BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

A REDUCED NEURAL NETWORK FOR CLASSIFYING THE PRESENCE OF PEOPLE IN AN IMAGE

REDUKOVANÁ NEURONOVÁ SÍŤ PRO KLASIFIKACI PŘÍTOMNOSTI POSTAV V OBRAZE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **RASTISLAV SAMUEL STANČEK**
AUTOR PRÁCE

SUPERVISOR **Ing. TOMÁŠ GOLDMANN**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

Institut:	Department of Intelligent Systems (DITS)	156893
Student:	**Stanček Rastislav Samuel**	
Programme:	Information Technology	
Title:	**A Reduced Neural Network for Classifying the Presence of People in an Image**	
Category:	Artificial Intelligence	
Academic year:	2023/24	

Assignment:

1. Become familiar with the problem of detecting people in an image.
2. Summarize available people detection algorithms and learn about the technique of knowledge distillation in neural networks.
3. Using the knowledge distillation technique, create a small neural network model that will be used for binary classification of the presence of people in an image.
4. Design and implement a simple application using the Python programming language to demonstrate the functionality of your neural network.
5. Conduct experiments to assess the accuracy of your solution.

Literature:

- GOU, Jianping, et al. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021, 129: 1789-1819.
- ALKHULAIFI, Abdolmaged; ALSAHLI, Fahad; AHMAD, Irfan. Knowledge distillation in deep learning and its applications. *PeerJ Computer Science*, 2021, 7: e474.
- PUTRA, Muhammad Hadi, et al. Convolutional neural network for person detection using yolo framework. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 2017, 9.2-13: 1-5.

Requirements for the semestral defence:
Aims 1 a 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Goldmann Tomáš, Ing.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	6.11.2023

Abstract

This thesis focuses on the topic of computer vision, more specifically, on classifying people's presence in image data. The goal is to create a reduced neural network utilizing knowledge distillation. Object classification and detection is a computationally an expensive operation. A student model created utilizing knowledge distillation shows equivalent accuracy while being smaller and having better inferencing speed compared to the teacher model. Such model can be interdisciplinarily utilized on end devices having relatively low computational capabilities.

Abstrakt

Táto práca sa zameriava na tému počítačového videnia, presnejšie, na binárnu klasifikáciu prítomnosti ľudí v obrazových dátach. Cieľom tejto práce je vytvoriť redukovanú neurónovú sieť s využitím metódy knowledge distillation. Klasifikácia a detekcia objektov je výpočtovo náročná operácia. Študentský model vytvorený pomocou knowledge distillation vykazuje ekvivalentnú presnosť, pričom je menší a má vyššiu inferenčnú rýchlosť v porovnaní s učiteľským modelom. Takýto model môže byť interdisciplinárne všestranný a to predovšetkým na koncových zariadeniach, ktoré majú relatívne slabé výpočtové schopnosti.

Keywords

knowledge distillation, model compression, object classification, people detection, deep learning, convolutional neural network, reduced neural network, image processing, student model, teacher model, python, tensorflow, keras

Kľúčové slová

knowledge distillation, kompresia modelu, klasifikácia objektov, detekcia ľudí, hlboké učenie, konvolučná neurónová sieť, redukovaná neurónová sieť, spracovanie obrazu, študentský model, učiteľský model, python, tensorflow, keras

Reference

STANČEK, Rastislav Samuel. *A Reduced Neural Network for Classifying the Presence of People in an Image.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Tomáš Goldmann

Rozšírený abstrakt

Moderná doba sa stala takmer synonymná s pojmom umelá inteligencia. Umelá inteligencia zastrešuje nespočetné množstvo odvetví, medzi ktoré patrí počítačové videnie. Prirodzene, počítačové videnie zastrešuje množstvo ďalších odvetví, v neposlednej rade detekciu a klasifikáciu objektov v obrazových dátach.

Posledné desaťročia zažili revolúciu vo vývoji nových lepších hardvérových prostriedkov čo umožnilo vytvárať väčšie a presnejšie trénovateľné modely tvorené neurónovými sieťami. V disciplíne počítačového videnia sú prevalentné konvolučné neurónové siete, schopné spracovať obrazové dáta, ktoré obsahujú veľa informácií. Cez proces konvolúcie sú tieto siete schopné extrahovať základné črty objektov, ktoré sú následne hierarchicky poskladané a tak dokáže sieť identifikovať objekt, v prípade objektových detektorov aj daný objekt lokalizovať.

Proces trénovania sa dá stručne zhrnúť ako schopnosť siete učiť sa z vlastných chýb. Toto je dosiahnuteľné vďaka stratovým funkciám, ktoré sú schopné vyčísliť odchýlku predpovedí modelu od skutočnej pravdivostnej hodnoty.

Človek sa každodenne priamo či nepriamo stretáva s modelmi, ktoré dokážu vykonávať úlohu klasifikácie a detekcie. Takéto modely môžu byť nasadené v rôznych zariadeniach, ako napríklad v mobilných telefónoch alebo kamerách. Toto však prináša výzvu v tom, že takéto zariadenia nemajú takú vysokú výpočetnú silu v porovnaní s výkonnými počítačmi. V takom prípade je potrebné modely zmenšiť a zefektívniť, aby boli schopné fungovať na slabších zariadeniach. Toto je možné dosiahnuť využitím rôznych prístupov známych ako Kompresia Modelov, medzi ktoré patrí proces Knowledge Distillation, voľne preložiteľný ako destilácia vedomostí, na ktorý je táto práca zameraná.

Tento proces pozostáva z využitia predpovedí predtrénovaného modelu, ktorý figuruje ako učiteľ počas procesu trénovania, aby slúžili ako zdroj informácii pre trénovaný model, ktorý figuruje ako študent. Týmto spôsobom je študent schopný napodobniť chovanie učiteľa, no dokonca v niektorých prípadoch aj vylepšiť chovanie učiteľa. Tento proces je spravidla efektívny pri trénovaní študentského modelu, ktorý by pri zvyčajných trénovacích podmienkach nebol schopný dosiahnuť použiteľnú presnosť. Trpezlivý učiteľ je dobrý učiteľ, čo sa prenieslo do paradigmy trénovania modelov za využitia knowledge distillation. Takéto trénovacie procesy trvajú rádovo tisíce epoch, v porovnaní s bežnými procesmi, ktoré sa pohybujú v desiatkach. V rámci knowledge distillation procesu sa využíva hyper parameter zvaný temperature, a síce, teplota, ktorý pomáha so zjemnením pravdepodobnostných distribúcií z predikcií modelov, čo vo výsledku umožnuje študentovy lepšie extrahovať vedomosti a naučiť sa z nich.

Ako učiteľský model bol zvolený populárny model RetinaNet, trénovaný na dátovej sade PascalVOC, ktorého klasifikovateľné triedy zahŕňajú triedu typu osoba. Ako študentský model bol zvolený model patriaci do rodiny modelov EfficientNetV2. Tieto modely sú vylepšením prvej generácie a poskytujú vyššiu efektivitu počas trénovania a následne aj počas používania. Trénovanie prebehlo na dátovej sade Human Detection Dataset, ktorá pozostávala z obrázkov dvoch tried, a síce, tých ktoré obsahovali, alebo neobsahovali osobu. Keďže študent bude schopný klasifikovať len 2 triedy oproti učiteľovi, ktorý ich dokáže klasifikovať 20, bolo potrebné učiteľov výstup rozumne vymaskovať a upraviť tak, aby nenastala strata vo vedomostiach alebo k ich poškvrneniu.

Trénovací proces študenta bol testovaný v rôznych konfiguráciách. Prvotne bol študent trénovaný bez pomoci učiteľa, no tento prístup zlyhal, nakoľko študent nebol schopný dostatočne generalizovať problém a miesto zvýšenia presnosti pri dlhšom trénovaciom pláne sa nadmerne prispôsobil dátovej sade, čo spôsobilo značnú nepresnosť na dátach, ktoré

nepatrili do dátovej sady. Ďalší pokus spočíval v trénovaní študenta len pomocou predikcií učiteľa počas veľmi dlhého trénovacieho plánu. Tento proces tiež zlyhal, pričom študent nebol schopný získať dostatočnú presnosť. Tento fakt môže byť atribuovaný k relatívne nízkej presnosti učiteľského modelu. Prístup ktorý bol eventuálne implementovaný a úspešný spojil tieto 2 prístupy. Toto spôsobilo, že študent konvergoval k určitému výsledku rýchlejšie a dokonca pomohol študentovi zistiť nepresnosti v pravdivostných hodnotách dátovej sady, ktoré boli následne opravené.

Výstup tejto práce poskytuje redukovaný model, ktorý je schopný binárne klasifikovať prítomnosť, alebo neprítomnosť osoby v obrazových dátach.

A Reduced Neural Network for Classifying the Presence of People in an Image

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Tomáš Goldmann. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Rastislav Samuel Stanček
May 16, 2024

Acknowledgements

Firstly, I would like to express my sincerest gratitude to my supervisor, Ing. Tomáš Goldmann. His patience, understanding as well as his feedback and guidance proved to be an invaluable resource throughout this thesis. My utmost gratitude also extends towards my family and close friends. I find it hard to imagine that I would be where I currently am without their never-ending and loving support.
Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Contents

Chapter 1

Introduction

In the ever evolving world of Artificial Intelligence (AI), we have grown accustom to having AI included in various ways into our daily workflows, whether it utilizes a model capable of processing human language and generating a coherent text response in Natural Language Processing, or classifying objects within images in Computer Vision (CV). Naturally, most models in these fields are some sort a deep neural network, trained using Deep Learning. These models constantly improve in accuracy, speed and features, but also proportionally increase in size, complexity and computational power requirements. Subsequently, they can be difficult and computationally expensive to deploy.

There are many models based on various techniques in CV for classifying objects. This thesis will focus on analyzing and comparing models capable of classifying and detecting objects such as You Only Look Once (YOLO), Region-based Convolutional Neural Networks (R-CNN) or Single-Shot Detector (SDD) or RetinaNet and produce a reduced and simpler model retaining the accuracy of the teacher model, in some cases possibly even surpassing the teacher.

Empirically speaking, after inventing or creating something new and complete, the natural way to improve it, is to optimize, make it faster, more efficient and in many cases, smaller. In machine learning terms, to make a model smaller is to utilize a technique called Model Compression. This thesis will delve deeper into a particular concept of Model Compression called Knowledge Distillation. Making models smaller can help in deployment, particularly on end devices that are easier to distribute.

Knowledge Distillation (KD) generally consists of taking a large, accurate teacher model and using it to train a student model. The student model, retains comparable performance while being quite simpler in structure, having less layers and parameters that is. It can also be trained to mimic a specific subset of behaviors of the teacher model. In terms of CV models, to only classify a particular class or classes of objects, in case of this thesis a single class, a person.

In the beginning, basic principles behind Deep Neural Networks, their types and training process will be discussed. Furthermore, the chapter will include the discipline of Object Detection and Classifications and discuss the various challenges and approaches within it. The following chapter will focus on particular algorithms and approaches existing in Computer Vision used to classify and detect objects in images. The subsequent chapter will delve deeper into the concept of Knowledge Distillation, explain the process and various possible configurations. The last two chapters will describe the practical implementation of this thesis in great detail and eventually compare and present the obtained results.

Chapter 2

Neural Networks and Person Detection

This chapter will discuss the basis of what an *Artificial Neural Network* is and how it works, what does it consist of and how can it be utilized in *Computer Vision* (CV). Finally, it will discuss the topic of *Person Detection* in CV, problems in CV, historical approaches to solving these problems and state of the art models and algorithms tackling complex tasks.

2.1 Neural Networks

Pattern recognition in general is a natural component of human intellect. Humans are able to traverse new and unknown environments thanks to years of evolution and learning. As problems and tasks grew in complexity, people devised various ways to aid them in solving these problems. Among many approaches to solutions to these problems lied the *Artificial Intelligence*. Although AI on its own is a powerful tool, it is not the „one-size-fits-all“ solution, since not every problem can be described with a finite set of rules and constraints. In comparison, humans are not bound by some rule set to create a decision, thanks to their empirical knowledge gained by learning. Consequently, it had to be made possible for computers to learn and teach them selves through means of *Machine Learning* (ML).

Nowadays, one of the most popular branches of ML are *Artificial Neural Networks* (ANN). Similarly to a human brain, ANNs consist of interconnected neurons and are able to learn and adapt to solve a generalized problem effectively. The reason why ANNs are so effective at solving problems is the vast parallelism of all the neurons throughout ANN's layers.

Building blocks

As human brains consist of neurons, ANNs [8] consist of *Artificial Neurons* (AN). The simplest representation of an AN is a *Perceptron*. Comparably to a natural neuron, that has a set of *dendrites* as inputs and a single output as *axon* a perceptron has a set of inputs with one output.

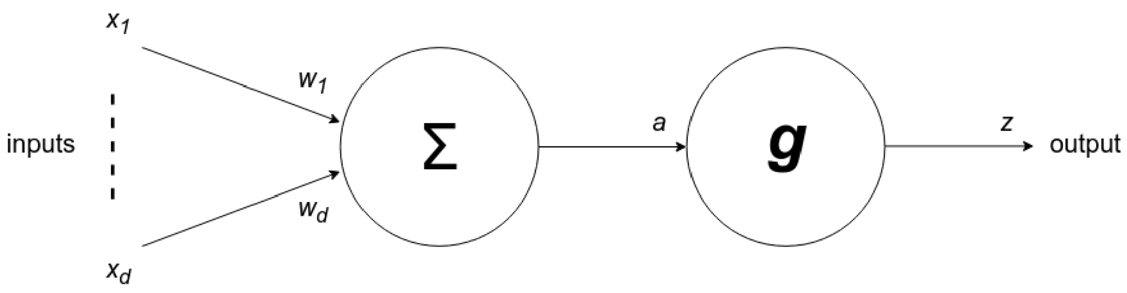


Figure 2.1: The McCulloch-Pitts model of a perceptron

The perceptron forms a weighted sum (2.1) of the inputs $x_1, \ldots, x_d$ and then transforms this sum using an activation function (2.2) $g()$ to give a final output of $z = g(a)$

$$a = \sum_{i=0}^{d} w_i x_i \tag{2.1}$$

The activation function [26] decides whether the perceptron, or *node* should be activated or not. If there were no activation function, the ANN would essentially be just a linear regression model. The activation function introduces the non-linearity making the ANN capable to learn and solve more complex problems.

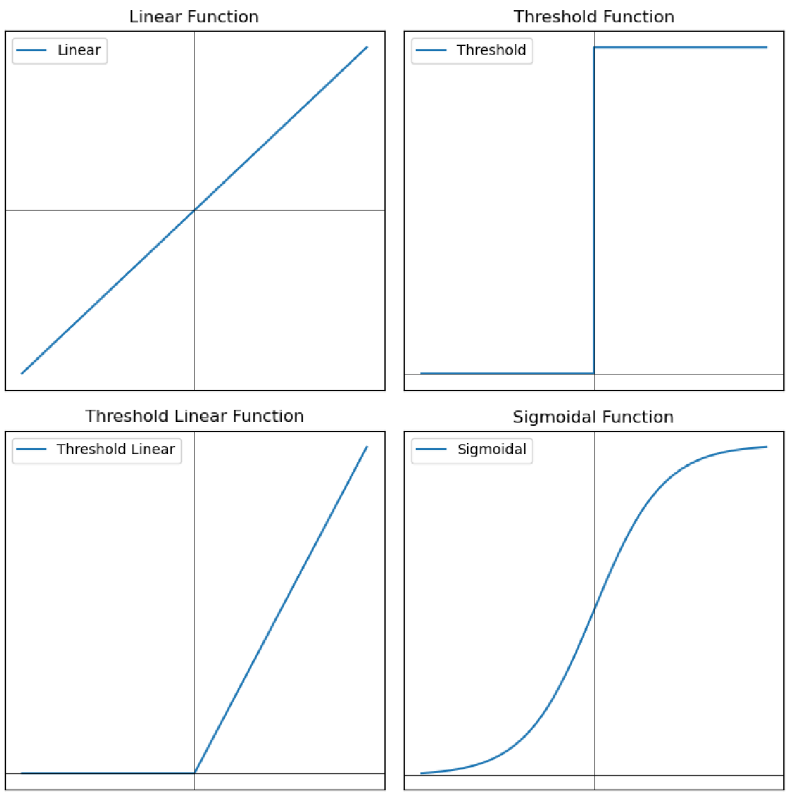


Figure 2.2: A selection of typical activation functions

Neuron Connection

When multiple biological neurons join, they form a *synapse*. Analogically, the joining of artificial neurons form a *layer*.

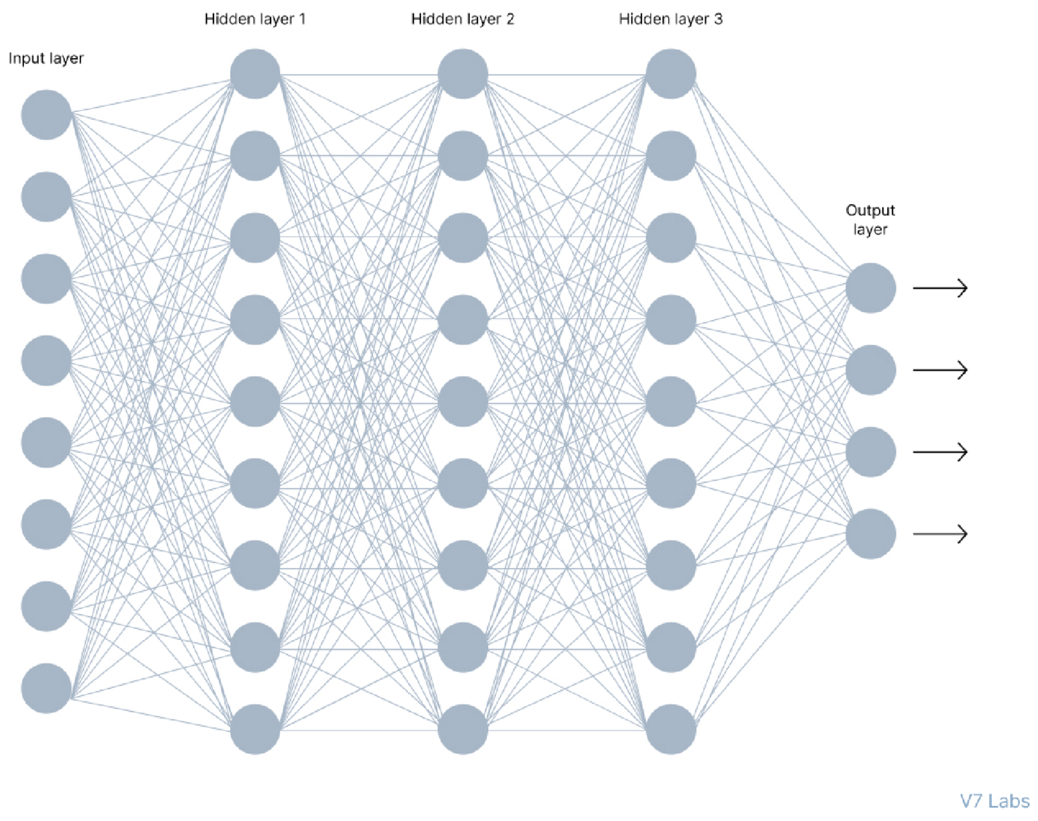


Figure 2.3: Interconnected neurons forming layers of the neural network. Image available at [26]

Artificial Neural Networks typically have an *Input layer*, *0 to N Hidden layers* and an *Output layer*. For an ANN to be considered a *deep neural network*, it needs to have at least 2 hidden layers.

Input Layer

The input layer [25] is used to communicate with the outer environment. Once an input is introduced to the input layer of the network, it proceeds to transfer it to the hidden layers to be processed. It also defines the conditions on which the network training depends.

Hidden layer

Hidden layer(s) is an intermediate layer between the input and the output layer of the network. Number of hidden layers within the network changes from problem to problem.

Selection of hidden layers is a complex task as in some cases due to the number of hidden layers a condition known as *overfitting* and *underfitting* occurs (2.2), which negatively impacts the efficiency, accuracy and time complexity of the network.

Output layer

The output layer is connected with the outer environment and represents the output of the input given to the network. This last layer attempts to produce a class score based on activations from previous layers.

Types of Neural Networks

After the discovery of a simplest form of a neural network, the Multi Layer Perceptron, many other types were created to better suit particular problems. Most prevalent type of neural network in computer vision is a Convolutional Neural Network [2, 19] (CNN)

When it comes to image data, an ordinary ANN starts to struggle due to the computational intensity of image processing. It would take 784 weights per neuron in the first hidden layer to process a 28×28 black and white image. In comparison, Convolutional Neural Networks bring a significant reduction in required weights an parameters thanks to so called *max-pooling* layers. In practice, the input (color) image with a dimensionality of $28 \times 28 \times 3$ would lead into a final output layer with a dimensionality of $1 \times 1 \times N$, where N represents the number of classes.

CNNs consist of three types of layers. The *Convolutional layers*, *Pooling layers* and *Fully-connected* layers (2.4).

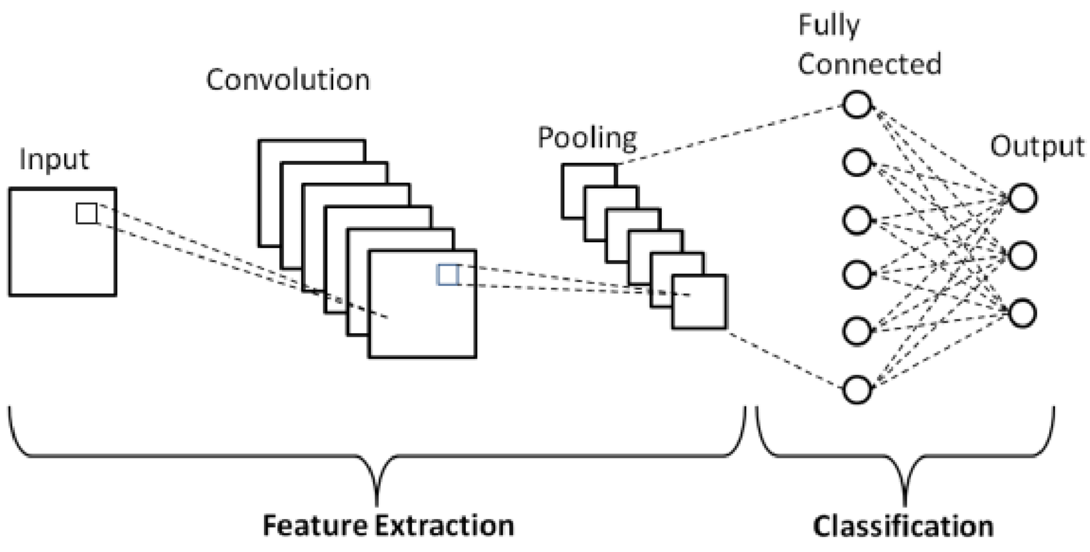


Figure 2.4: This image depicts the separate layers of a simple Convolutional Neural Network. Image available at [20]

Convolutional Layer

The layer's parameters focus around the use of learnable *kernels*. As the kernel glides through the input, the scalar product is calculated for each value in that kernel. From this the network will learn the kernels that activate when they see a specific feature at a given spatial position of the input. Every kernel will have a corresponding activation map. Convolutional layers are also able to significantly reduce complexity of the model through the optimisation of their output through so called *hyperparameters*: depth, stride and zero-padding.

Pooling Layer

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number parameters and the computational complexity of the model. The pooling layer operates over each activation map in the input. In most CNNs these come in the form of max-pooling layers with kernels of a dimensionality of 2×2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size while maintaining the depth volume to its standard size. Due to the destructive nature of this layer, having a kernel size greater than 3 will usually greatly decrease the performance of the model.

Fully-connected Layer

The fully-connected layer contains neurons which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. This is analogous to way that neurons are arranged in traditional forms of ANN.

2.2 Deep Learning

The greatest strength of neural networks lies in their adaptability. The same architecture can be used and adapted to suit multiple use cases. For example, in context of this thesis, the same model of a convolutional neural network can be used to classify hand written digits, or detect a person in an image. The desired result can be achieved through a process called *learning* (in the case of deep neural networks *deep learning*), more specifically *Supervised Learning*.

In simplified terms [9], the supervised learning process consists of feeding input data[1] to a model with initialized weights with random values through *forward propagation*. Afterwards, the model's predicted value has to be compared to actual target value for the given input, using so called *loss function*. Through the use of *backward error propagation*, individual node weights are adjusted according to the cost function value.

This process can be iteratively repeated until the desired accuracy is reached or a terminating condition is met. Each iteration is called an *epoch*.

[1]Input data consist of 2 sets of labeled dataset, the training set and the validation set

Underfitting and Overfitting

During the model's learning process, two major accuracy decreasing issues can occur [5]:

Underfitting occurs when the model is not able to obtain a sufficiently low error value on both the training and validating set. It usually means, that either the model is too simple or not trained for enough epochs.

Overfitting occurs when the gap between the training error and test error is to large. It performs well on the training set, but fails to perform with the validation set. It usually means that the model is too complex or that the training dataset is too small or noisy. It may also happen when the training goes on for too many epochs and is not stopped early.

2.3 Object Detection and Classification

Before we delve deeper into the challenges of Person Detection (2.3), the discipline of Object Detection (OD) needs to be discussed first.

Object detection is in the subset of domains of Computer Vision. This technique involves processing, analysing and extraction information from digital images. This subset also includes topics like scene reconstruction, activity recognition, image restoration and others, though this thesis will only focus on OD. While non machine learning approaches can solve OD tasks, neural networks, predominantly convolutional neural networks have shown to be extremely useful due to their ability to effectively extract features.

Feature Extraction

Feature extraction plays a crucial role in helping various OD algorithms to understand image data. As previously stated, convolutional neural networks excel at this task. Each convolutional layer condenses the input by extracting features of interest and produces *feature maps* in response to different feature detectors [18]. The first convolutional layer begins with simple features, such as edges or points. As data pass through, each layer combines shapes from the previous layer into higher-order shape[2]. The composition of these convolutional layers is called a *feature extraction backbone*. Finally the fully connected layer activates a particular neuron in the output layer. The number of neurons in the output layer correspond with the number of classes that a particular model has been trained to classify.

[2]i.e. multiple edges can form a circle

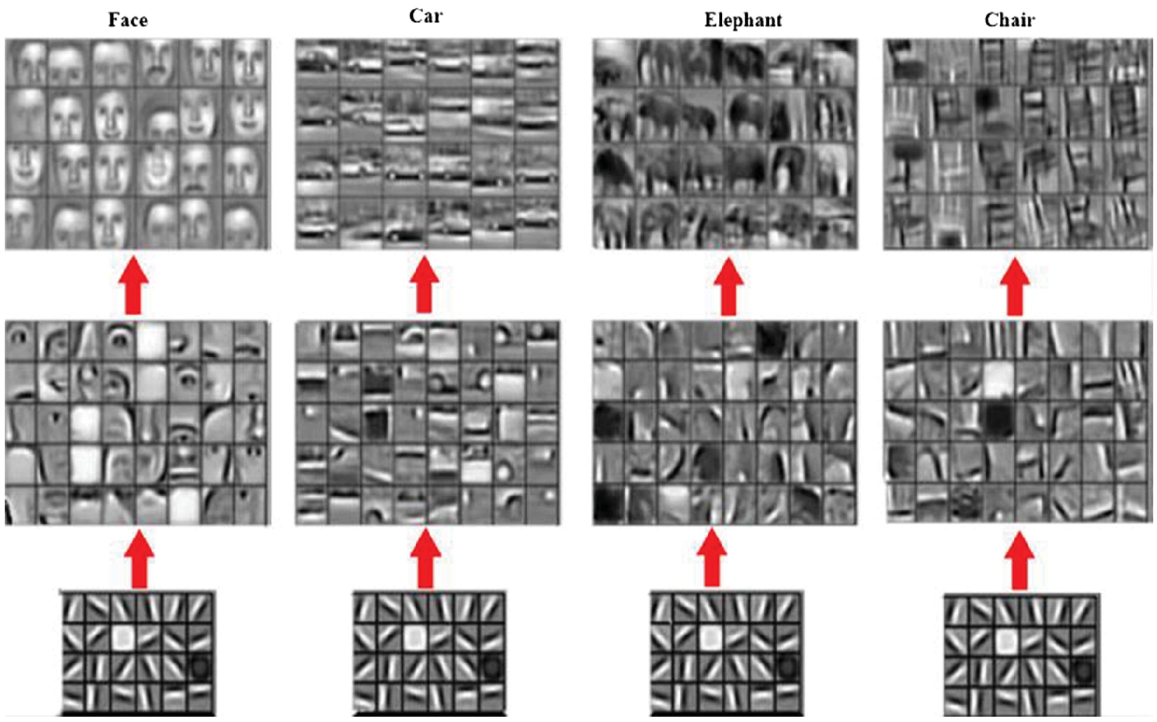


Figure 2.5: This image depicts the feature stacking, from the simplest ones (bottom) to complete higher-order shapes (top). Image available at [35]

Person Detection

As with any other class of object detection, person detection poses various challenges [15]:

- **Variability in appearance** — it is safe to assume that the chance of multiple similarly looking people appearing on one image is quite low. Each person can differ in their clothing or poses
- **Scale and resolution** — varying scales[3] and small resolutions with poor image quality prove to have distorting properties
- **Occlusion** — only a part of a person can be visible in a particular image due to the person being obscured by an object
- **Background complexity** — detecting a person can be made difficult by a busy and complex environments, thus detecting a false positive
- **Pose variation** — including all possible poses and positions that a person in the training dataset can be near impossible
- **Deformable clothing** — not all pieces of clothing follow the standard human silhouette

[3]i.e. person's distance from the camera

To combat these challenges, various approaches exist:

- **Bottom-Up Feature-Based Approaches** — these algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary, and then use them in the detection procedure
- **Top-Down Knowledge-Based Methods** — these rule-based methods encode knowledge of what constitutes a typical human body. These methods are designed mainly for human body localization
- **Template Matching Methods** — several standard patterns of humans or human body parts are used to describe either the human body globally or as distinct human body parts (limbs, face, head etc). The correlations between the input image and the patterns are computed for detection. These methods have been used for both localization and detection with considerable accuracy
- **Appearance-Based Methods** — in contrast to template matching, the models (or templates) are learned from a set of training images, which should capture the representative variability of human appearance. These learned models are then used for detection
- **Integration of Parts detectors** — in contrast to all the techniques described above, this last category fuses the detection results derived by robust part detectors. They are commonly deployed in order to perform reliable approximation of the bodies' shape and extent

An ideal person detecting algorithm should be invariant to the previously stated issues. Though these various approaches exist, not all have the same performance and accuracy as approaches based on neural networks.

Performance Evaluation Metrics

There are two popular metrics to determine the predictive performance and accuracy of different object detection models, **Intersection over Union** and **Average Precision** [27]

Intersection over Union (IOU) serves to assert the localization accuracy of model's prediction compared to ground truth bounding boxes according to this formula

$$IOU = \frac{AoI}{AoU} \tag{2.2}$$

where AoI is the area of intersection[4] and AoU is the area of union[5]. The resulting ratio provides a good estimate of how close the prediction bounding box is to the ground truth.

Average Precision (AP) is calculated as the area under a precision-recall curve[6] for a set of predictions. Recall is calculated as the ratio of the total predictions made by the model under a class with a total of existing labels for the class. Precision refers to the ratio of true positives with respect to the total predictions made by the model. Recall and precision offer a trade-off that is graphically represented into a curve by varying the classification threshold. The area under this precision vs. recall curve gives us the AP per class for the model. The average of this value, taken over all classes, is called mean Average Precision (mAP).

[4]intersection denotes the region where predicted bounding box and ground truth bounding box overlap
[5]union denotes the total region covered by both predicted and ground truth bounding boxes
[6]the curve is created by connecting points representing a particular confidence threshold

Chapter 3

People Detection Algorithms

Many algorithms began their development in the last century. The greatest breakthrough for computer vision has been the *Deep Learning Revolution* in the late 2000s bringing massive performance improvements. This chapter will discuss both algorithms used before this time and after. These algorithms can divided into ones **not reliant on neural networks** and algorithms **based on neural networks**. The latter can be further divided into two categories by the required times the same input image has to pass through the network into **One-Stage/Proposal-Free** algorithms and **Two-Stage/Proposal** algorithms.

3.1 Algorithms not reliant on Neural Networks

Viola-Jones

As discussed in the previous chapter, it can be challenging do determine which feature of the human body can be chosen to be the defining one, although, the human face can be considered as one. This algorithm [29] performs the best on full view, frontal, upright, well lit, full sized faces in fixed-resolution images and is computationally quite inexpensive. For an image to be successfully processed, it has to be correctly rescaled, grayscaled with an increase to contrast and individual pixels normalized into the range of *<0;1>*.

This algorithm relies on extracting certain features from an image, passing them into a cascading decision making algorithm that determines in a binary way, whether the image contains a face or not.

Features utilized by this algorithm can be described as *Haar-like scalable rectangles*[1] divided into 3 categories:

- **two-rectangle features** consisting of 2 rectangles, one black, one white. Useful for detecting edges.
- **three-rectangle features** consisting of 3 rectangles, two are of the opposite color of the remaining one. Useful for detecting straight lines.
- **four-rectangle features** consisting of 4 rectangles in a diagonal configuration, two black and two white. Use for detecting diagonal lines.

While these features are simple in nature and inherently not very accurate, they have a great computational advantage compared to their counterparts[2]

[1]that have the same dimensions and are adjacent

[2]i.e. steerable filters

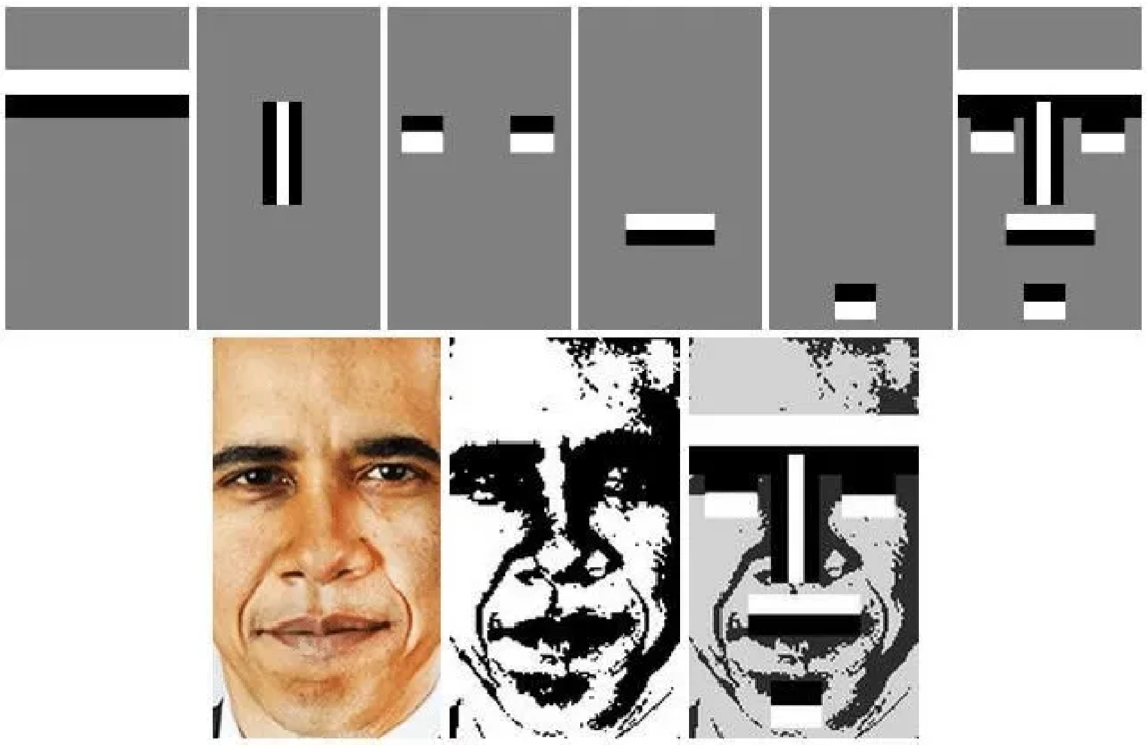


Figure 3.1: Example of Haar-like features matching a preprocessed image. As can be seen, transitions from the forehead to the brow or the eyes to the cheeks have been classified as **Edge-features**. Additionally, the nose follows a black-white-black pattern, thus being classified as **Line-feature**. Image available at [6]

The Attentional Cascade, in other words, the cascade of classifiers is the backbone of this algorithm. Each layer consists of *weak* classifiers utilizing aforementioned features to mark a particular region of an image for further potential processing. Additionally, each of the layers is trained by a boosting algorithm called *AdaBoost*. Each consecutive layer has increasing requirements for *detection rate* and *false positive rate*. This particular trait is responsible for the performance of this algorithm, the reason being, the layer that is higher in the cascade takes the previous layer of weaker classifiers and rejects and „throws out“ the regions falsely marked as positives[3], thus decreasing the number of classifiers that will be processed by the next layer.

3.2 Algorithms Based on Neural Networks

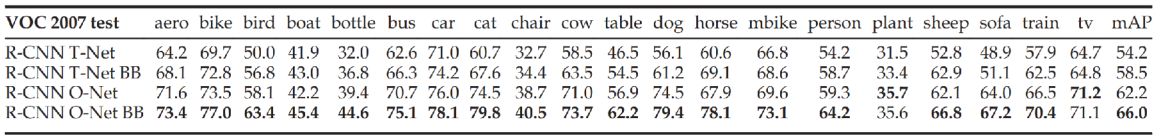
Two-stage or **Two-shot** [27] algorithms will be discussed first. These algorithms require two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than **single-shot**[4] algorithms at the cost of computational requirements.

[3]positive as in containing a relevant feature
[4]or **single-stage**

Region-based Convolutional Neural Network

Region-based Convolutional Network (RCNN) [12] introduced the concept of using deep learning for object detection. It was followed and improved upon by many algorithms (*Fast R-CNN*, *Faster R-CNN*, *RFCN*, *Mask RCNN*), though for the sake of simplicity, only the concept of RCNN will be discussed further. RCNN can be divided into 3 modules. First generates category-independent region proposals. The second is a CNN that extracts a fixed-length feature vector from each region. The third is a set of class-specific linear support vector machines.

The modular design allows for combining and interchanging different implementations. For regional proposal methods like selective search, category-independent object proposals, constrained parametric min-cuts and others can be used, though RCNN is agnostic to the particular region proposal method. Similarly, different CNN architectures can be used in the second module, though these can impact the performance and accuracy of the whole system (3.2).



VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN T-Net	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN T-Net BB	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8	58.5
R-CNN O-Net	71.6	73.5	58.1	42.2	39.4	70.7	76.0	74.5	38.7	71.0	56.9	74.5	67.9	69.6	59.3	**35.7**	62.1	64.0	66.5	**71.2**	62.2
R-CNN O-Net BB	**73.4**	**77.0**	**63.4**	**45.4**	**44.6**	**75.1**	**78.1**	**79.8**	**40.5**	**73.7**	**62.2**	**79.4**	**78.1**	**73.1**	**64.2**	35.6	**66.8**	**67.2**	**70.4**	71.1	**66.0**

Figure 3.2: This figure depicts how different architectures influence the accuracy of RCNN. Image available at [12]

Finally, the **Single-stage** or **Single-shot** [27] algorithms will be discussed. These algorithms require just a single pass of the input image to make predictions about the presence and location of objects in the image. Although, these algorithms are computationally less intensive, they struggle with detecting small objects and are generally less accurate.

Single Shot Detector

Compared to RCNN, Single Shot Detector (SSD) [17, 4] proved to be both fast and accurate, averaging at around 59 frames per second and mAP of 74.3%[5]. SSD builds upon a predefined network (3.3) called the *backbone* and adds auxiliary structures called the *SSD head*. The backbone is used as a feature extractor since the final fully connected classification layer has been removed. The head is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers activations. Instead of using a *sliding window*, SSD divides the image using a grid and have each grid cell be responsible for detecting object in that region of the image.

[5]on the VOC2007 test

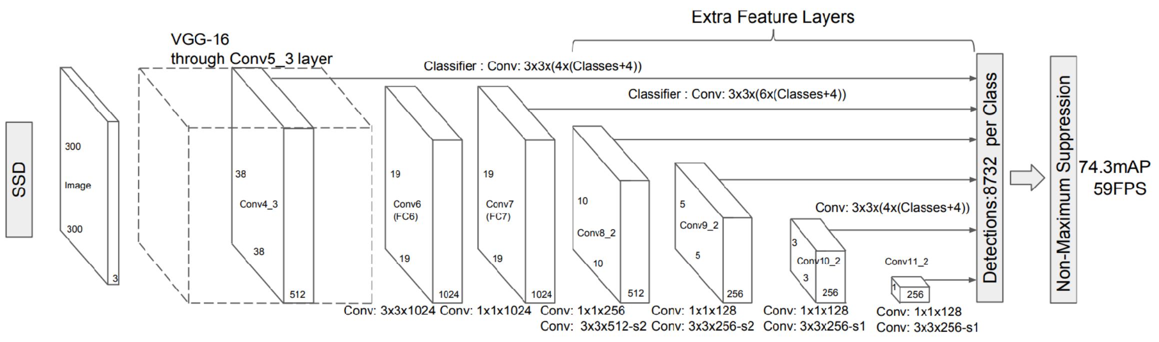


Figure 3.3: The figure depicts SSD architecture including the VGG-16 model. Image available at [17]

You Only Look Once

Despite having many versions increasing in accuracy and efficiency, this thesis will discuss the first iteration of You Only Look Once (YOLO) algorithm for the sake of simplicity [27, 21]. The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates by using the features extracted from the convolutional layers. Similarly to SSD, YOLO divides an input image into a grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. One key technique used in the YOLO models is *non-maximum suppression* (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection by identifying and removing redundant or incorrect bounding boxes.

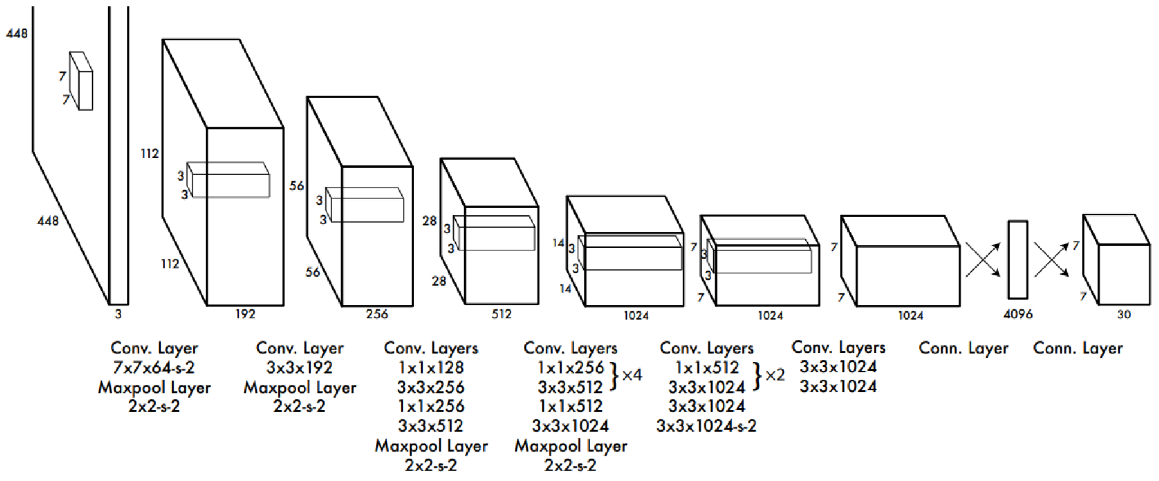


Figure 3.4: This figure depicts the architecture of YOLO model. Image available at [21]

RetinaNet

RetinaNet [16] is a single, unified network analogical to SSD comprising of a backbone network but having two task-specific subnetworks. To further augment the backbone network a Feature Pyramid Network (FPN) is used with top-down pathway and lateral connections so the network efficiently constructs a rich, multi-scale feature pyramid from a single resolution input image. Each level of the FPN can be used for detecting objects at a different scale. RetinaNet uses Anchor Boxes (AB), which are fixed sized boxes to predict the bounding boxes for objects. To achieve that, the model regresses the offset between the location of the object's center and the center of an anchor box, and then uses the width and height of the anchor box to predict a relative scale of the object. Each location on a given feature map has nine anchor boxes. The two aforementioned subnetworks are for classification and bounding box regression on each of the AB.

The classification subnetwork is particularly important in this thesis as it serves as the basis for distilling knowledge to the student. It is a small fully convolutional network attached to each FPN level. Taking an input feature map from a given pyramid level, the subnetwork applies four 3×3 convolutional layers each followed by ReLU activations, followed by a 3×3 convolutional layer with sigmoid activations.

In parallel with the object classification subnetwork, a second fully convolutional network is attached to each pyramid level for the purpose of regressing the offset from each AB to a nearby ground-truth object, if one exists. The design is identical to the classification subnetwork, except it terminates with 4 outputs, as in 4 spacial locations.

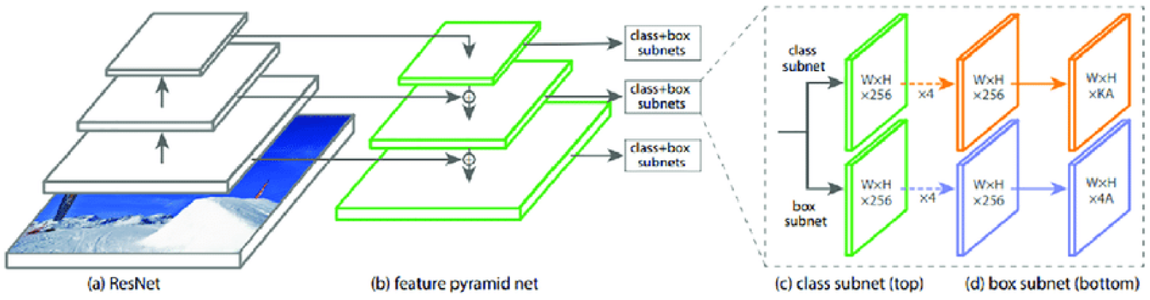


Figure 3.5: This figure depicts the architecture of RetinaNet model. This particular diagram uses ResNet network as the backbone. Image available at [16]

Generally, single-shot object detectors are better suited for real-time applications, while two-shot object detectors are better for applications where accuracy is more important [27, 4].

Chapter 4

Knowledge Distillation

This chapter will focus on the principle and process of *Knowledge Distillation* (KD) [14]. As discussed in previous chapters, deep learning has made an immense impact in the discipline of computer vision, spawning many various implementations of Deep Neural Networks (DNN). In addition to their versatility and performance, they often consist of millions of parameters, thus requiring high computational power. KD can help to alleviate some of these issues. The generic premise of KD is that a *Teacher model* mediates *Knowledge* to a *Student model* from which the student model trains. One important metric that necessitates mentioning is *Knowledge* or *Distillation loss*. This metric measures the difference between the predictions of the student model and the soft targets (4.1) provided by the teacher model.

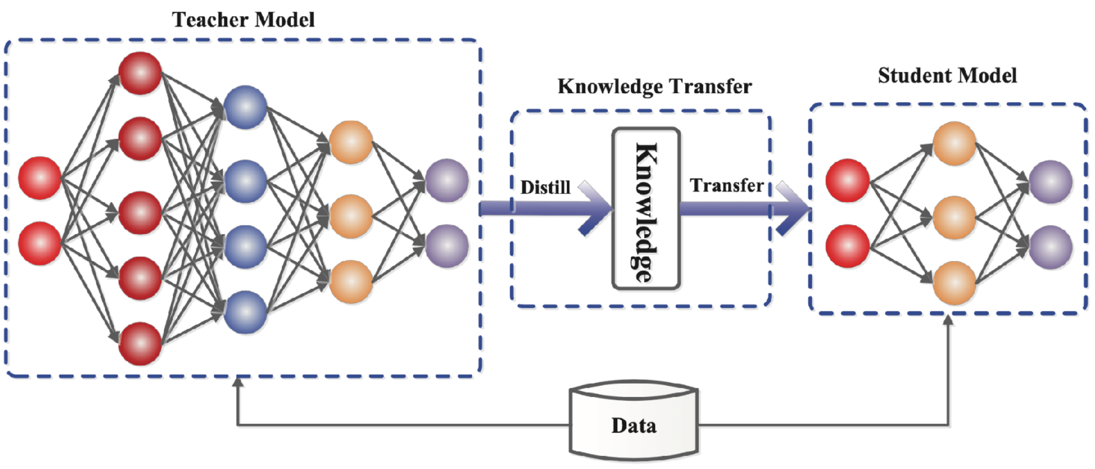


Figure 4.1: This figure shows a generic teacher-student framework for knowledge distillation. Image available at [14]

4.1 Knowledge

There are multiple kinds of knowledge to differentiate from in the process of KD.

Response-Based Knowledge

Response-based knowledge usually refers to the neural response of the last output layer of the teacher model. The main idea is to directly mimic the final prediction of the teacher model. The response-based knowledge distillation is simple yet effective for model compression. The response-based knowledge can be used for different types of model predictions. For example, the response in object detection task may contain the logits[1] together with the offset of a bounding box. The most popular response-based knowledge for object classification in images is known as soft targets. Soft targets are the probabilities that the teacher model outputs for a given input and can be estimated by a softmax function as

$$p(z_i, T) = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{4.1}$$

where z_i is the logit for the i-th class, and a *temperature*[2] factor T is introduced to control the importance of each soft target. However, the response-based knowledge usually relies on the output of the last layer, thus fails to address the intermediate-level supervision from the teacher model, which turns out to be very important for representation learning using very deep neural networks.

Feature-Based Knowledge

Deep neural networks are good at learning multiple levels of feature representation with increasing abstraction. Therefore, both the output of the last layer and the output of intermediate layers can be used as the knowledge to supervise the training of the student model. Specifically, feature-based knowledge from the intermediate layers is a good extension of response-based knowledge, especially for the training of thinner and deeper networks.

4.2 Distillation Schemes

There also are various configurations of teacher to student model while distilling. Moreover, these configurations can be combined to complement each other.

Offline Distillation

In the generic sense of KD, the knowledge is transferred from a pre-trained teacher model into a student model. Therefore, the whole training process has two stages:

- The large **teacher** model is first trained on a set of training samples before distillation.
- The **student** model is used to extract the knowledge in the forms of logits or the intermediate features.

[1]particular raw output of the model
[2]an important **hyperparameter**

though, usually, the first stage in not discussed, since it is assumed that the teacher model is pre-defined.

Little attention is paid to the teacher model structure and its relationship with the student model. Therefore, the offline methods mainly focus on improving different parts of the knowledge transfer, including the design of knowledge and the loss functions for matching features. The main advantage of offline methods is that they are simple and easy to be implemented. The offline distillation methods usually employ one-way knowledge transfer and two-phase training procedure.

Online Distillation

To overcome the limitation of offline distillation, online distillation is proposed to further improve the performance of the student model, especially when a large-capacity high performance teacher model is not available. In online distillation, both the teacher model and the student model are updated simultaneously, and the whole knowledge distillation framework is end-to-end trainable. Co-distillation in parallel trains multiple models with the same architectures and any one model is trained by transferring the knowledge from the other models, thus improving mutual learning.

Self-Distillation

In self-distillation, the same network is used for the teacher and the student model. This can be regarded as a special case of online distillation. Distilling knowledge from deeper sections into shallower section is also a possibility proposed by researchers. Snapshot distillation is a special variant of self-distillation, in which knowledge in the earlier epochs of training is transferred into network's later epochs to support the supervised training process.

4.3 Teacher-Student Architecture

The quality of knowledge acquisition and distillation from teacher to student is determined by the design of the teacher and student networks. Analogically, when we think of humans learning, we hope that a student can find a right teacher. In an incorrect configuration or designs of the student model architecture, a phenomenon called *model capacity gap* can occur, which can degrade knowledge transfer during the distillation process. Another thing mentioning is working with models that do not share the same output specification, as in this thesis. Outputs must be very carefully masked and transformed as not to pollute the knowledge passed from the teacher.

Chapter 5

Solution proposition and Implementation

This chapter focuses on a practical realization of distilling knowledge from the teacher model to a student model. First and foremost, a suitable TM and SM architecture had to be selected. Focusing on object classification, multiple candidates were eligible. A basic object classifier would serve quite well, nevertheless, I opted to use an object detector as it would provide richer knowledge to distill from. The choice was distilled down to two single-stage detectors for their inference speed and relative accuracy. The detectors in question are YOLO and RetinaNet, from which the latter was selected. Since the focus of this thesis lies in training a reduced neural network, a pre-trained TM was selected. For that the Keras_CV library, further discussed in later section, came in very useful. Secondly, a fitting dataset had to be chosen and adapted. Lastly, a scheme for distillation had to be devised and implemented.

