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.
- [6] Terrence J. Sejnowski. *The Deep Learning Revolution*. The MIT Press, 2018.
- [7] Bernhard Mehlig. *Machine Learning with Neural Networks*. Cambridge University Press, oct 2021.
- [8] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [9] Example of a neural network’s neural unit. *Wikipedia Commons*.
- [10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.
- [12] Amir Hossein Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. A comprehensive survey on activation functions in neural networks. *Neural Networks*, 122:287–316, 2020.
- [13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [14] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.
- [15] Tomas Per Rolf Bjoerklund. License plate recognition using convolutional neural networks trained on synthetic images. 2018.
- [16] Ethem Alpaydin. *Introduction to machine learning*. The MIT Press, 2020.
- [17] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

- [18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [20] Andreas Maier, Christopher Syben, Tobias Lasser, and Christian Riess. A gentle introduction to deep learning in medical image processing. *Zeitschrift für Medizinische Physik*, 29(2):86–101, 2019. Special Issue: Deep Learning in Medical Physics.
- [21] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.
- [22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.
- [25] 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.
- [26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.
- [27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.
- [28] Gorlapraveen123. Residual neural network, Oct 2021.
- [29] 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.
- [30] Christiane Fellbaum. Wordnet: A lexical database for english. <https://wordnet.princeton.edu/>, 1998.
- [31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Ilsvrc 2015. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.
- [32] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91. PMLR, 23–24 Feb 2018.
- [33] Tusonggao. Create imagenet train subset 100k, Jan 2023.

- [34] Vasilis Vryniotis. How to train state-of-the-art models using torchvision’s latest primitives.
- [35] Samuel G. Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation, 2021.
- [36] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.
- [37] Sangdoon Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.
- [38] TorchVision maintainers and contributors. Torchvision: Pytorch’s computer vision library. <https://github.com/pytorch/vision>, 2016.
- [39]
- [40] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview, 2020.
- [41] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks, 2020.
- [42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

A Appendix

A.1 Subset maker code

Below we present the code used to create subsets of ImageNet, modified from [33].

```
1 import os
2 import time
3 import random
4
5 train_path = '../tiny-imagenet-server-vol-1/ILSVRC/Data/CLS-LOC/train/'
6 new_train_path = '../tiny-imagenet-server-vol-1/ILSVRC/Data/CLS-LOC/train_25/'
7
8 os.system(f'rm -rf {new_train_path}')
9 os.system(f'mkdir {new_train_path}')
10
11 filenames = []
12
13 dirs = [d for d in os.listdir(train_path) if os.path.isdir(os.path.join(train_path
14 , d))]
15 print('len(dirs): ', len(dirs))
16
17 def get_all_files(directory):
18     filenames = []
19     class_name = directory.split('/')[-1]
20     for filename in os.listdir(directory):
21         if os.path.isfile(os.path.join(directory, filename)):
22             filenames.append(filename)
23             part_name = filename.split('_')[0]
24             assert part_name == class_name
25     return filenames
26
27 process_start_t = time.time()
28 part_start_t = time.time()
29
30 percentage = 0.25 #Percentage of data set
31
32 for i, directory in enumerate(dirs):
33     directory_path = train_path + '/' + directory
34     filenames = get_all_files(directory_path)
35     number_per_class = round(len(filenames) * percentage)
36     #print("Len filenames is ", len(filenames))
37     #print("number_per_class is ", number_per_class)
38     filenames = random.sample(filenames, number_per_class)
39     assert len(filenames) == number_per_class
40
41     os.system(f"mkdir {new_train_path + '/' + directory}")
42
43     for filename in filenames:
44         src_file_path = train_path + '/' + directory + '/' + filename
45         tgt_file_path = new_train_path + '/' + directory + '/' + filename
46         os.system(f'cp {src_file_path} {tgt_file_path}')
47
48     if (i+1) % 100 == 0:
49         print(f'directory: {directory} filenames: {filenames[:3]}')
50         print(f'now i: {i+1}, cost time: {time.time()-part_start_t:.2f} sec')
51         part_start_t = time.time()
```

```

52 #     if i > 50:
53 #         break
54
55 print(f'finished, total cost time: {time.time()-process_start_t:.2f} sec')

```

A.2 Command prompts

This section provides a comprehensive list of command prompts used for training the model, aiming to ensure the reproducibility of the results obtained. By including all the parameters and configurations used during the training process, experiments can be accurately replicated the experiments and consistent outcomes can be achieved.

A.2.1 MobileNetV2

```

1 torchrun --nproc_per_node=4 train.py --model mobilenet_v2 --epochs 90 --lr 0.045
  --wd 0.00004 --lr-step-size 1 --lr-gamma 0.98 --output-dir 'MobilenetV2_100' |
  tee MobilenetV2_100/results_log.txt

```

For all trainings parameters have remained constant except for epochs and lr-step-size, which have been adapted according to the parameters described in table 6.

A.2.2 ResNet50

```

1
2 torchrun --nproc_per_node=4 train.py --model resnet50 --batch-size 128 --lr 0.5 --
  lr-scheduler 'cosineannealinglr' --lr-warmup-epochs 5 --lr-warmup-method '
  linear' --lr-warmup-decay 0.01 --weight-decay 2e-5 --mixup-alpha 1.0 --cutmix-
  alpha 1.0 --label-smoothing 0.1 --auto-augment 'ta_wide' --random-erase 0.1 --
  ra-reps 4 --output-dir 'Resnet50_100' --epochs 60 --lr-step-size 30 | tee
  Resnet50_100/results_log.txt

```

For all trainings parameters have remained constant except for epochs, lr-warmup and lr-step-size, which have been adapted according to the parameters described in table 6.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden

www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY