UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Deep Learning for content-based indexing of TV programs

Author:
Aitor LUCAS CASTELLANO
Noel RABELLA GRAS

Supervisor:
Dr. Jordi VITRIÀ MARCA
Paula GÓMEZ DURAN

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

July 1, 2021

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Deep Learning for content-based indexing of TV programs

by Aitor LUCAS CASTELLANO Noel RABELLA GRAS

In recent years, deep neural networks have been successful in a lot of tasks in both industry and academia due to its scalability to mange large volumes of data and model parameters. Unfortunately, creating those large models and use their predictions can be computationally expensive to deploy on devices with limited resources. There is a TV channel called TV3 that wants to improve its recommendation engine without the mentioned impediments. In that thesis, we aim to solve part of that problem by using YOLO and Places to detect objects and scenes respectively, and build a smaller model able to learn from them and extract frame objects and scenes by itself. To do it, we have analyzed in depth *Heterogeneous Classifiers (HC)*, that ensemble models with some different classes in a smaller model using a convex optimization approach. As HCs do not handle an scenario where classes differ completely between models, which is the TV3 case, we have implemented the smaller model following a label prediction approach by using RMSE and we have evaluated the model with ranking metrics as we have faced an unsupervised problem.

Acknowledgements

I would like to thanks my parents, my little brother and my grandparents for supporting me during my academic days until now. Also, thanks to Noel for sharing with me all the funny moments that this thesis has left us and for all the days we have spend in your house. Last but not least, thanks to Estefania for being by my side and cheering for me during all this time that I have been doing the thesis.

I would like to first of all thank my family for being my main point of support, especially Isabel. Also to Tamara, for always being there when I need it. And of course also to Aitor, for knowing how to get a laugh out of any problem.

Finally, on behalf of the two, we would like to thanks Jordi and Paula for all the meetings, help and moments we have shared during the past months. We wish you the best.

Contents

Chapter 1

Introduction

1.1 Project PICAE

Our thesis is born from a spanish project called PICAE ("Proyecto PICAE: Publicación Inteligente de Contenidos Audiovisuales y Editoriales"). From their website, their main objective is to study innovative recommendation models of audiovisual and editorial content with a double objective: to improve the user experience based on their profile and their environment and, therefore, their satisfaction and loyalty, as well as to improve the index of digital consumption of these contents, based on identifying the products that these new forms of consumption demand and how they should be produced, distributed and promoted to respond to the needs of this emerging market.

1.2 The problem of TV programs indexation

The objective of our thesis is to solve and improve the content indexation of TV3 ("Televisió de Catalunya - CCMA"), the primary television channel from Catalonia. On one side, we cannot label all the frames from the TV channels in a supervised manner because it is unfeasible and can be expensive to hire someone to do it. On the other side, another problem is related to the metadata of the TV programs. For instance, the first season of *Merlí* was produced by TV3, making the corporation to be the one managing all the stored data in their servers. But for some circumstances, the second season was managed by another audiovisual production from a different TV channel. Then, because of having different distributions, there is a lack of data for TV3 to build its own recommendation engine for their consumers. That is the reason why project PICAE wants to solve the indexation problem by implementing a new recommendation engine with innovative features such as image and scene recognition or apply Natural Language Processing techniques to the generated subtitles of each program.

1.3 The problem of distillation

Another key aspect to address the TV3 program indexation is related to hardware and computation. In recent years, deep neural networks have been successful in a lot of tasks in both industry and academia due to its scalability to mange large volumes of data and model parameters. Unfortunately, creating those large models and use their predictions can be computationally expensive to deploy on devices with limited resources, e.g, embedded devices and mobile phones, if we consider a huge amount of customers or users.

To this end, (Geoffrey Hinton and Dean, 2015) show that knowledge distillation transfers effectively the knowledge of large models to more compressed models, i.e, models with smaller size and obtain significant results, solving hardware and computation problems and being able to deploy the new models on the mentioned devices.

1.3.1 Introduction to knowledge distillation

A very simple way to improve the performance of machine learning algorithms is to train many different models on the same data and ensemble in a very large model trained with a very strong regularization to avoid overfitting problems. Once the ensemble model has been trained, we can then use a distillation technique to transfer the knowledge from the ensemble model, which is called teacher, to a smaller model called student, that is more suitable for deployment.

The key concept in distillation is to not lose validity while transferring the knowledge. The distribution of values among the outputs for a sample provides information on how a model represents knowledge, because it assigns a greater value to the most probable output class, and smaller values to the other classes. If both models, teacher and student, are trained on the same data independently, the student model might not have enough capacity to learn a precise knowledge representation given the same computational resources and same data as the teacher model.

The key difference in distillation is that in a teacher-student architecture such as in Figure 1.1, a richer knowledge representation is stored in the last layer (*soft labels*) of the teacher model, compared to the usage of *hard labels* from a typical supervised problem. During the thesis report we will be refering soft labels to logits and viceversa.

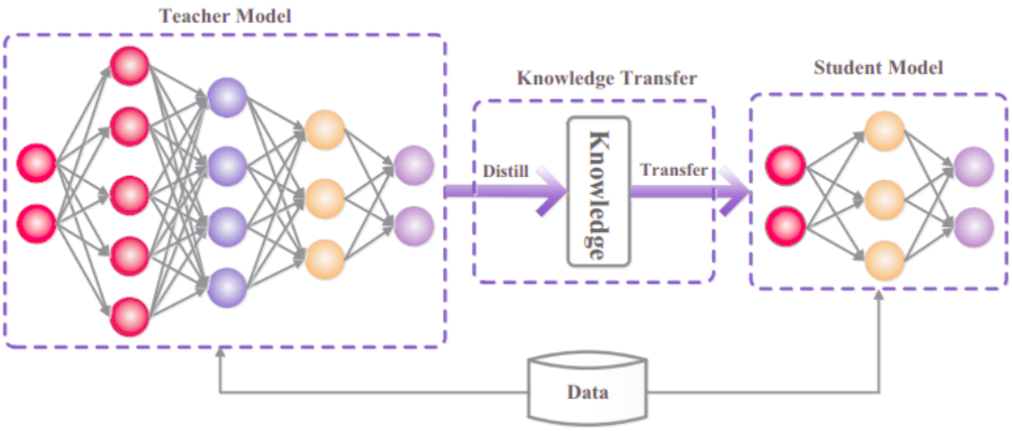


FIGURE 1.1: A simple teacher-student architecture

By this, we do not mean that we do not use the hard labels. On a standard pipeline such as the one in Figure 1.2, source from (Daniele Foffano and Scholman, 2021), to train a student model, first we introduce the image to both the teacher and the student, from the teacher side we extract the soft labels of the image and we combine it with the soft labels of student to input them in a Kullback–Leibler divergence loss, that measures how one probability distribution is different from another. On the other side, we compute a supervised part where we extract the hard labels of the student prediction and we compare it with the true labels of the input image inside a cross-entropy loss. The resultant loss is the sum from both losses and the backpropagation part is computed with the new loss.

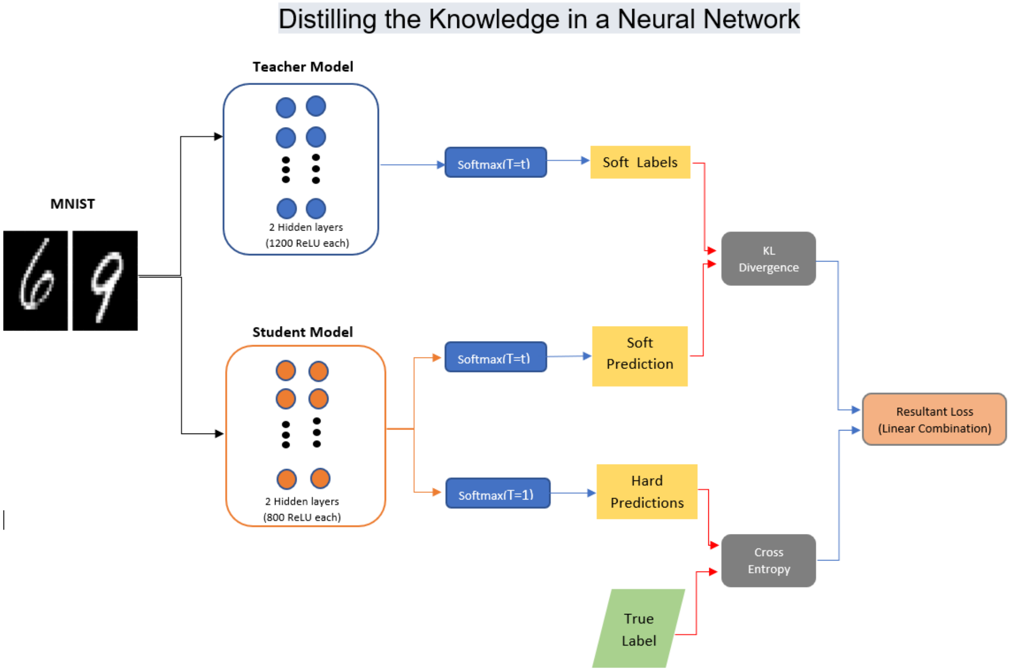


FIGURE 1.2: Pipeline to train a student network

1.3.2 Unifying Heterogeneous Classifiers with Distillation (SOTA)

The success of machine learning in image classification tasks has been largely enabled by the availability of big datasets and the creation of very powerful models. Nowadays, data collection is transitioning towards a more distributed landscape where data is sourced from multiple entities and then combined to train a classifier in a central model. However, in many real case scenarios, due to bandwidth restrictions and privacy concerns, those sources do not share their information and only privately trained models can be shared. Moreover, each source may not be able to create powerful models because of low availability for some model classes or maybe they are not able to train the same classification model due to different computational resources. To face this problem, (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019a) propose a generalisation of knowledge distillation unifying knowledge from a set of classifiers with different architectures and target classes into a single classifier, given only a generic set of unlabelled data.

As we have mentioned in 1.3.1, knowledge distillation main functionality is to compress multiple complex teacher models into a simpler architecture. The main restriction is that distillation assumes that target classes are the same in teachers and students. This paper goes one step further and propose an ensemble of Heterogeneous Classifiers (HCs). HCs are defined as a set of classifiers which may have different architectures and, more importantly, may be trained to classify different sets of target classes. To combine the HCs, authors derive a probabilistic relationship connecting the outputs of HCs with the output of the unified classifier, which is able to classify all target classes of all input HCs. Authors refer to this problem as Unifying Heterogeneous Classifiers (UHC).

Based on this relationship, authors propose two classes of methods, one based on cross-entropy minimisation and the other on matrix factorisation with missing entries, to estimate the probability over all classes of a given sample and use it to train the unified classifier. From this approach, it is possible to train any classifiers with unlabelled data given the soft labels and its major entropy. Figure 1.3 visualizes the main differences between the distillation embedding and the UHC.

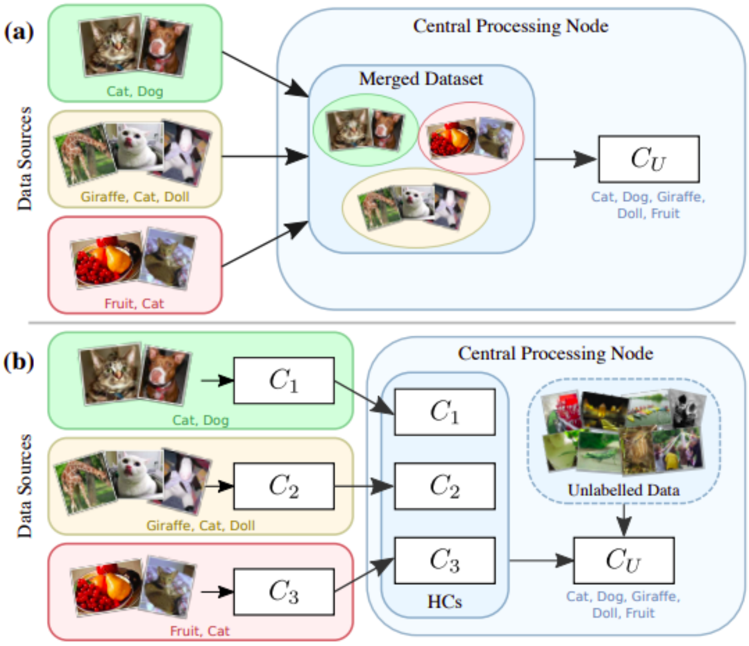


FIGURE 1.3: Unifying Heterogeneous Classifiers (UHC). (a) Common knowledge distillation pipeline. (b) UHC pipeline.

1.3.3 Mathematical notation

After introducing the basics from Distillation in 1.3.1 and UHC in 1.3.2, we define from (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019a) each variable as the following:

- let $\mathcal{U}$ be an unlabelled set of images, which authors call the *transfer set*.
- let $\mathcal{C} = \{C_i\}_{i=1}^{N}$ be a set of N Heterogeneous Classifiers (HCs) (teachers), where each C_i is trained to predict the probability $p_i(Y = l_j)$ of an image belonging to class $l_j \in \mathcal{L}_i$.

Mathematical review of Distillation

Distillation is an algorithm for compressing multiple trained models/teachers C_i into a single unified model/student C_U. Compared to UHC, distillation corresponds to the case where $\mathcal{L}_i = \mathcal{L}_j, \forall(i,j)$. The student model is trained by minimising the cross-entropy between the soft labels of C_i and C_U as:

$$J(q) = -\sum_{i}\sum_{l\in\mathcal{L}_U} p_i(Y = l)\ log\ q(Y = l) \qquad (1.1)$$

In the case of our work, as we are working with neural networks, class probabilities are usually computed with softmax function:

$$p(Y = l) = \frac{exp(z_l/T)}{\sum_{k \in \mathcal{L}_U} exp(z_k/T)} \tag{1.2}$$

The parameter z_l is the logit for class l, T is the temperature parameter.

Extending Knwoledge Distillation with UHC

Given $\mathcal{U}$ and $\mathcal{C}$, the goal of UHC is to learn a *unified classifier* $\mathcal{C}_U$ (student) that estimates the probability $q(Y = l_j)$ of an input image belonging to class $l_j \in \mathcal{L}_U$, where $\mathcal{L}_U = \bigcup_{i=1}^{N} \mathcal{L}_i = \{l_1, l_2, \ldots, l_L\}$. Take into account that $\mathcal{C}_i$ might be trained to classify different sets of classes, *i.e.*, we may have $\mathcal{L}_i \neq \mathcal{L}_j$ or even $|\mathcal{L}_i| \neq |\mathcal{L}_j|$ for $i \neq j$. To tackle the UHC, the three main steps are:

- Pass the image $\mathbf{x} \in \mathcal{U}$ to C_i to obtain p_i $\forall i$.
- Estimate q from $\{p_i\}_i$.
- Use the estimated q to train C_U in a supervised manner.

Going through the second step and defining $\mathcal{L}_{-i}$ as the set of classes in $\mathcal{L}_U$ but outside $\mathcal{L}_i$, we have to relate in some manner the output p_i of each C_i to the probability q over $\mathcal{L}_U$. Since p_i is defined only in the subset $\mathcal{L}_i \subseteq \mathcal{L}_U$, authors consider $p_i(Y = l)$ as the probability q of $Y = l$ given that Y cannot be in $\mathcal{L}_{-i}$, which leads to:

$$\begin{aligned} p_i(Y = l) &= q(Y = l | Y \notin \mathcal{L}_{-i}) && (1.3) \\ &= q(Y = l | Y \in \mathcal{L}_i) && (1.4) \\ &= \frac{q(Y = l, Y \in \mathcal{L}_i)}{q(Y \in \mathcal{L}_i)} && (1.5) \\ &= \frac{q(Y = l)}{\sum_{k \in \mathcal{L}_i} q(Y = k)} && (1.6) \end{aligned}$$

Method 1: Cross-entropy approach

Based on Eq. 1.6, we generalize Eq. 1.1 to:

$$J(q) = -\sum_i \sum_{l \in \mathcal{L}_i} p_i(Y = l) \; log \; \hat{q}_i(Y = l) \tag{1.7}$$

$$\hat{q}_i(Y = l) = \frac{q(Y = l)}{\sum_{k \in \mathcal{L}_i} q(Y = k)} \tag{1.8}$$

Minimization of 1.7 is converted to a convex problem. Authors define $u_l \in \mathbb{R}$ for $l \in \mathcal{L}_U$ and replace $q(Y = l)$ with $exp(u_l)$. Eq 1.7 is transformed to:

$$\hat{J}(\{u_l\}_l) = -\sum_i \sum_{l \in \mathcal{L}_i} p_i(Y = l) \left(u_l - log \left(\sum_{k \in \mathcal{L}_i} exp(u_k) \right) \right) \tag{1.9}$$

We minimise it using gradient descent. Once the optimal $\{u_l\}_l l$ is obtained, we transform it to q with the softmax function.

Method 2: Matrix factorization in probability space

Matrix factorization casts UHC as a problem of filling an incomplete matrix of soft labels. Consider a matrix $\mathbf{P} \in [0,1]^{LxN}$, where we set P_{li} (element in row l and column i) to $p_i(Y = l)$ if $l \in \mathcal{L}_i$ and zero otherwise. Then, we define $\mathbf{M} \in [0,1]^{LxN}$ as a mask matrix where M_{li} is 1 if $l \in \mathcal{L}_i$ and 0 otherwise. We can factorize P into a masked Hadamard product of vectors as:

$$\mathbf{M} \odot \mathbf{P} = \mathbf{M} \odot (\mathbf{u}\mathbf{v}^\top) \tag{1.10}$$

$$\mathbf{u} = \begin{bmatrix} q(Y = l_1) \\ \vdots \\ q(Y = l_m) \end{bmatrix}, \mathbf{v} = \begin{bmatrix} \frac{1}{\sum_{l \in \mathcal{L}_1} q(Y=l)} \\ \vdots \\ \frac{1}{\sum_{l \in \mathcal{L}_N} q(Y=l)} \end{bmatrix} \tag{1.11}$$

In 1.11, $\mathbf{u}$ is the vector containing q and $\mathbf{v}$ the normalization factor for each C_i. To estimate the $\mathbf{u}$, we minimize the following problem:

$$\underset{\mathbf{u},\,\mathbf{v}}{minimize} \quad ||\mathbf{M} \odot (\mathbf{P} - \mathbf{u}\mathbf{v}^\top)||_F^2 \tag{1.12}$$

$$subject\ to \quad \mathbf{u}^\top \mathbf{1}_L = 1 \tag{1.13}$$

$$\mathbf{v} \geq \mathbf{0}_N, \mathbf{u} \geq \mathbf{0}_L, \tag{1.14}$$

where $\mathbf{0}_k$ and $\mathbf{1}_k$ represent vectors of zeros and ones of size k. This formulation is a non-negative matrix factorization problem solved using Alternating Least Squares (ALS), where $\mathbf{u}$ is normalized to sum 1 in each iteration.

Method 2: Matrix factorization in logit space

Let z_l^i be the given logit output from class l of C_i, and u_l be that of C_U to be estimated. Consider a matrix $\mathbf{Z} \in \mathbb{R}^{LxN}$ where $Z_{li} = z_l^i$ if $l \in \mathcal{L}_i$ and 0 otherwise. We estimate the vector of logits $\mathbf{u} \in \mathbb{R}$ as:

$$\underset{\mathbf{u},\,\mathbf{v},\,\mathbf{c}}{minimize} \quad ||\mathbf{M} \odot (\mathbf{Z} - \mathbf{u}\mathbf{v}^\top - \mathbf{1}_L \mathbf{c}^\top)||_F^2 + \lambda(||\mathbf{u}||_2^2 + ||\mathbf{v}||_2^2) \tag{1.15}$$

$$subject\ to \quad \mathbf{v} \geq \mathbf{0}_N, \tag{1.16}$$

where $c \in \mathbb{R}^N$ deals with shift in logits, and $\lambda \in \mathbb{R}$ is a hyperparameter controlling regularization. Optimizing $\mathbf{v} \in \mathbb{R}^N$ is equivalent to optimize the temperature of logits.

1.4 Models

To develop all the experiments we have used the following Deep Learning models:

1.4.1 MNIST Convolutional Neural Network

As we have mentioned during the thesis, we have used the MNIST dataset to check and scale the problem of UHC before implenting it in the real case scenario. To investigate how the knowledge is transferred from the teachers to the student, we have used a simple Convolutional Neural Network (CNN), whose architecture is shown in Figure 1.4 (further details on the channels parameters are exposed in the Experiment contents). The CNN is composed by:

- Three convolutional layers, two flatten layers and the output layer.
- Activation function ReLU and Softmax for the output.
- Dropout of 50% after each convolutional layer.

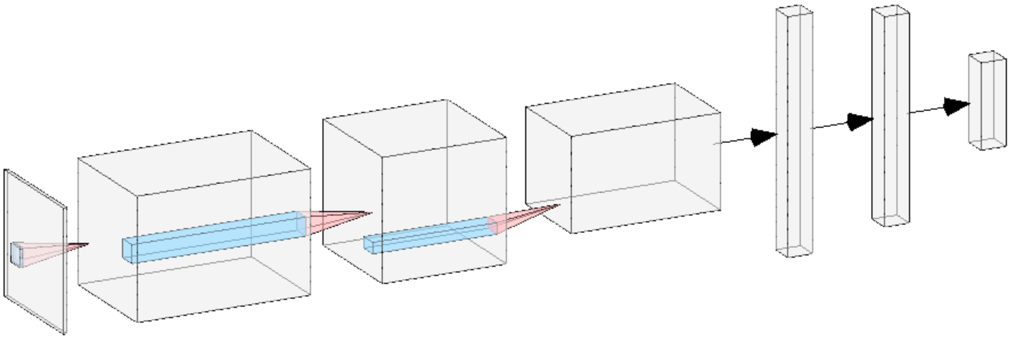


FIGURE 1.4: Architecture of the Convolutional Neural Network

1.4.2 YOLO (You Only Look Once)

YOLO (Joseph Redmon and Farhadi, 2015) is one of the most popular algorithms that provides object detection in real-time. Its popularity comes mainly from its speed to detect objects without having to trade-off with a high error rate. As the name suggest (You only look Once), the algorithm uses a unique convolutional neural network requirying only a single forward propagation in an image to detect objects and predict their classes.

The pretrained model of YOLO used along ours experiments was built with the COCO Dataset. From the original paper (Tsung-Yi Lin, 2014), the MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.

The first version of MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 additional test set of 81K images was released, including all the previous test images and 40K new images.

Based on community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. The new split uses the same images and annotations. The 2017 test set is a subset of 41K images of the 2015 test set. Additionally, the 2017 release contains a new unannotated dataset of 123K images.

On the Figure 1.5 below, we show the detected objects and its predictions of YOLO over an image from our dataset:

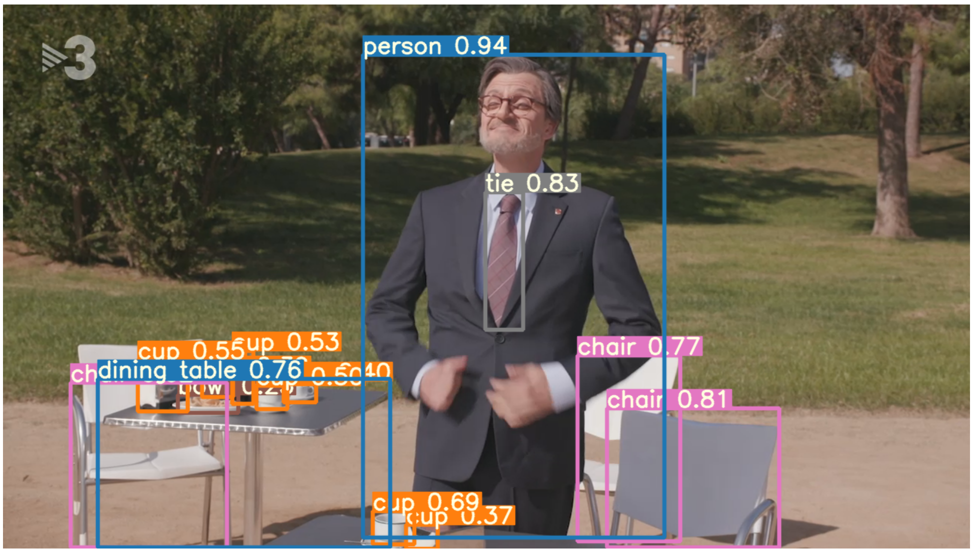


FIGURE 1.5: Object detection and prediction of the scene from program *Polònia* with YOLO

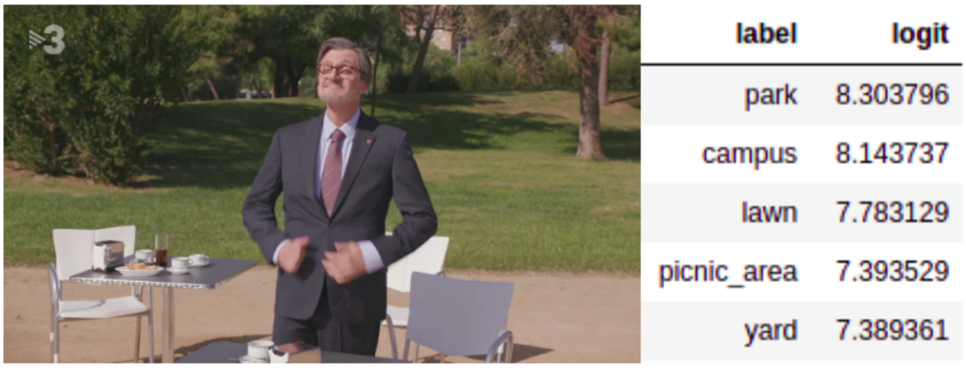
1.4.3 Places365

From the main Github page (“Release of Places365 - CNNs”), Places365 is a subset of images from Places2 database. It has been trained with various Convolutional Neural Networks such as (“ImageNet Classification with Deep Convolutional Neural Networks” 2012, AlexNet), (“Going deeper with convolutions” 2014, GoogLeNet), (“Very Deep Convolutional Networks for Large-Scale Image Recognition” 2014, VGG) and (“Deep Residual Learning for Image Recognition” 2015, ResNet). The main train set of Places365 has around 1.8 million images from 365 scene categories, where there are at most 5000 images per category. Meanwhile, the extended train set of Places365 has extra 6.2 million images along with all the images of the standard Places365, where there are at most 40,000 images per category.

From the official Places website (“Places2 - CSAIL MIT”), the Places dataset is designed following principles of human visual cognition. The goal is to build a core of visual knowledge that can be used to train artificial systems for high-level visual understanding tasks, such as scene context, object recognition, action and event prediction, and theory-of-mind inference. The semantic categories of Places are defined by their function: the labels represent the entry-level of an environment. To illustrate, the dataset has different categories of bedrooms, or streets, etc, as one does not act the same way, and does not make the same predictions of what can happen next, in a home bedroom, an hotel bedroom or a nursery.

In total, Places contains more than 10 million images comprising 400+ unique scene categories. The dataset features 5000 to 30,000 training images per class, consistent with real-world frequencies of occurrence. Using Convolutional Neural Networks (CNN), Places dataset allows learning of deep scene features for various scene recognition tasks, with the goal to establish new state-of-the-art performances on scene-centric benchmarks. Here we provide the Places Database and the trained CNNs for academic research and education purposes.

On the following Figure 1.6, we show an example from our dataset and the Places365 prediction:



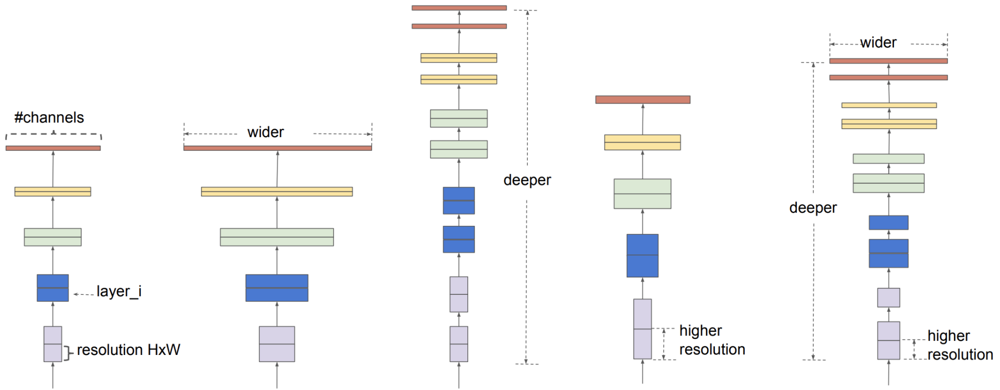
label	logit
park	8.303796
campus	8.143737
lawn	7.783129
picnic_area	7.393529
yard	7.389361

FIGURE 1.6: Scene prediction of the program *Polònia*

1.4.4 EfficientNet

From the official paper ("EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" 2019), EfficientNet is a convolutional neural network architecture and scaling method (see Figure 1.7) that uniformly scales all dimensions using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. For example, if we want to use times more computational resources, then we can simply increase the network depth by, width by, and image size by, where are constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient to uniformly scales network width, depth, and resolution in a principled way.

The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. The base EfficientNet-B0 network is based on the inverted bottleneck residual blocks of MobileNetV2, in addition to squeeze-and-excitation blocks.



(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling (e) compound scaling

Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

FIGURE 1.7: Scaling of the EfficientNet architecture

1.5 Thesis structure

In Chapter 1, we have introduced how Project PICAE and our thesis comes from it. We have talked about the current problem of TV3 and given a theorical view of the algorithms we have implemented to solve the problem. Finally, we present, in a high level perspective, the machine learning models used to develop our experiments during the thesis.

In Chapter 2, we introduce the experimental setup of the thesis: the main datasets, why we have used them and how, a friendly introduction to the software and hardware used to give the reader the smallest detail in our proposed choices.

In Chapter 3, we talk about the motivations and goals behind the project. We ask ourselves which are the key points we want to solve and how we can achieve possible solutions.

In Chapter 4, we explain all the procedure from the first motivation and goal using MNIST. We explain in detail each step to help readers to reproduce the results if they want and we show the results.

In Chapter 5, we explain all the procedure from the motivation and goal of the TV3 problem. We explain in detail each step to help readers to reproduce the results if they want and we show the results.

In Chapter 6, we discuss about all the project, the obtained results and which could be future steps to improve some of the results.

Chapter 2

Experimental setup

2.1 Datasets

Two types of data sets have been used to perform the study carried out in this Master's thesis. On the one hand, to test the different methods from Unifying Heterogeneous Classifiers with Distillation we have used different datasets to scale the complexity of the scenario. First, we have used the MNIST dataset to reproduce and validate if the three exposed methods are able to work in a simple situation. On the other hand, after checking the algorithms and their performance, we have applied the methods to the main dataset of the thesis from TV3, which is a real case scenario.

2.1.1 MNIST

From its main page, the MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm and are centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field. Figure 2.1 from ("Josef Steppan", Wikipedia), show a few samples from the MNIST test dataset.

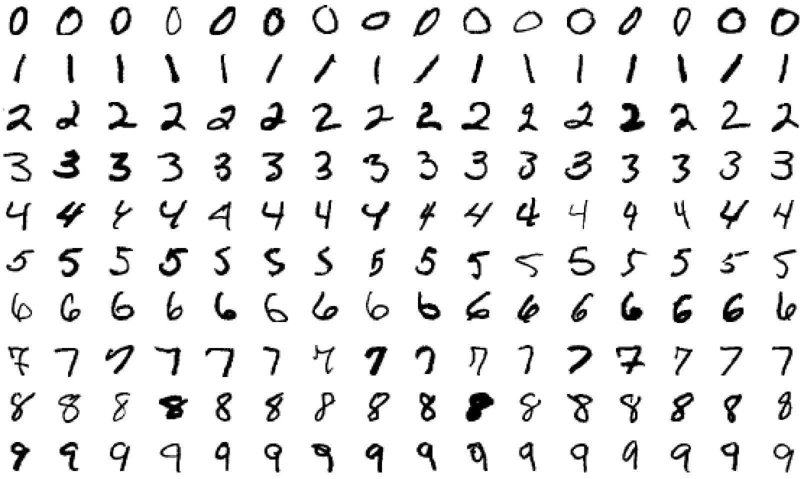


FIGURE 2.1: Examples of the different handwritten digits

2.1.2 TV3

The provided dataset is composed of 80 television programs from the primary catalan channel. From each program, there are approximately 10 different episodes with multiple images. The approach to capture all the frames in the dataset has been to capture them every ten seconds. The total number of the images in the TV3 dataset is 318299, with multiple sizes that have been reshaped automatically by the software. To show which are the programs and the images, we have Table 2.1:

2324	240	El nen clonic
33 recomana	60 minuts	A ritme de pedal
Acció política	Afers exteriors	Amb filosofia
Ànima	Blog Europa	Campiones
Càpsules de ciència	Catalunya Experience	Cinema 3
Com som	Còmics	Crackovia
Cuines	Divendres	Dr. w
Economia en colors	El faro	Cruïlla de camins
El gran dictat	El mur	Els dies clau
Els últims artesans	En clau de vi	Espai Terra
Ets música	Fútbol cat	Generació digital
Gr Barcelona	Gran nord	Identitats
Infidels	Ja t'ho faràs	Jet lag
Karakia	Km 0	Kmm
Entrevista d'estiu	L'hora del lector	La riera
Més 324	Missa conventual	Món 324
Néixer de nou	No serà fàcil	Oh happy day
Oh! Espanya	Òpera en texans	Parlament
Polònia	Polònia: grans moments	Programa sindical UGT
Quarts de nou	Quequicom	Retrats
Riu avall	Sala 33	Sense ficció
Singulars	Sitges	Som dones
Suckers	Teatral	Telenoticies comarques
Thalassa	Tot un món	Tria 33 resum
Tria 33	Valor afegit	Ventdelplà Vespre 324
Veterinaris	Via llibre	Xarxa natura
Zona unef	.cat	-

TABLE 2.1: Different programs in the dataset

Just for illustration purposes, Figure 2.2 shows a sequence of frames showing the sampling of a scene from a program:

FIGURE 2.2: Sequence from the program *El gran dictat*

2.2 Software

2.2.1 Python

Python language is incredibly easy to use and learn for new programmers. It is one of the most accessible programming languages available because it has simplified and not complicated syntax, which gives more emphasis on natural language. Due to its ease of learning, usage and big supportive community of python, python has excellent libraries with different specific focus, such as scikit-learn, numpy or opencv. As Python has a lot of different machine learning libraries, it is the perfect programming language to use for project such as our thesis.

2.2.2 Pytorch

From the official Wikipedia page ("Pytorch-Wikipedia", Wikipedia), PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. PyTorch provides two high-level features that are tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU) and deep neural networks built on a type-based automatic differentiation system.

In PyTorch, models are directed acyclic graphs (DAG). We can define, change and execute nodes plus use modules in an Object Oriented Programming (OOP) manner making the framework more integrated with Python language, which feels more native and pythonic. Morever, since computation graph in Pytorch is defined at runtime, we can use debugging tools easily. Another key aspect on why we have chosen Pytorch is data paralellism, because we are able to leverage multiple GPUS in most of the computations that we have done during the thesis.