5.1 Technologies

This section briefly discusses the major technologies and libraries used to implement the practical aspect of this thesis.

Python

Python [28] is a high-level scripting language with many usecase due to large amount of open source libraries. Although, Python on its own is relatively slow, many of the libraries utilize acceleration on GPUs, thus making it very powerful in the field of machine learning compounded by its readability and relative simplicity. The 3.10.12 version was selected to implement the practical part of this thesis.

TensorFlow / Keras

TensorFlow [1] is a popular open-source machine learning library used mainly to design, train and evaluate neural networks of all kinds. This library can utilize even multi-GPU architectures to provide high performance and acceleration during operations with neural networks. Since version 2.0, TensorFlow integrates the Keras API [10] making it very versatile, since Keras provides very comprehensive tools to create and train models and providing the possibility to interchangeably use features from both libraries within one project. This thesis utilizes TensorFlow and Keras versions 2.15.0.

Keras CV

Keras CV [32] is a horizontal extension of the Keras API that works with TensorFlow, JAX, or PyTorch. This library is focused on computer vision task containing many pre-trained models and meta-architectures with comprehensive tutorials and documentation. Finding a pre-trained teacher model for this thesis would prove to be a challenge were it not for this library. For the implementation, version 0.9.0 was used.

5.2 Teacher model

As previously mentioned, the Keras CV library provides various pre-trained models to choose from, and each of the models have various presets to choose from. I opted to select the RetinaNet 3.2 with the *retinanet_resnet50_pascalvoc* preset [1]. This preset is built upon a Resnet50 v1 backbone trained on the PascalVOC 2012 dataset [2] since other presets were trained on datasets that did not contain a „person“ class. This model achieves a final MaP of 0.33 on the evaluation dataset. Since the PascalVOC 2012 dataset consists of 20 classes a method to mask the model's output to contain the „person“ class probability had to be devised, which will be discussed in a later section.

5.3 Student model

As for the student model the EfficientNetV2B2 [23] network was chosen. In comparison to the ResNet50 backbone used in the teacher model which has top-1 accuracy of 74.9%, top-5 accuracy of 92.1% and 25.6M parameters, EfficientNetV2B2 has top-1 accuracy of 80.1%, top-5 accuracy of 94.9% and 9.2M parameters[3]. For this thesis, the original model's architecture was left unchanged and initialized with two classes, 0 meaning **not containing a person** and 1 meaning **containing a person** and final layer activated by the softmax function.

The key idea behind EfficientNet [23] family of convolutional neural networks is to simultaneously scale the network's depth, width, and resolution. Traditionally, neural network architectures have been scaled by increasing just one of these dimensions, which can lead to sub-optimal performance or computational inefficiency. EfficientNet introduces a compound scaling method that uniformly scales all dimensions of the network, resulting in better performance.

[1]https://keras.io/api/keras_cv/models/tasks/retinanet/#frompreset-method
[2]http://host.robots.ox.ac.uk/pascal/VOC/
[3]Values found at https://keras.io/api/applications/

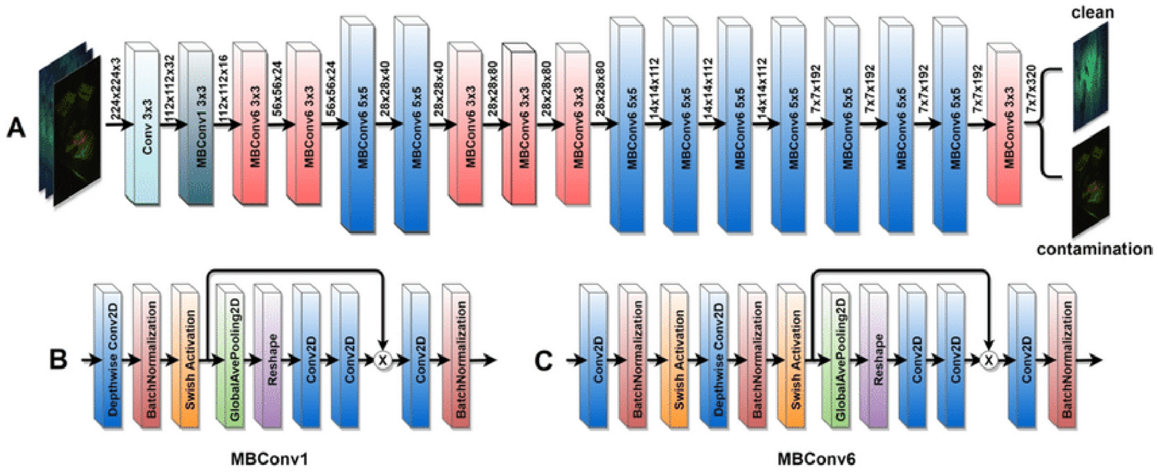


Figure 5.1: This figure depicts a concise representation of the EfficientNet-B0 model with its building blocks. Image available at [22]

EfficientNetv2 [24] family aims to improve upon original models by optimizing building blocks in exchange for better training and computational efficiency, for example by gradually replacing the original MBConv structures with Fused-MBConv.

5.4 Dataset

Naturally, the only eligible datasets were the ones containing people. Multiple were considered, such as WiderPerson [34]. Although being large and quite diverse dataset, it proved to have too many features and fine details for the student model to accurately extract and classify during testing. A smaller, concise dataset was chosen, the Human Detection Dataset [30]. This contains two classes of images, ones containing a person (557), and those without people (364).

Figure 5.2: This figure depicts an entry from the WiderPerson dataset available at [34]

5.5 Distilling strategy and loss calculation

This section will discuss the most important part of this thesis, the knowledge distillation process. The implementation utilizes the response-based, offline distillation method, meaning during a training step an image is first fed through the pre-trained teacher model to compute so called soft labels. After which, the same image is fed through the student model to compute student's output. Finally, teacher's soft labels and student's output are compared in a loss function, from which the final loss is computed and eventually back propagated to the student model. This process can be further separated into, so called *inconsistent* and *consistent* teaching.

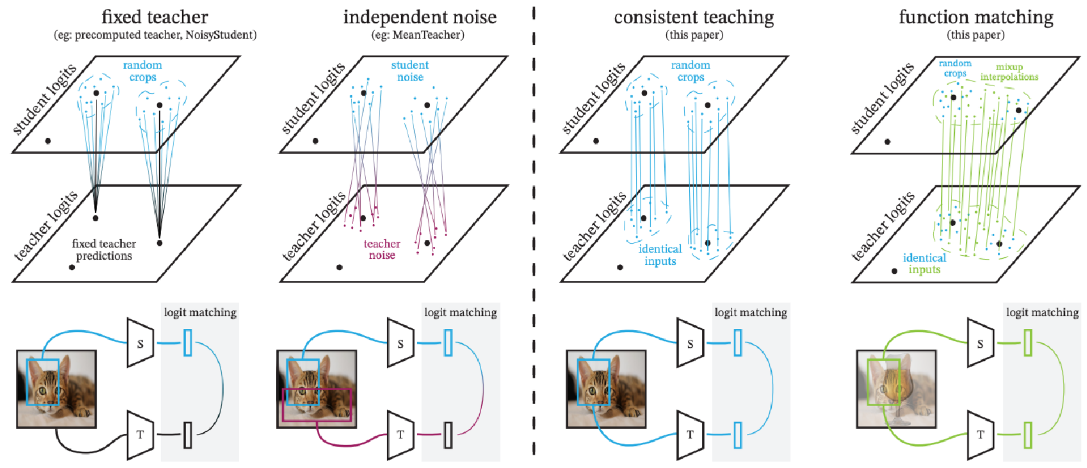


Figure 5.3: This figure depicts various designs of knowledge distillation with inconsistent teaching designs on the left and consistent teaching designs on the right. Image available at [7]

- **Fixed teacher** — this approach first computes logits or soft labels from the teacher network independently from the training process and stores them, for example in a database for further processing.
- **Independent noise** — this approach feeds an image to the teacher and student network with independent stochastic augmentations respectively.
- **Consistent teaching** — this approach feeds an image to the teacher and student network with consistent stochastic augmentation.
- **Function matching** — this approach utilizes so called mixup [33] augmentation, which in essence is convex combination of pairs of images and their labels.

Statistically consistent teaching designs outperform inconsistent in accuracy over long training periods which generally range in thousands of epochs. Both the consistent teaching and function matching designs were tested during training of the student model within this thesis but they ended up being detrimental since any image augmentation would skew the teachers output making it quite inaccurate, thus making the student „confused“ for the lack of a better word. Eventually the consistent teaching approach was selected without any image augmentation.

Distiller implementation

TensorFlow / Keras allow for fully customizing the training process and even making it possible to implement a custom training loop. This thesis' implementation consisted of subclassing the *keras.Model()* and overriding the *train_step()* method. This allowed for fine control over each step of the training process. Following the selected distillation strategy, first the batch of input images is fed to the teacher. The RetinaNet teacher produces a dictionary of tensors consisting of **bounding_box** and **classification** raw outputs, from which only the latter is relevant within this implementation.

The **classification** tensor has the shape of $BATCH\_SIZE \times NUM\_DETECTIONS \times TNUM\_CLASSES$ where $NUM\_DETECTIONS$ is in order of thousands belonging to each of the $BATCH\_SIZE \times images$ consisting of tensors of shape $TNUM\_CLASSES$ which is 20. This output has to be trimmed, because the student's raw output is the shape of $BATCH\_SIZE \times SNUM\_CLASSES$ where there are $BATCH\_SIZE \times images$ tensors of shape $SNUM\_CLASSES$ which for the student model is 2. This is done by taking the teacher's output, slicing out the values containing the raw prediction of a person being present in the image which is on the *14th* index[4]. As per the RetinaNet's implementation on getting the real probability value from raw output, the sigmoid function is applied on these values. At this point, there are 2 problems that need solving, that is to **A)** — reduce the $NUM\_DETECTIONS$ from the order of thousands to just one per image, and **B)** — reduce the $TNUM\_CLASSES$ tensor from 20 to just 2.

One of possible solutions to the first problem was to average all the values out into one tensor. This approach though, would greatly pollute and provide untruthful knowledge. A better alternative that was implemented was to select a single tensor value with the highest probability of the image containing a person (the *14th* index), calling it the **person_probability**.

Similarly, the second problem had various possible solutions, one of which was to average out all the other 19 class probabilities and use that value as the probability of the image not containing a person. This approach was also scrapped as it would greatly soften all the probabilities. Approach that ended up being implemented is to use a value denoting that the image does not contain an person calculated as $1 -$ **person_probability**, essentially softmaxing the probabilities, which is useful since the student's output is also softmaxed.

Naturally, neither of these approaches are 100% accurate and transparent relative to clarity of the knowledge, though beneficially, in a way, they put a greater weight on the probability of the image containing a person.