2.3 Hardware

To develop and run all the experiments we have used the following configuration:

- Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz
  - RAM memory (in GB): 94
  - Number of cores: 8
  - Number of threads: 16
- PNY Nvidia Quadro RTX 4000 8GB GDDR6

Chapter 3

Motivation and goals

3.1 Transferring knowledge depending on class overlapping

Is it possible to train teachers with fewer and fewer classes in common and transfer knowledge to a student correctly? We have used (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019a) as the base of our work and we have analyzed in depth the "advantage" of being able to train classifiers with different class labels to see if that applies to solve our problem.

That motivation comes from the experiments part in (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019a), where (Jia Deng, 2009, Imagenet), (Fisher Yu, 2015, LSUN) and ("Release of Places365 - CNNs") are used to train teachers but not with all their available classes. Authors do not tell anything about that decision, maybe results do not vary in terms of accuracy and the computational cost decreases significantly in some manner.

The key fact is that if we consider UHC as a step further from distillation and we know its premise tell us that soft labels contain richer information than hard labels, their results section should perform better even if the discarded classes from each dataset are not significant enough because their soft labels carry important information that can be helpful to the most significant ones.

From that, our first goal has been to demonstrate if the proposed methods in 1.3.3 work properly using all label classes from the MNIST dataset, choosen because of its simplicity, and to check the effect of adding less overlapping until achieving the case where teachers have no class intersection, which is not considered in UHC experiments.

3.2 Is it possible to apply UHC to the TV3 problem?

From the above goal, we have seen in Chapter 4 that without any class overlapping, even in a simple classification problem such as MNIST, the accuracy of the students decay considerably, even though soft labels are more robust than hard labels. How can we deal then with indexation problem of TV3? To tackle the problem we have though to use YOLO for detecting objects in frames and Places365 for scene recognition. The premise is that in an ideal scenario, TV programs usually have specific locations and objects, so we should be able to compute a mean from the frame predictions and be able to cluster each program.

Considering to model two teachers from YOLO and Places365, we are facing the problem of implementing UHC without class overlapping. The main difference in that case compared to MNIST is that both models have different functions. In MNIST, even if we train a teacher that learns to identify digits from 0 to 4 and another teacher from 5 to 9, we are playing with dependant probability distributions at the end, even if a teacher that has never seen a 4, it will predict something similar to one of their classes, for instance a 9. In the case of YOLO and Places365, we are playing with class labels that differ in their main functions. Moreover, YOLO returns independent probabilities for each recognized object in a scene, contrary to Place365, where classes compete (by competition, we mean the sum of all predictions is 1).

Then, we have asked ourselves, is it possible to build an UHC from here? The only problem resides in how to manage logits and merge them in order to transfer correctly the knowledge to the student.

3.3 Changing the loss approach to train the TV3 student

After not being able to merge UHC with TV3, we have thought to use another loss criteria, which is explained in Chapter 5 and train the student. The goal then is train a student classifier with the TV3 dataset with the RMSE loss.

3.4 Use a more generalized dataset to train a better student

If we train a student on the TV3 dataset, we are training a model that cannot generalize, and we want something more robust at the end, that the reason on trying to use the base datasets from YOLO and Places. Can we use samples from COCO and Places datasets to train a student model that is able to generalize and have a good performance on the TV3 images?

Chapter 4

Experiments on transfering knowledge with MNIST

4.1 Development

To demonstrate and reproduce if the cross-entropy and matrix factorization methods from 1.3.3 work correctly and help a student model to learn, we have develop several experiments with the CNN architecture mentioned in 1.4.1 by creating different teacher configurations based on their classes. Models are shown in Table 4.1:

Teacher 1	Teacher 2	Common labels
0 to 4	5 to 9	0
0 to 5	5 to 9	1
0 to 5	4 to 9	2
0 to 6	4 to 9	3
0 to 6	3 to 9	4
0 to 7	3 to 9	5
0 to 7	2 to 9	6

TABLE 4.1: Convolutional Neural Network configurations

The training setup for each teacher is the following one:

- Filter train images by targets and create a Pytorch Dataloader with $batchSize = 32$.
- Create the teacher model with 128 channels.
- 50 epochs.
- Learning rate 0.001 with learning rate decay of 0.5.
- Cross-entropy loss.
- Stochastic Gradient Descent optimizer with $momentum = 0.9$.
- Filter test images as the first step and extract accuracies from teachers.
- Save teacher models in a .pt file.

To check if the models have trained correctly, we have plotted the accuracies on the train and test set in Table 4.2:

Teacher	Train acc	Test acc
0 to 4	99%	99%
0 to 5	98%	98%
0 to 6	97%	97%
0 to 7	97%	97%
2 to 9	97%	97%
3 to 9	93%	93%
4 to 9	96%	96%
5 to 9	98%	97%

TABLE 4.2: Convolutional Neural Network configurations

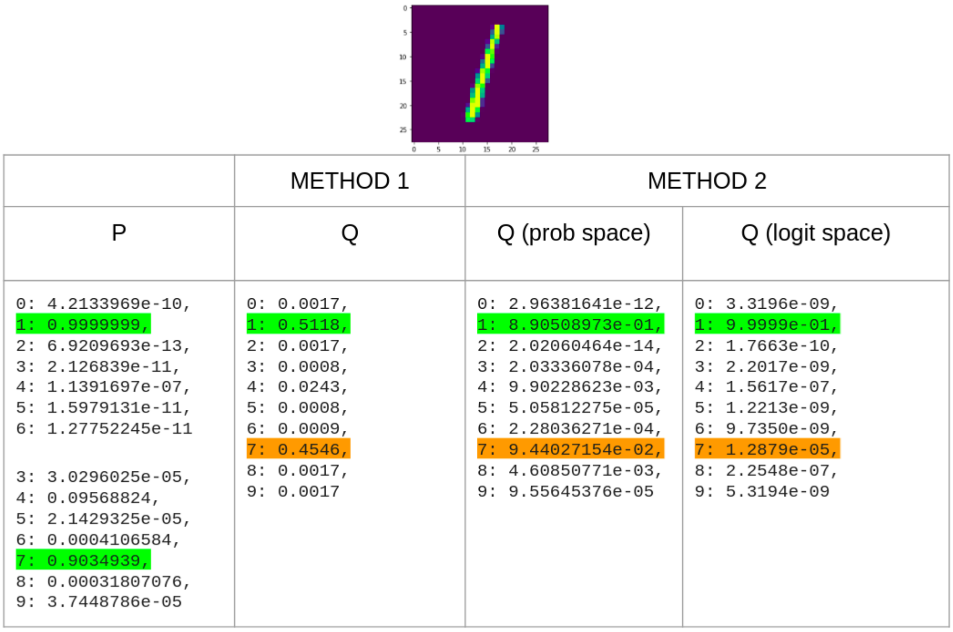
After training each pair of teachers, we compute the estimated q following the next steps:

- Pass train images through teachers to extract logits *(soft labels)* and probabilities.
- Create the mask M, the probability matrix P and the logit matrix Z.
- Use probabilities for method 1 (Cross-entropy approach) and extract the estimated q.
- Use M and P for method 2: Matrix Factorization in Probability Space and M and Z for method 2: Matrix Factorization in Logit Space to extract the estimated q.

4.2 Experiment: extracting method estimations

Figures 4.1, 4.2, 4.3 show different tables that we have developed during our experiments on implementing each method. In that scenario, we have 4 labels in common. The first teacher has been trained with digits 0 to 6, meanwhile the second teacher has been trained with digits from 3 to 9. For analysis purposes, we have plotted all estimated values as probabilities.

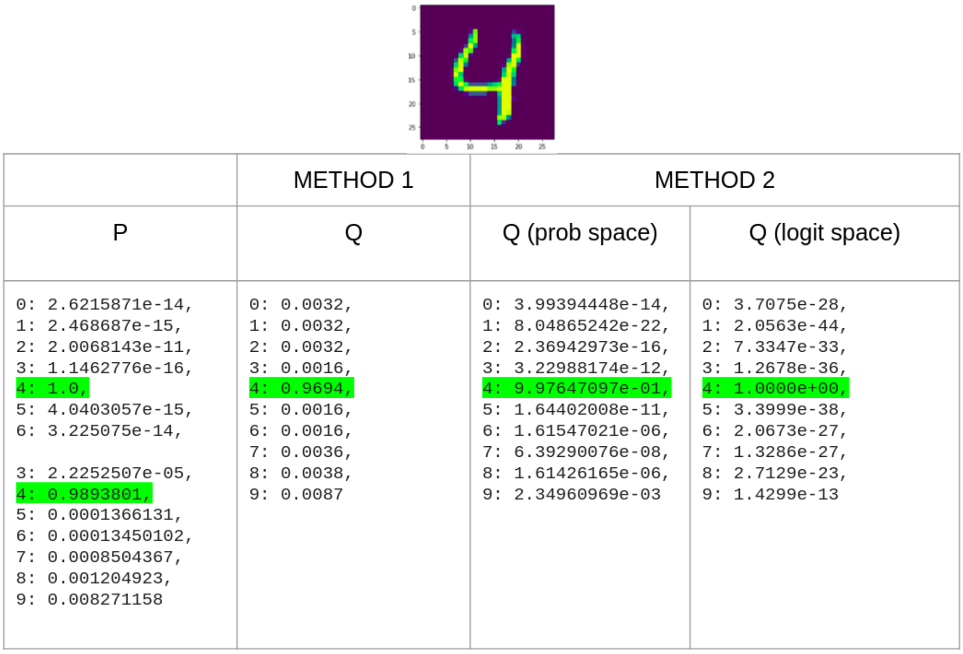
Figure 4.1 show the estimated q values from the digit 1. In green, the max value is remarked, in orange, the second max value. In that case, the first teacher is the one that knows about digit 1, meanwhile the other teacher recognize the digit as the number 7, because it has never seen a number 1 and obviously, the most similar is 7, which we appreciate in the probability distribution. The cross-entropy method converges into a distribution that is confused between 1 and 7. On the other hand, matrix factorization methods converge almost perfectly to the correct number, with accuracies of 89% and 99% respectively.


0
5
10
15
20
25
0 5 10 15 20 25

	METHOD 1	METHOD 2	
P	Q	Q (prob space)	Q (logit space)
0: 4.2133969e-10, 1: 0.9999999, 2: 6.9209693e-13, 3: 2.126839e-11, 4: 1.1391697e-07, 5: 1.5979131e-11, 6: 1.27752245e-11 3: 3.0296025e-05, 4: 0.09568824, 5: 2.1429325e-05, 6: 0.0004106584, 7: 0.9034939, 8: 0.00031807076, 9: 3.7448786e-05	0: 0.0017, 1: 0.5118, 2: 0.0017, 3: 0.0008, 4: 0.0243, 5: 0.0008, 6: 0.0009, 7: 0.4546, 8: 0.0017, 9: 0.0017	0: 2.96381641e-12, 1: 8.90508973e-01, 2: 2.02060464e-14, 3: 2.03336078e-04, 4: 9.90228623e-03, 5: 5.05812275e-05, 6: 2.28036271e-04, 7: 9.44027154e-02, 8: 4.60850771e-03, 9: 9.55645376e-05	0: 3.3196e-09, 1: 9.9999e-01, 2: 1.7663e-10, 3: 2.2017e-09, 4: 1.5617e-07, 5: 1.2213e-09, 6: 9.7350e-09, 7: 1.2879e-05, 8: 2.2548e-07, 9: 5.3194e-09

FIGURE 4.1: Estimated q values from digit 1

Figure 4.2 show the estimated q values from the digit 4. In green, the max value is remarked. In that case, both teachers know about the digit 4. As we can see it seems that there is no problem in a simple scenario such in MNIST when digits overlap. The cross-entropy method converges into a distribution where 4 has a 96% of accuracy. On the other hand, matrix factorization methods converge almost perfectly to the correct number, with accuracies of 99%.

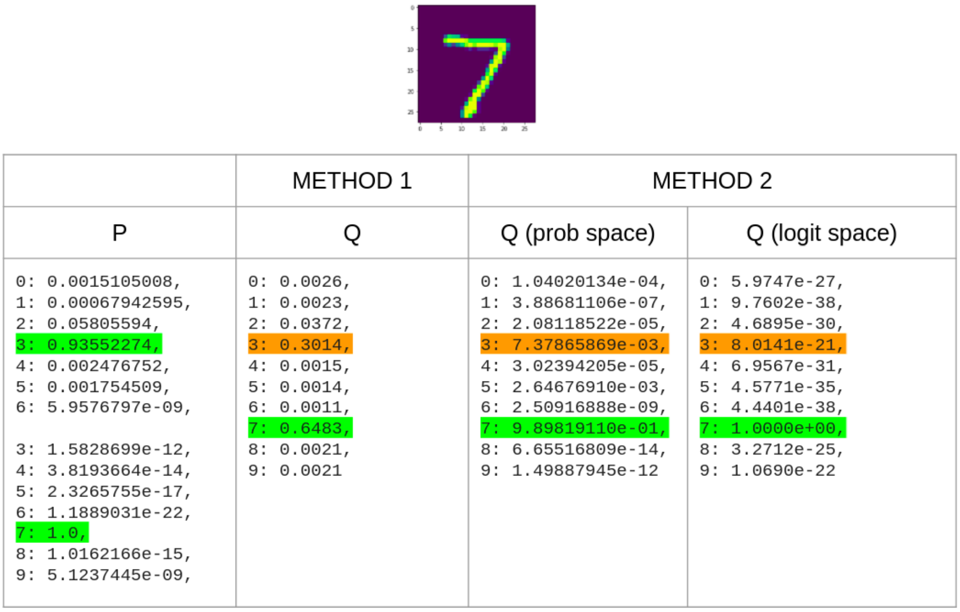

0
5
10
15
20
25
0 5 10 15 20 25

	METHOD 1	METHOD 2	
P	Q	Q (prob space)	Q (logit space)
0: 2.6215871e-14, 1: 2.468687e-15, 2: 2.0068143e-11, 3: 1.1462776e-16, 4: 1.0, 5: 4.0403057e-15, 6: 3.225075e-14, 3: 2.2252507e-05, 4: 0.9893801, 5: 0.0001366131, 6: 0.00013450102, 7: 0.0008504367, 8: 0.001204923, 9: 0.008271158	0: 0.0032, 1: 0.0032, 2: 0.0032, 3: 0.0016, 4: 0.9694, 5: 0.0016, 6: 0.0016, 7: 0.0036, 8: 0.0038, 9: 0.0087	0: 3.99394448e-14, 1: 8.04865242e-22, 2: 2.36942973e-16, 3: 3.22988174e-12, 4: 9.97647097e-01, 5: 1.64402008e-11, 6: 1.61547021e-06, 7: 6.39290076e-08, 8: 1.61426165e-06, 9: 2.34960969e-03	0: 3.7075e-28, 1: 2.0563e-44, 2: 7.3347e-33, 3: 1.2678e-36, 4: 1.0000e+00, 5: 3.3999e-38, 6: 2.0673e-27, 7: 1.3286e-27, 8: 2.7129e-23, 9: 1.4299e-13

FIGURE 4.2: Estimated q values from digit 4

Figure 4.3 show the estimated q values from the digit 7. In green, the max value is remarked, in orange, the second max value. In that case, opposite to the first scenario, the first teacher does not know about digit 7, meanwhile the other teacher has learned in its training about the digit 7.

The interesting part here is how the first teacher recognizes the digit. The network has learned a pattern in which relates 7 to 3, surprisingly, it is not matching 7 with 1 and the probability distribution goes completely towards to 3. The cross-entropy method converges into a less confused distribution, but it still has a strong collision between 3 and 7 probabilities. On the other hand, matrix factorization methods converge almost perfectly again to the correct number, with accuracies of 98% and 99% respectively.



	METHOD 1	METHOD 2	
P	Q	Q (prob space)	Q (logit space)
0: 0.0015105008, 1: 0.00067942595, 2: 0.05805594, 3: 0.93552274, 4: 0.002476752, 5: 0.001754509, 6: 5.9576797e-09, 3: 1.5828699e-12, 4: 3.8193664e-14, 5: 2.3265755e-17, 6: 1.1889031e-22, 7: 1.0, 8: 1.0162166e-15, 9: 5.1237445e-09,	0: 0.0026, 1: 0.0023, 2: 0.0372, 3: 0.3014, 4: 0.0015, 5: 0.0014, 6: 0.0011, 7: 0.6483, 8: 0.0021, 9: 0.0021	0: 1.04020134e-04, 1: 3.88681106e-07, 2: 2.08118522e-05, 3: 7.37865869e-03, 4: 3.02394205e-05, 5: 2.64676910e-03, 6: 2.50916888e-09, 7: 9.89819110e-01, 8: 6.65516809e-14, 9: 1.49887945e-12	0: 5.9747e-27, 1: 9.7602e-38, 2: 4.6895e-30, 3: 8.0141e-21, 4: 6.9567e-31, 5: 4.5771e-35, 6: 4.4401e-38, 7: 1.0000e+00, 8: 3.2712e-25, 9: 1.0690e-22

FIGURE 4.3: Estimated q values from digit 7

4.3 Experiment: evaluating the whole scenario

To check the general effect of the estimated q values, we have extracted three confusion matrices that go from 0 to 2 labels in common. Figure 4.4 corresponds to the confusion matrices of teachers with any labels in common. As we can appreciate, the predictions are not the most accurate ones, for instance, there is a problem for predicting 9, because most of the times is confused with the number 4.

In Figure 4.5, for the matrix factorization in logit space, the problem of 9 and 4 starts to mitigate as both teachers share the number 4. Anyway, the scenario with one label in common is still not decent enough. For the cross-entropy method, the problems still persist, meanwhile matrix factorization in probability space has improved its predictions but not significantly.

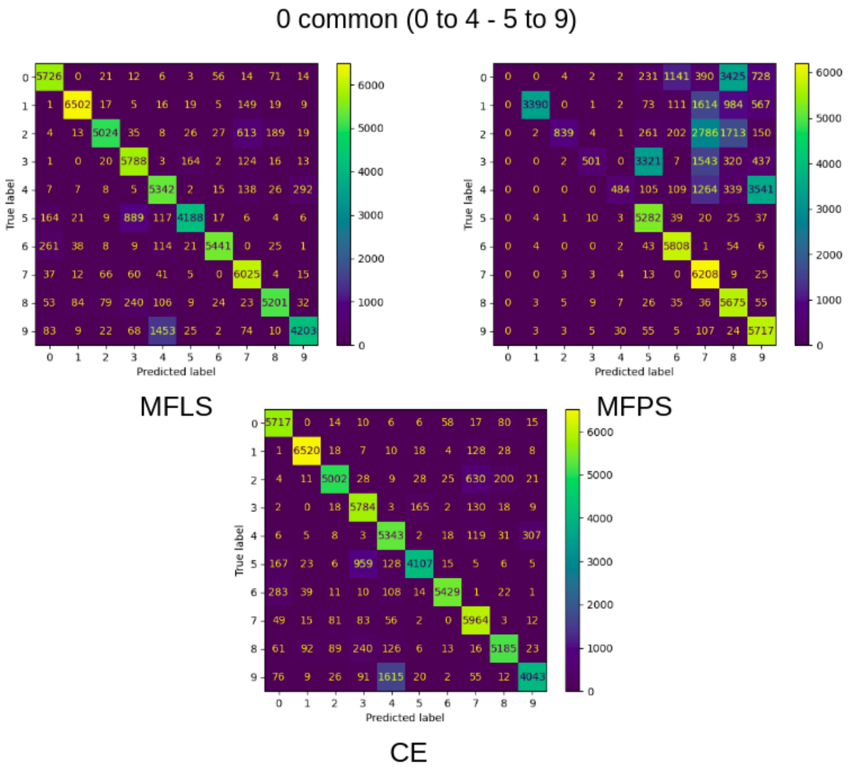


FIGURE 4.4: Confusion matrix with 0 digits in common

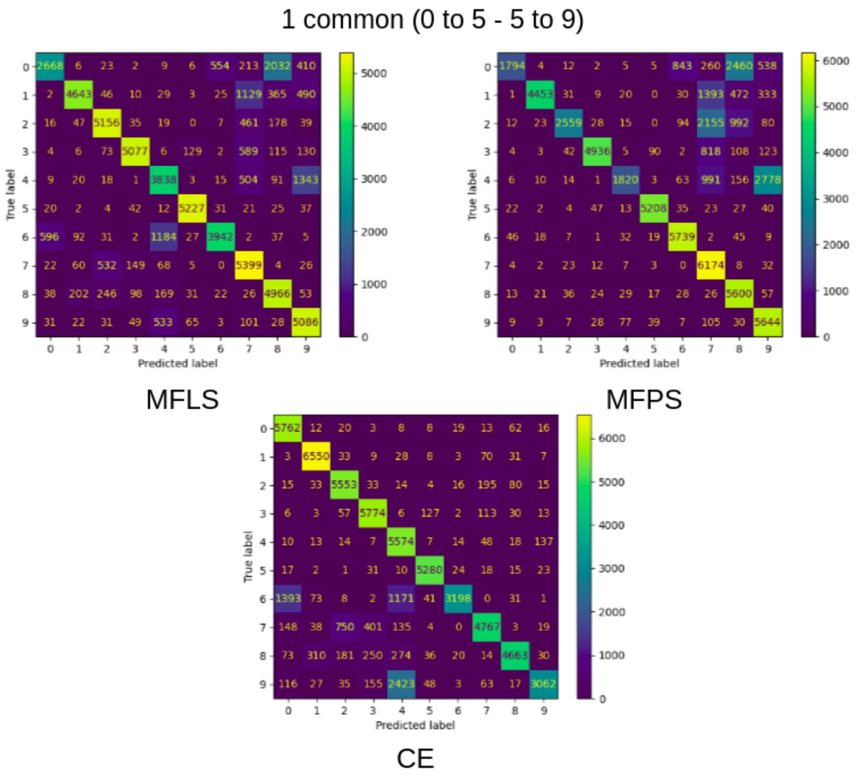
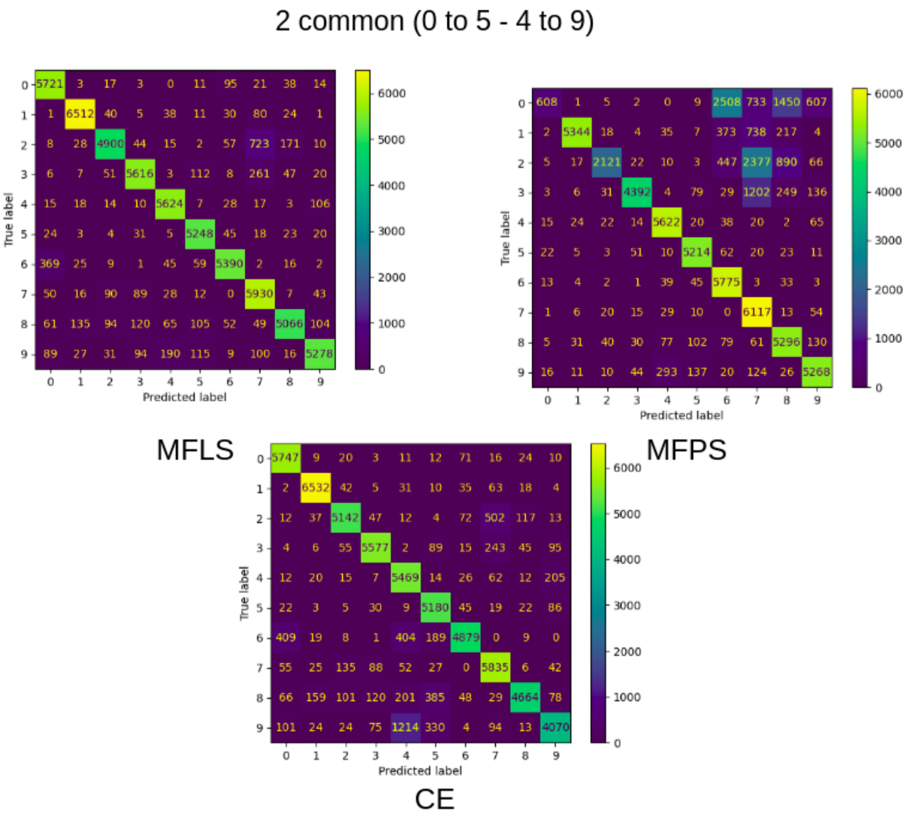


FIGURE 4.5: Confusion matrix with 1 digit in common

In Figure 4.6, the confusion between labels mitigates significantly overall. For the matrix factorization in logit space, the errors have decreased a lot, and for the cross-entropy part there are also less errors.