After masking the teacher's output, the same image is passed through the student network and both outputs are ready to be compared to calculate the loss. This thesis utilizes combined loss computed by two different loss functions, for distillation loss and ground truth loss.

[4]https://github.com/NVIDIA/DIGITS/blob/master/examples/semantic-segmentation/pascal-voc-classes.txt, naturally, the index is off by 2 because of 0 indexing and model ignoring the *background* class

KL Divergence loss

Kullback-Leibler divergence [31, 10][5] measures how one probability distribution diverges from a second, refference probability distribution. The KL divergence is non-negative and equals 0 when the two probabilities are identical. This kind of loss is particularly utilized in knowledge distillation training scenarios. It can be expressed as

$$D_{KL}(P_T \mid\mid P_S) = \frac{P_T}{T} \log \frac{\frac{P_T}{T}}{\frac{P_S}{T}} \tag{5.1}$$

where P_T is the softmax output of the teacher network, P_S is the softmax output of the student network and T is the temperature hyperparameter. This loss will be noted as the **distillation loss** ($Loss_{KL}$).

Binary cross-entropy

Binary cross-entropy[13, 10][6] is a type of cross-entropy used in binary classification tasks with mutually exclusive classes. It computes loss between true, ground truth labels and predicted labels. This kind of loss penalizes the model more heavily for predicting the incorrect label. It can be expressed as

$$B_{CE}(y, P_S) = \frac{1}{N} \sum_{i=1}^{N} -(y_i \cdot \log(P_{Si}) + (1 - y_i) \cdot \log(1 - P_{Si})) \tag{5.2}$$

where y are the ground truth labels, P_S are predictions of the student network and N represents number of samples. This loss will be noted as the **hard loss** ($Loss_{hard}$).

Total Loss

Finally, the total combined loss for the student model can be expressed as follows

$$Loss(y, P_T, P_S) = \alpha Loss_{KL}(P_T, P_S) + \beta Loss_{hard}(y, P_S) \tag{5.3}$$

where α and β are hyperparameters assigning weight to each of the losses. This approach proved to be very beneficial for the long training schedule. During testing, when only ground truth labels were used (without the teacher model) during training, the student model started to overfit the dataset very quickly without gaining any significant accuracy. On the other hand, using only the distillation loss gained from comparing teacher's and student's outputs did not converge at all, even during much longer training periods. The loss combination helped to achieve higher accuracy within reasonable number of epochs, given the usual training schedules in knowledge distillation scenarios.

[5] https://github.com/keras-team/keras/blob/v3.3.3/keras/src/losses/losses.py#L335
[6] https://github.com/keras-team/keras/blob/v3.3.3/keras/src/losses/losses.py#L387

Chapter 6

Experiments and Metrics

This chapter summarizes results of the implementation. First, it summarizes the training process. Finally, the student model (SM) training results will be evaluated and compared obtained by custom metrics and compared to the teacher model (TM).

6.1 Training

This section discusses the training pipeline and parameters used during the process. The dataset was split into a training and testing dataset, where the testing dataset contained 5% of images from each class, 27 and 18 images respectively to each class. Labels are obtained from the first number of the image filename where 0 is denoting that the picture does not contain a person and 1 is denoting that the image contains a person. Thanks to Metacentrum[1] the training was conducted on various hardware configurations varying in available GPU computational power and VRAM capacities. Parameters were selected through the vast testing process and were as follows. The number of **epochs** was set to 2000. Any lower than that, the student model would not provide significant accuracy, any higher than that and the student's accuracy would not change in any meaningful way. The **batch size** was set to 64 providing a good balance between training speed and SM's ability to learn. The **image size** was set to 224×224, again striking good balance between computational cost and extractable information, also conforming with the SM's input layer shape. As for the optimizer, the *SGD* optimizer was selected with **learning rate** of 0.01 and **momentum** of 0.9. The parameter that was the most experimented with was the **temperature**. Eventually the value of 2 was selected. Naturally, value of 1 would not change the probability distribution in any way and higher values proved to be detrimental in SM's eventual accuracy. Parameters α and β in respect to overall loss calculation 5.3 were lightly experimented with but the best result was yielded with both values set to 1. During the training process, various checkpoints were saved, based on the lowest overall loss value, though the final model trained over the entirety of epochs was selected, as it displayed the highest accuracy.

[1]https://metavo.metacentrum.cz/en/

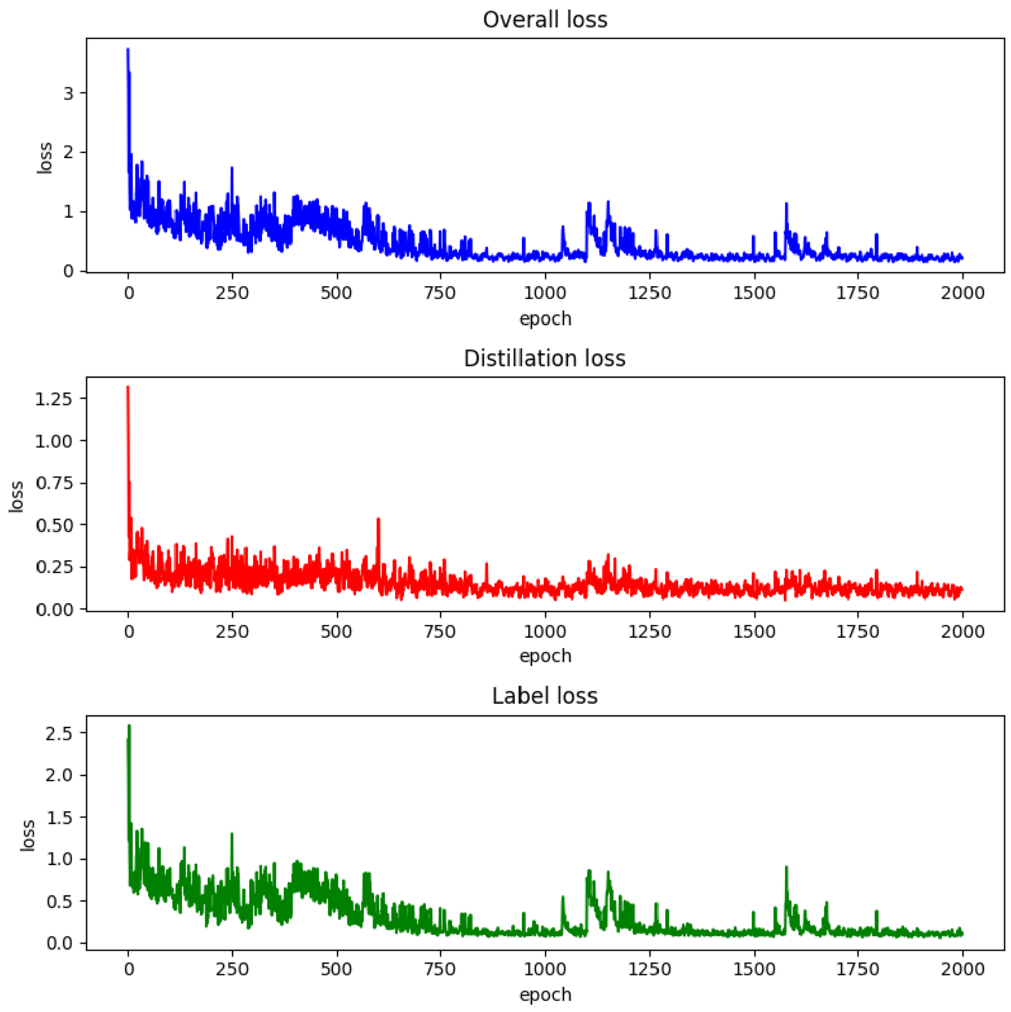


Figure 6.1: This figure depicts the loss reduction over the training process

6.2 Evaluation

This presents custom evaluation metrics. Firstly, model sizes are compared, after which the accuracy metric will be discussed. Finally, the **distillation score** metric will be presented.

Model sizes

When it comes to RetinaNet models, various backbones can be utilized to build the model. Since the SM is from the EfficientNetV2 family, it too can be used as the backbone for the RetinaNet model. Consequently, the SM is relatively more simple when it comes to the architecture. The TM consists of multiple subnetworks, where the SM can be thought of as just one network. The SM contains *8 772 192 (33.46 MB)* parameters in total, compared to the TM's *35 596 952 (135.79 MB)*, which results in approximately 4 times reduction in parameters.

Inference speed

With reduction in parameters and simpler architecture comes the benefit of improved inference speed. The benchmark was conducted on a system with Ryzen 5 2600 CPU clocked at 3.9GHz and a NVIDIA GeForce GTX 980 Ti. Naturally, trying to time model's prediction time with inserting for example Python's *time* module before and after the prediction execution would not provide accurate result [11]. Luckily, Keras' *Model.predict()* provides concise information to the standard output during the execution of the program. It measures the time it took to pass a batch of images through the network. In this case, each step consisted of just 1 image and the benchmark went through the entire dataset of 921 images. This output was captured both for the TM and SM, the times were extracted and averaged out over 5 independent runs. On average the TM took *123.91ms* to process a single image while the SM only took *32.87ms*. This translates into approximately 3.77 times reduction of inference time.

Accuracy

Since the TM and SM outputs are not the same, a custom accuracy metric had to be devised. The proposition is as follows. Images and labels from a particular dataset split will be passed to a model. The model's output will be processed and compared to the ground truth labels. This will be done in a *thresholded* manner, meaning, if a probability of a particular class corresponding to a label will be higher than the threshold, the prediction will be evaluated as a correct one. For example, let the threshold be *0.8* and an image with ground truth label of *1* will be passed to a model. If the model predicts the image to be classified as a class 1 (i.e. containing a person in context of this thesis) with the probability of *0.81*, it will be counted as a correct prediction, whereas if the prediction probability were to be *0.7*, it would count as an incorrect prediction. The number of correct predictions will be compared to the total amount of images within a particular dataset split and the final percentage will be regarded as the final accuracy. Multiple threshold values will be evaluated.

Similarly to previous operations with the TM, it's output had to be masked out to contain only relevant information. This was made easier by RetinaNet's prediction decoder implementation utilizing non maximum suppression. This filtered out much of the model's predictions with low probabilities and overlapping predictions. With RetinaNet being an object detection model, it can predict multiple occurrences of a particular class within one image. For the intensive purposes of computing the accuracy, classes other than *person* were filtered out from model's output and the highest probability was compared against the threshold while predicting an image containing a person. Firstly, both the TM and SM were evaluated on the entire dataset with threshold values *0.70, 0.80, 0.90, 0.95.*

Threshold	**0.70**	**0.80**	**0.90**	**0.95**
Teacher	76.87%	68.51%	52.88%	43.87%
Student	98.15%	91.53%	81.43%	65.91%

Here we can see that TM's accuracy drops off quite significantly, which is to be expected from object detection models with relatively large amount of differentiable classes. On the contrary, we can see that the SM retains quite high accuracy even with higher threshold values and that most of the confidence values lie between 90% and 95% which greatly surpass the TM.