MFLS

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	5721	3	17	3	0	11	95	21	38	14
1	1	6512	40	5	38	11	30	80	24	1
2	8	28	4900	44	15	2	57	723	171	10
3	6	7	51	5616	3	112	8	261	47	20
4	15	18	14	10	5624	7	28	17	3	106
5	24	3	4	31	5	5248	45	18	23	20
6	369	25	9	1	45	59	5390	2	16	2
7	50	16	90	89	28	12	0	5930	7	43
8	61	135	94	120	65	105	52	49	5066	104
9	89	27	31	94	190	115	9	100	16	5278

MFPS

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	608	1	5	2	0	9	2508	733	1450	607
1	2	5344	18	4	35	7	373	738	217	4
2	5	17	2121	22	10	3	447	2377	890	66
3	3	6	31	4392	4	79	29	1202	249	136
4	15	24	22	14	5622	20	38	20	2	65
5	22	5	3	51	10	5214	62	20	23	11
6	13	4	2	1	39	45	5775	3	33	3
7	1	6	20	15	29	10	0	6117	13	54
8	5	31	40	30	77	102	79	61	5296	130
9	16	11	10	44	293	137	20	124	26	5268

CE

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	5747	9	20	3	11	12	71	16	24	10
1	2	6532	42	5	31	10	35	63	18	4
2	12	37	5142	47	12	4	72	502	117	13
3	4	6	55	5577	2	89	15	243	45	95
4	12	20	15	7	5469	14	26	62	12	205
5	22	3	5	30	9	5180	45	19	22	86
6	409	19	8	1	404	189	4879	0	9	0
7	55	25	135	88	52	27	0	5835	6	42
8	66	159	101	120	201	385	48	29	4664	78
9	101	24	24	75	1214	330	4	94	13	4070

FIGURE 4.6: Confusion matrix with 2 digits in common

4.4 Experiment: evaluating students

After computing the estimated values for each image in the three methods, we have done the following steps:

- Create a Pytorch Dataloader with $batchSize = 32$.
- Create the student model with lower size, for instance, 64 channels.
- 50 epochs.
- Learning rate 0.001 with learning rate decay of 0.5.
- KL-divergence loss.
- Stochastic Gradient Descent optimizer with $momentum = 0.9$.
- In each batch, compute the loss between the student output and their corresponding q value and do the backpropagation step.

After doing several experiments and taking as reference the accuracies from (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019a), which are bad in general, we can conclude that for our first goal, transferring the knowledge to a student is not good enough to considerate the model a good one for production environments. On the other hand, following the distillation premise, we have seen how Matrix Factorization in Logit Space is the best method and the one that carries more significant information thanks to *soft labels*, coinciding with (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019a).

Linking the TV3 problem, from the poor performance of the transferring knowledge problem, it is not feasible to continue with the UHC approach. Moreover, in case it could work, we should think about an approach to deal with the independant logits from YOLO.

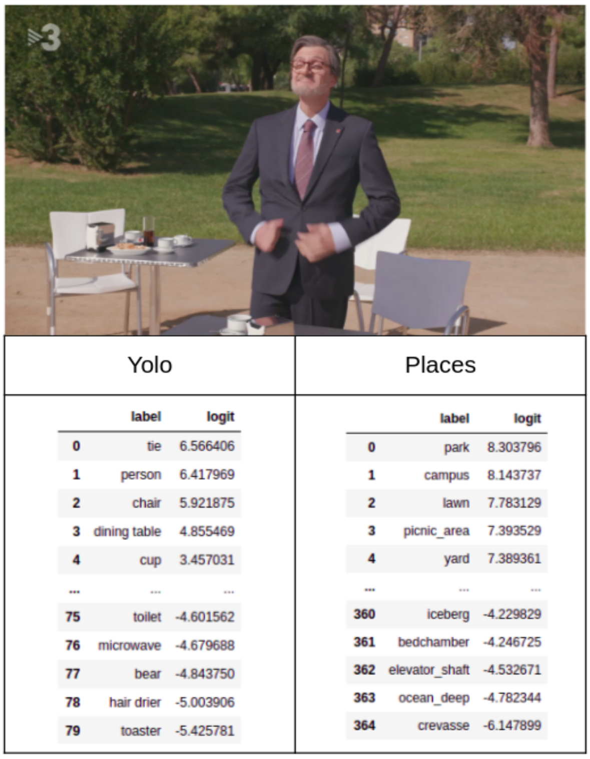
Chapter 5

Experiments on the TV3 problem

After checking the transfer of knowledge from MNIST until having unique different classes in each teacher and the performace, we have to tackle the indexation problem using YOLO for detecting objects in frames and Places365 for scene recognition from a different perspective. The main motivation of this chapter is to go one step further from UHC and be able to find a viable approach to merge the independent logits from YOLO with Places.

5.1 First approach: YOLO, Places on TV3 dataset

Remembering the main objective of our thesis, due to hardware and legal requirements, it is hard to train different classifiers and use them as an embedding later. In our specific case, if these requirements did not exist we could just train one model for object detection and another one for recognition scene. The next figures show some examples of soft and hard labels of the TV3 dataset with great results:



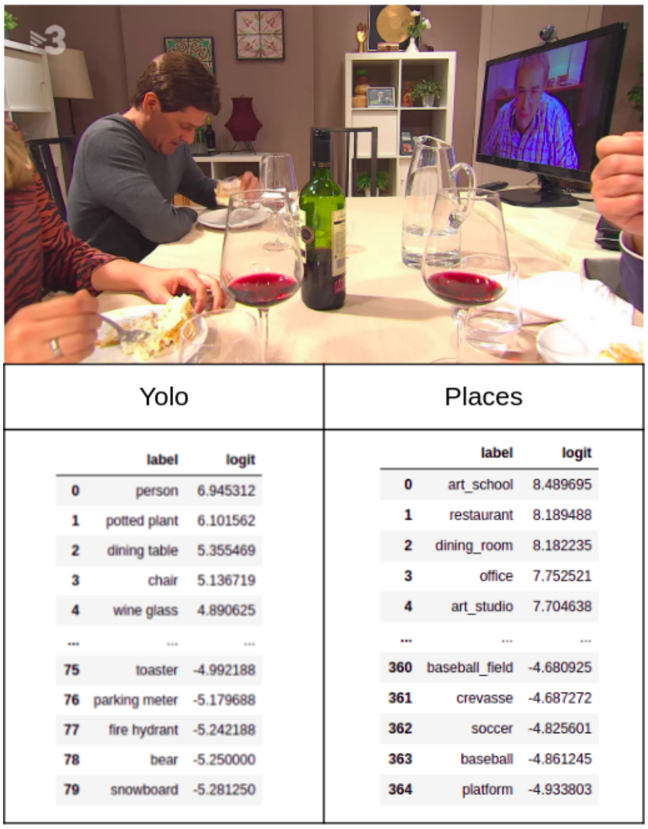
Yolo

	label	logit
0	tie	6.566406
1	person	6.417969
2	chair	5.921875
3	dining table	4.855469
4	cup	3.457031
...	...	...
75	toilet	-4.601562
76	microwave	-4.679688
77	bear	-4.843750
78	hair drier	-5.003906
79	toaster	-5.425781

Places

	label	logit
0	park	8.303796
1	campus	8.143737
2	lawn	7.783129
3	picnic_area	7.393529
4	yard	7.389361
...	...	...
360	iceberg	-4.229829
361	bedchamber	-4.246725
362	elevator_shaft	-4.532671
363	ocean_deep	-4.782344
364	crevasse	-6.147899

FIGURE 5.1: Logit outputs from an image from the TV3 dataset



Yolo

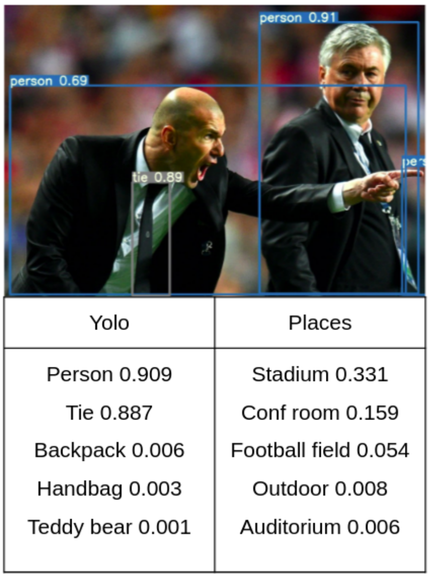
	label	logit
0	person	6.945312
1	potted plant	6.101562
2	dining table	5.355469
3	chair	5.136719
4	wine glass	4.890625
...	...	...
75	toaster	-4.992188
76	parking meter	-5.179688
77	fire hydrant	-5.242188
78	bear	-5.250000
79	snowboard	-5.281250

Places

	label	logit
0	art_school	8.489695
1	restaurant	8.189488
2	dining_room	8.182235
3	office	7.752521
4	art_studio	7.704638
...	...	...
360	baseball_field	-4.680925
361	crevasse	-4.687272
362	soccer	-4.825601
363	baseball	-4.861245
364	platform	-4.933803

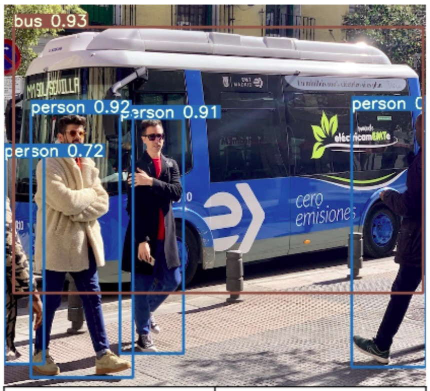
FIGURE 5.2: Logit outputs from another image from the TV3 dataset

From Figures 5.1 and 5.2 we have extracted the best and worst logits to analyze how the logit distributions vary. Focusing on results, we see how easy is for the networks to predict values from images that have never seen.



Yolo	Places
Person 0.909	Stadium 0.331
Tie 0.887	Conf room 0.159
Backpack 0.006	Football field 0.054
Handbag 0.003	Outdoor 0.008
Teddy bear 0.001	Auditorium 0.006

FIGURE 5.3: Probability outputs from an image from the TV3 dataset



Yolo	Places
Bus 0.929	Bus station 0.661
Person 0.916	Street 0.234
Tie 0.246	Crosswalk 0.054
Handbag 0.040	Plaza 0.008
Book 0.012	Phone booth 0.006

FIGURE 5.4: Probability outputs from another image from the TV3 dataset

5.1.1 Extracting YOLO and Places logits

To decide how to extract the final logits from YOLO, we have gone through different steps until achieving a vector of logits for each image. First, we have faced the issue of how to manage the different detected boxes that can appear in a single image, since every box has its own logits. Figure 5.5 show an example of the whole output after using the YOLO in an image that detects 3 objects:

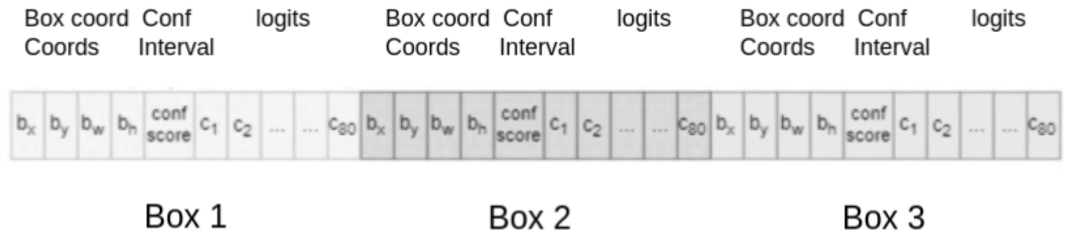
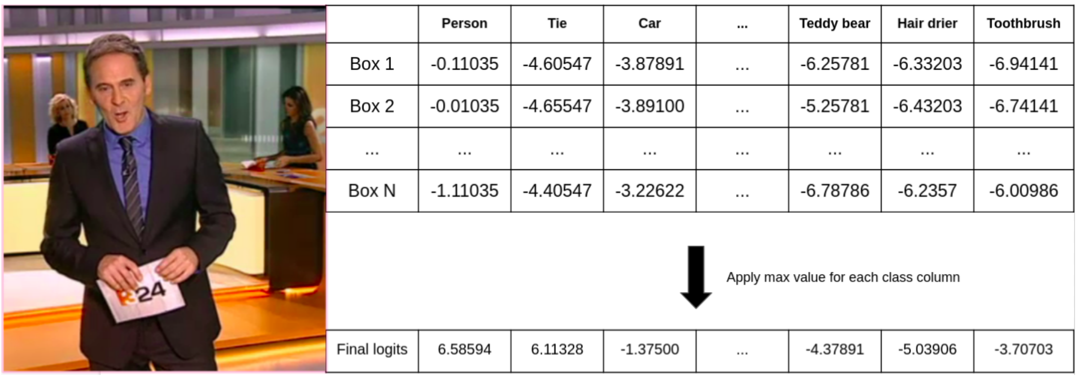

Box coord Coords
Conf Interval
logits
Box coord Coords
Conf Interval
logits
Box coord Coords
Conf Interval
logits
b_x b_y b_w b_h conf score c_1 c_2 ... ... c_80
b_x b_y b_w b_h conf score c_1 c_2 ... ... c_80
b_x b_y b_w b_h conf score c_1 c_2 ... ... c_80
Box 1
Box 2
Box 3

FIGURE 5.5: Example of one output vector from YOLO

To solve and obtain a unique vector of 80 logits for each image, we have decided to consider all initial candidate boxes and extract all their logits. Compared to the main step inside the YOLO algorithm, in which it applies what is called *Non-Max Suppression*, we have decided to choose the max logit value for each class label. To illustrate that step, we can take a look to Figure 5.6. As we can appreciate, person and tie are two objects from the frame, so their logit values are high compared to the rest.



	Person	Tie	Car	...	Teddy bear	Hair drier	Toothbrush
Box 1	-0.11035	-4.60547	-3.87891	...	-6.25781	-6.33203	-6.94141
Box 2	-0.01035	-4.65547	-3.89100	...	-5.25781	-6.43203	-6.74141
...	...	...	...	...	...	...	...
Box N	-1.11035	-4.40547	-3.22622	...	-6.78786	-6.2357	-6.00986

Apply max value for each class column

Final logits	6.58594	6.11328	-1.37500	...	-4.37891	-5.03906	-3.70703

FIGURE 5.6: Extracting the max logit from all initial box candidates

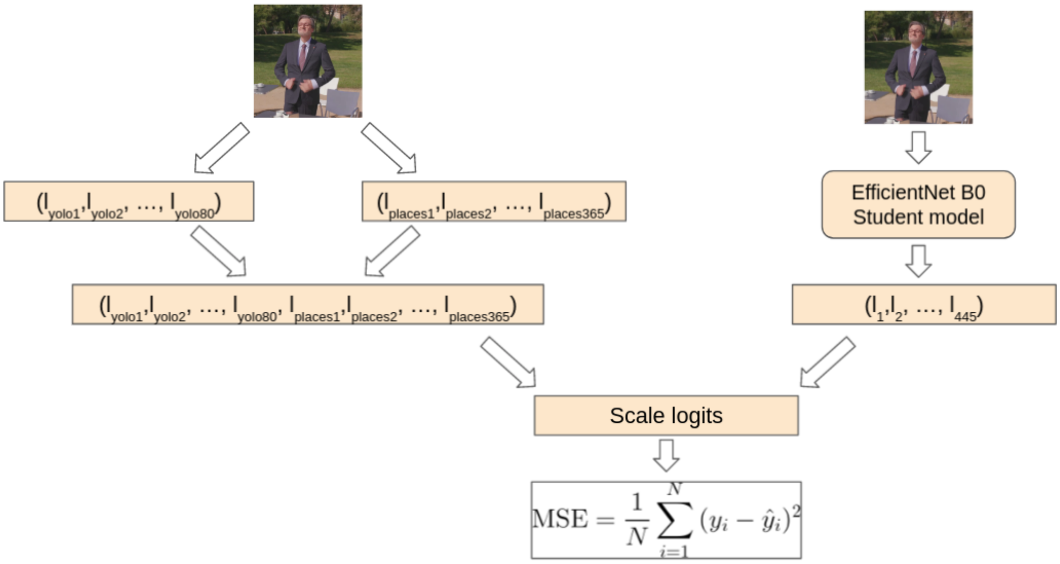
On the other hand, we have extracted the logits from Places365 in a more easy way. After passing the image through a Resnet architecture, we have obtained the 365 logits. In that case, logits and probabilities have dependant distributions, meaning that if we sum all the probabilities we get as result 1.

5.1.2 Training the student

- Create a Pytorch Dataloader with $batchSize = 16$.
- Create the student model from EfficientNetB0 changing the output layer to 445.
- GPU paralellization with nn.DataParallel().
- 10 epochs.
- Learning rate 0.001 with learning rate decay of 0.5.
- MSE loss.
- Stochastic Gradient Descent optimizer with $momentum = 0.9$.
- Propagate the loss (see Figure 5.7 and 5.8 for the variations) and do the backpropagation step.

To train the student model we have followed two different pipelines. In Figure 5.7, as we are merging all the logits in the same vector, we have scaled yolo logits to the Places domain because the different between both vector values is significant and can cause some unbalance towards Places learning.

After the scaling, we compute the RMSE to predict the student outputs and we update the model until finishing all the epochs.



$(l_{yolo1}, l_{yolo2}, \ldots, l_{yolo80})$
$(l_{places1}, l_{places2}, \ldots, l_{places365})$
$(l_{yolo1}, l_{yolo2}, \ldots, l_{yolo80}, l_{places1}, l_{places2}, \ldots, l_{places365})$
EfficientNet B0
Student model
$(l_1, l_2, \ldots, l_{445})$
Scale logits
$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

FIGURE 5.7: Using RMSE on merged logits

The formula we have used to scale logits is the following one:

$$scaled\ logits = \frac{x - min(x)}{max(x) - min(x)} \cdot [max(t) - min(t)] + min(t) \tag{5.1}$$

where x are the logits from YOLO and t the logits from Places. By doing that normalization, we can balance the range of values from YOLO.

On the other side, Figure 5.8 show another approach. We have divided every output and we have computed a YOLO loss and a Places loss and we have computed the sum of both before backpropagation.

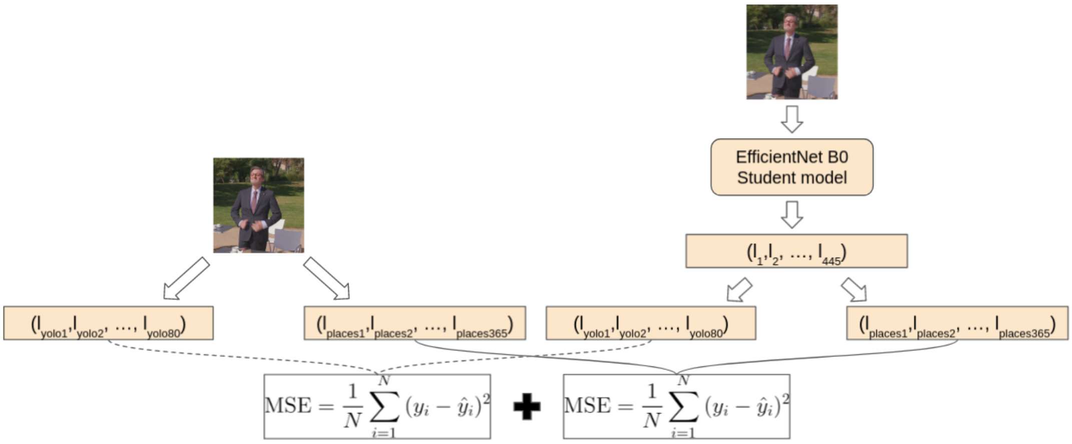

EfficientNet B0
Student model
$(l_1, l_2, \dots, l_{445})$
$(l_{yolo1}, l_{yolo2}, \dots, l_{yolo80})$
$(l_{places1}, l_{places2}, \dots, l_{places365})$
$(l_{yolo1}, l_{yolo2}, \dots, l_{yolo80})$
$(l_{places1}, l_{places2}, \dots, l_{places365})$
$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

FIGURE 5.8: Using RMSE on splitting logits

5.2 Second approach: YOLO, Places on COCO and Places dataset

After seeing the problems of implementing the student directly from the TV3 dataset, we have decided to take a image subset from COCO and another one from Places2 to create a new more balanced dataset. We have downloaded a total of:

- 118287 images from COCO.
- 36500 images from Places2.

After having the dataset, we have followed the same steps such as in section 5.1.2 to train the student model. This time, we have followed the approach of summing the two splitted losses. After 10 epochs, we have discovered that the student could train even more, so we have added 50 more epochs to the initial 10. After testing a bit some outputs we have discovered that for the part of Places it has been learning and having a better performance, meanwhile for the YOLO part the performance has been poor. The problem has been that Places loss is huge compared to the YOLO loss and has decreased a lot compared to the YOLO one. To solve it, we have balanced each loss before computing the total sum (see Figure 5.9) to force the student to learn better about the YOLO logits.

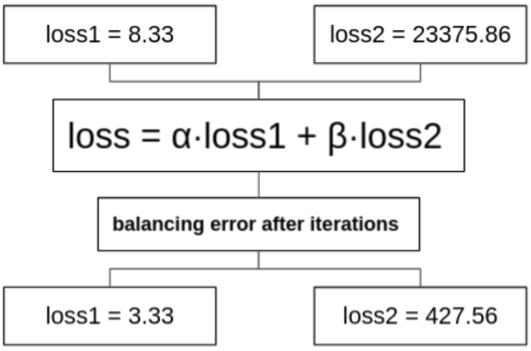


FIGURE 5.9: Balancing student loss with α and β

5.3 Evaluation

To evaluate all our results from the TV3 problem, we cannot extract any supervised metric such as MNIST accuracies. The dataset from TV3 are images without labeling, obviously due the cost of doing the annotation step. We have decided to take as groundtruth the extracted labels and values from YOLO and Places and we have computed ranking metrics to rank the list of classes based on their relevance in a each task.