Furthermore, the SM was evaluated on the test dataset with the same threshold values.

Threshold	**0.70**	**0.80**	**0.90**	**0.95**
Student	100%	93.33%	88.89%	68.89%

Yet again, the SM yields good accuracy values, thus proving it usable.

Distillation score

The last metric that will be introduced is the distillation score [3]. This metric can materialize the efficiency of a particular knowledge distillation method. It can be described as follows

$$DS = \alpha \cdot (\frac{size_s}{size_t}) + (1 - \alpha) \cdot (1 - \frac{accuracy_s}{accuracy_t}) \tag{6.1}$$

where *size_s* and *size_t* represents the size in parameters of the SM and TM respectively, similarly, *accuracy_s* and *accuracy_t* denotes the accuracy of both models. The closer the score is to 0, the more effective the distillation method was. Additionally, values lower than 0 are valid. This happens when the SM's accuracy is higher than the TM's, which is not common. For this thesis, the α value of 0.5 and the accuracy values thresholded by 0.9 will be selected. The final distillation score evaluated to *-0.15*. In contrast, the method utilized in the Keras documentation[2] achieves the score of *0.12*

Additional notes

During the development and evaluation process, a couple of interesting events happened, which are worth mentioning. When implementing the masking function for TM's output, I noticed that in some cases, the TM predicted an incorrect label. Particularly, in this image

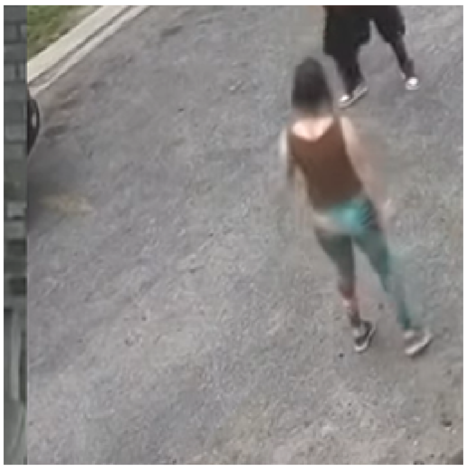


Figure 6.2: This figure depicts incorrectly labeled class

the TM predicted the label as a *dog* with the probability of 71.33%. Despite this, the SM correctly predicted that this image contains a person with the probability of 86.70%. Furthermore, during the testing evaluation, the trained SM found various mislabeled images with an incorrect class, which were corrected and the SM was re-trained. Both these accomplishments can be attributed to the composite loss calculation and mutual complementation between TM's soft labels and hard labels.

[2]https://keras.io/examples/keras_recipes/better_knowledge_distillation/

Chapter 7

Conclusion

The main goal of this thesis was to implement a reduced neural network model capable of classifying the presence of people in an image utilizing the knowledge distillation process, in which it succeeded. It presented various approaches applicable to this discipline, ranging from algorithmical solutions to solutions based on neural networks. The latter approach is highly adaptable, offering the potential to detect or classify multiple classes within an image.

This thesis' implementation utilized a pre-trained RetinaNet model, a single-shot object detector as the teacher model from which knowledge was distilled upon the student model which was a EfficientNetV2 object classifier, yet to be trained. Multiple training and distilling designs were tested with varying results. Finally, a training process utilizing both the teacher's predictions and ground truth labels was implemented yielding good results. The student model ended up being approximately 4 times smaller, in terms of parameters, approximately 3.77 times faster, in terms of inference speeds and had an accuracy with 90% confidence of 81.43% while the teacher only had 52.88% accuracy with the same confidence. Therefore, it can be concluded that the student model exhibited improved performance compared to the teacher model.

Lastly, a simple application was implemented to showcase the capabilities of the trained student model.

The experiments proved, that knowledge distillation is a viable form of model compression capable of training smaller neural networks with the help of a teacher network, that would otherwise be unable to accurately fit the dataset and thus, the usecase.

Bibliography

[1] Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* 2015. Available at: https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Albawi, S.; Mohammed, T. A. and Al Zawi, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET).* 2017, p. 1–6.

[3] Alkhulaifi, A.; Alsahli, F. and Ahmad, I. Knowledge distillation in deep learning and its applications. *PeerJ Computer Science.* PeerJ, april 2021, vol. 7, p. e474. ISSN 2376-5992. Available at: http://dx.doi.org/10.7717/peerj-cs.474.

[4] ArcGIS Developers. *How SSD Works* online. Available at: https://developers.arcgis.com/python/guide/how-ssd-works/.

[5] Bengio, Y.; Goodfellow, I. and Courville, A. *Deep learning.* MIT press Cambridge, MA, USA, 2017.

[6] BenMauss. *Haar-like Features: Seeing in Black and White.* 2021. Available at: https://levelup.gitconnected.com/haar-like-features-seeing-in-black-and-white-1a240caaf1e3.

[7] Beyer, L.; Zhai, X.; Royer, A.; Markeeva, L.; Anil, R. et al. Knowledge distillation: A good teacher is patient and consistent. *CoRR*, 2021, abs/2106.05237. Available at: https://arxiv.org/abs/2106.05237.

[8] Bishop, C. M. Neural networks and their applications. *Review of Scientific Instruments*, june 1994, vol. 65, no. 6, p. 1803–1832. ISSN 0034-6748. Available at: https://doi.org/10.1063/1.1144830.

[9] Bishop, C. M. *Pattern Recognition and Machine Learning.* Springer, 2006.

[10] Chollet, F. et al. *Keras* https://keras.io. 2015.

[11] Deci. *The Correct Way to Measure Inference Time of Deep Neural Networks* online. 2023. Available at: https://deci.ai/blog/measure-inference-time-deep-neural-networks/.

[12] Girshick, R.; Donahue, J.; Darrell, T. and Malik, J. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence.* IEEE, 2015, vol. 38, no. 1, p. 142–158.

[13] Godoy, D. *Understanding Binary Cross-Entropy / Log Loss: A Visual Explanation* online. 2018. Available at: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

[14] Gou, J.; Yu, B.; Maybank, S. J. and Tao, D. Knowledge distillation: A survey. *International Journal of Computer Vision*. Springer, 2021, vol. 129, p. 1789–1819.

[15] Kapsalas, P. and Avrithis, Y. *Encyclopedia of Multimedia*. Boston, MA: Springer US, 2008. 716–722 p. ISBN 978-0-387-78414-4. Available at: https://doi.org/10.1007/978-0-387-78414-4_60.

[16] Lin, T.; Goyal, P.; Girshick, R. B.; He, K. and Dollár, P. Focal Loss for Dense Object Detection. *CoRR*, 2017, abs/1708.02002. Available at: http://arxiv.org/abs/1708.02002.

[17] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S. et al. *SSD: Single Shot MultiBox Detector*. Springer International Publishing, 2016. 21–37 p. ISBN 9783319464480. Available at: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[18] Liu, Y. H. Feature extraction and image recognition with convolutional neural networks. In: IOP Publishing. *Journal of Physics: Conference Series*. 2018, vol. 1087, p. 062032.

[19] O'Shea, K. and Nash, R. *An Introduction to Convolutional Neural Networks*. 2015.

[20] Phung and Rhee. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*, october 2019, vol. 9, p. 4500.

[21] Redmon, J.; Divvala, S.; Girshick, R. and Farhadi, A. *You Only Look Once: Unified, Real-Time Object Detection*. 2016.

[22] Taheri Gorji, H.; Shahabi, M.; Sharma, A.; Tande, L.; Husarik, K. et al. Combining deep learning and fluorescence imaging to automatically identify fecal contamination on meat carcasses. *Scientific Reports*, february 2022, vol. 12, p. 2392.

[23] Tan, M. and Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR*, 2019, abs/1905.11946. Available at: http://arxiv.org/abs/1905.11946.

[24] Tan, M. and Le, Q. V. EfficientNetV2: Smaller Models and Faster Training. *CoRR*, 2021, abs/2104.00298. Available at: https://arxiv.org/abs/2104.00298.

[25] Uzair, M. and Jamil, N. Effects of Hidden Layers on the Efficiency of Neural networks. In: *2020 IEEE 23rd International Multitopic Conference (INMIC)*. 2020, p. 1–6.

[26] V7 Labs, P. B. *Neural Networks Activation Functions* online. 2021. Available at: https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=An%20Activation%20Function%20decides%20whether,prediction%20using%20simpler%20mathematical%20operations.

[27] V7 Labs, R. K. *YOLO (You Only Look Once): An Introduction to Object Detection* online. 2023. Available at: https://www.v7labs.com/blog/yolo-object-detection.

[28] Van Rossum, G. and Drake Jr, F. L. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[29] Viola, P. and Jones, M. J. Robust real-time face detection. *International journal of computer vision*. Springer, 2004, vol. 57, p. 137–154.

[30] Werner, C. *Human Detection Dataset* https://www.kaggle.com/datasets/constantinwerner/human-detection-dataset. 2021.

[31] Wikipedia contributors. *Kullback–Leibler divergence — Wikipedia, The Free Encyclopedia*. 2024. Available at: https://en.wikipedia.org/w/index.php?title=Kullback%E2%80%93Leibler_divergence&oldid=1221741123.

[32] Wood, L.; Tan, Z.; Stenbit, I.; Bischof, J.; Zhu, S. et al. *KerasCV* https://github.com/keras-team/keras-cv. 2022.

[33] Zhang, H.; Cissé, M.; Dauphin, Y. N. and Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. *CoRR*, 2017, abs/1710.09412. Available at: http://arxiv.org/abs/1710.09412.

[34] Zhang, S.; Xie, Y.; Wan, J.; Xia, H.; Li, S. Z. et al. WiderPerson: A Diverse Dataset for Dense Pedestrian Detection in the Wild. *IEEE Transactions on Multimedia (TMM)*, 2019.

[35] Zhuo, L.; Zhu, Z.; Li, J.; Jiang, L.; Zhang, H. et al. Feature extraction using lightweight convolutional network for vehicle classification. *Journal of Electronic Imaging*. SPIE, 2018, vol. 27, no. 5, p. 051222. Available at: https://doi.org/10.1117/1.JEI.27.5.051222.

Appendix A

Contents of the included storage media

- `data/` Folder containing the dataset
- `img/` Folder containing images meant to to demonstrate the application's functionality
- `models/` Folder containing the trained model with training history
- `src/` Folder containing the source files
- `latex/` Folder containing LaTeXsource files
- `thesis.pdf` This thesis
- `requirements.txt` List of library dependencies with their versions
- `detect.py` Application demonstrating the trained model
- `README.md` Manual file