5.3.1 First approach: take the most significant logit

From the approach we are facing, we can say we are facing a semi-supervised learning in some manner. We have decided to extract the class label with maximum logitf from the original groundtruth and from the output prediction of the student model. What we can achieve with that, is to check if atleast, the most relevant item of each image is significant enough for our student network. From that, we have computed an accuracy to have a more quantitative result.

5.3.2 Second approach: Normalized Discounted Cumulative Gain

From ("Discounted cumulative gain", Wikipedia), Normalized Discounted cumulative gain (nDCG) is a measure of ranking quality. In information retrieval, it is often used to measure effectiveness of web search engine algorithms or related applications. Using a graded relevance scale of documents in a search-engine result set, DCG measures the usefulness, or gain, of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks.

Two assumptions are made in using it:

- Highly relevant documents are more useful when appearing earlier.
- Highly relevant documents are more useful than marginally relevant documents, which are in turn more useful than non-relevant documents.

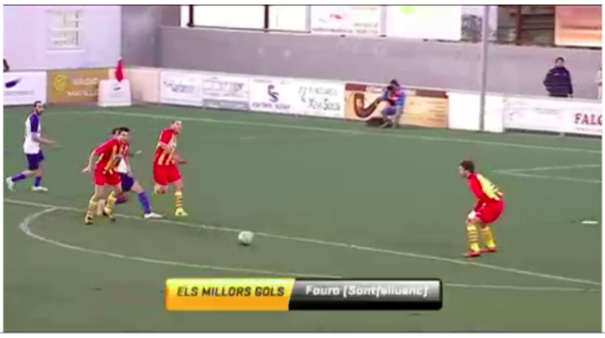
$$DCG_p = \sum_{i=1}^{P} \frac{rel_i}{log_2(i+1)} = rel_1 + \sum_{i=2}^{P} \frac{rel_i}{log_2(i+1)} \tag{5.2}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{5.3}$$

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{5.4}$$

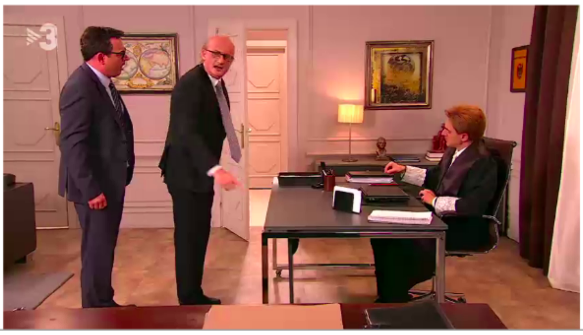
5.3.3 Results

Results on TV3 dataset



YOLO					
Original	Person	Sports ball	Clock	Chair	Tennis racket
Predicted	Person	Tie	Chair	Book	Clock
Places					
Original	Soccer field	Stadium/ soccer	Athletic_field/ outdoor	Stadium/ football	Basketball court
Predicted	Television studio	Office	Conference center	Science museum	Art gallery

FIGURE 5.10: Example on a football field



YOLO					
Original	Person	Tie	Chair	Dining table	Book
Predicted	Person	Tie	Chair	Book	Clock
Places					
Original	Art studio	Office	Home office	Art gallery	Reception
Predicted	Television studio	Office	Conference center	Science museum	Art gallery

FIGURE 5.11: Example on a TV studio

Figures 5.10 and 5.11show the main problem of training a student directly from TV3 images. As we can see, the predictions are biased towards TV studios, offices, i.e, to closed scenarios, which are the most frequent landscapes. If we try to predict a output scenario, it fails always due that imbalanced problem. Respect to TV3 we do not have a test set, and even if we split in a typical train/test split, we should make it for programs and try to balance everything, so we have decided to just extract quantitative results from COCO-Places.

Results on COCO-Places train dataset

Max logit approach:

- Label accuracy given max logit YOLO: **49%**.
- Label accuracy given max logit Places: **87%**.

nDCG approach:

- YOLO top-5 score: **0,43**.
- YOLO top-10 score: **0,37**.
- Places top-5 score: **0,98**.
- Places top-10 score: **0,98**.

Results on COCO-Places test dataset

Max logit approach:

- Label accuracy given max logit YOLO: **49%**.
- Label accuracy given max logit Places: **87%**.

nDCG approach:

- YOLO top-5 score: **0,41**.
- YOLO top-10 score: **0,34**.
- Places top-5 score: **0,98**.
- Places top-10 score: **0,98**.

Results on COCO-Places using as test the TV3 dataset

Max logit approach:

- Label accuracy given max logit YOLO: **72%**.
- Label accuracy given max logit Places: **3%**.

nDCG approach:

- YOLO top-5 score: **0,53**.
- YOLO top-10 score: **0,43**.
- Places top-5 score: **0,025**.
- Places top-10 score: **0,04**.

Chapter 6

Conclusions and future work

6.1 Conclusions

6.1.1 Conclusions on transferring knowledge on MNIST

First of all, we can conclude that we have been able to atleast reproduce each method and see how the estimated q values converge in a general way. We have seen that there is no problem on training different teacher configurations and get almost perfect results given a test set for each of these teacher configurations.

After studying how affects having and having less common labels until having anything in common, we have seen that the UHC approach is not viable to tackle the TV3 problem. We have demonstrated that soft labels/logits are more robust than hard labels and from the methods that estimate logits, the better is the Matrix Factorization in Logit Space, coinciding with the UHC paper.

6.1.2 Conclusions on TV3

We can conclude that if we use YOLO and Places individually, we do not have any problem to achieve a strong performance, but this is the opposite of what we wanted to achieve due the hardware and legal problems. We have also checked that our approach on taking the max of all candidate boxes in YOLO work because the outputs we extract are very good, in case we could solve hardware problems someday.

From the TV3 dataset, we have concluded that the dataset is not well balanced in terms of frames, specially for Places dataset because most of the frame scenes are indoor and in television studios. It does not mean that the student has not learn properly, because if we take the top 5 results and we compare it with the YOLO and Places groundtruth, we can be sure we would get very good results. The problem here is how we have thought about the problem. If we train a student on the TV3 dataset, we are training a model that cannot generalize, and we want something more robust at the end, that is why we have swapped to the COCO and Places dataset as training set.

6.1.3 Conclusions on COCO-Places datasets

From this point, we have faced the problem of having very unbalanced values between the ground truths. We have tried another scaling approach by adding a factor to each individual loss before summing them, but from what we have seen, or we need more and more epochs to balance it, or we think about another approach as future work.

Even though we have used another dataset to generalize, we have not been able to get enough samples to have a better class distribution because we have seen from results that we have faced the same unbalance problem. After testing with another dataset and obtaining very poor results, it is logic, and we have shared it that if we take the TV3 dataset as test set, the results are even worse as we have not finished to refine the training model, we need a lot of epochs in order to see if there is any possibility to make it work.

6.2 Future work

6.2.1 Datasets

One of the main issues during this project has been the TV3 dataset. Images have been given to us from a simple sampling of frames. We propose as future work to apply an algorithm of scene detection that is able to recognize the transition between frames. Principally, frame detection change between two adjacent frames simply requires the computation of an appropriate continuity or similarity metric. We assume it will not be the definitive solution, but atleast we think it can provide more base quality to the TV3 dataset.

For the COCO and Places dataset, we should follow sample images from the dataset taking into account a more balanced outcome. Assuming we have enough memory space and hardware computation, we could use the whole training dataset from each of them to teach the student in a more polite way.

6.2.2 Training procedure

We have faced some issues regarding value magnitudes. We would like to merge both approaches of merging the logits and scale them in a correct way, we think it can solve part of the imbalance because of how places benefits from low YOLO values.

Then, we can still add more and more epochs to the training phase until we reach a convergence to do an early stopping.

Another thing that we could research about is to fit another loss criteria inside the training.

6.2.3 Optimization formula

We could go back to UHC and study if there is a way to deal with YOLO outputs. At the end, if we go back to the formulas on 1.3.3, we understand that part of the restrictions are made for dependant probabilities.

Appendix A

Contribution of each member

During all the time we have developing the master thesis, we have been contributing and helping each other on every topic we have analyzed, developed or coded. But, if we have to specify who has lead every topic, the distribution is the following one:

- Unifying Heterogeneous Classifiers
  - Paper read by both of us.
  - Noel: Method 1 and coding part of method 1.
  - Aitor Method 2 and coding part of method 2.
- Yolo and Places
  - Noel: analyzed Yolo and coding part.
  - Aitor: analyzed Places and coding part.
- Pytorch code
  - Developed and revised by both of us.
- Master thesis report
  - Everything mentioned topic from above has been written by the assigned member but revised by both.
  - Remaining parts have been written and revised by both.

Appendix B

Link to the Github Repository

https://github.com/aitorlucasc/uhc_distillation/

Appendix C

Supplementay material on UHC methods

To check the math development of the used methods and the pseudocode, feel free to check (Jayakorn Vongkulbhisal and Visentini-Scarzanella, 2019b).

Bibliography

Daniele Foffano, Pradyot Patil and Renzo Scholman (2021). “Reproduction of Distilling the Knowledge in a Neural Network”. In: *Github*. URL: https://pradyot-09.github.io/Deep-Learning-Reproducibility-Project/.

“Deep Residual Learning for Image Recognition” (2015). In: *CVPR*. URL: https://arxiv.org/pdf/1512.03385v1.pdf.

“Discounted cumulative gain”. In: *WIkipedia* (). URL: https://en.wikipedia.org/wiki/Discounted_cumulative_gain.

“EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” (2019). In: *CVPR*. URL: https://arxiv.org/pdf/1905.11946.pdf.

Fisher Yu Ari Seff, et al. (2015). “LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop”. In: *CVPR*. URL: https://arxiv.org/pdf/1506.03365.pdf.

Geoffrey Hinton, Oriol Vinyals and Jeff Dean (2015). “Distilling the Knowledge in a Neural Network”. In: *NIPS*. URL: https://arxiv.org/pdf/1503.02531.pdf.

“Going deeper with convolutions” (2014). In: *CVPR*. URL: https://arxiv.org/pdf/1409.4842v1.pdf.

“ImageNet Classification with Deep Convolutional Neural Networks” (2012). In: *NeurIPS*. URL: https://www.cs.toronto.edu/~kriz/imagenet_classification_with_deep_convolutional.pdf.

Jayakorn Vongkulbhisal, Phongtharin Vinayavekhin and Marco Visentini-Scarzanella (2019a). “Unifying Heterogeneous Classifiers with Distillation”. In: *CVPR*. URL: https://openaccess.thecvf.com/content_CVPR_2019/papers/Vongkulbhisal_Unifying_Heterogeneous_Classifiers_With_Distillation_CVPR_2019_paper.pdf.

— (2019b). “Unifying Heterogeneous Classifiers with Distillation: supplementary material”. In: *CVPR*. URL: https://openaccess.thecvf.com/content_CVPR_2019/supplemental/Vongkulbhisal_Unifying_Heterogeneous_Classifiers_CVPR_2019_supplemental.pdf.

Jia Deng Wei Dong, et al. (2009). “ImageNet: A large-scale hierarchical image database”. In: *IEEE*. URL: https://ieeexplore.ieee.org/document/5206848.

“Josef Steppan”. In: *Wikipedia* (). URL: https://commons.wikimedia.org/wiki/File:MnistExamples.png.

Joseph Redmon Santosh Divvala, Ross Girshick and Ali Farhadi (2015). “You Only Look Once: Unified, Real-Time Object Detection”. In: *CVPR*. URL: https://arxiv.org/pdf/1506.02640.pdf.

“Places2 - CSAIL MIT”. In: *MIT* (). URL: http://places2.csail.mit.edu/.

“Proyecto PICAE: Publicación Inteligente de Contenidos Audiovisuales y Editoriales”. In: *website* (). URL: https://comunitatmedia.cat/es/portfolio/proyecto-picae/.

“Pytorch-Wikipedia”. In: *Wikipedia* (). URL: https://en.wikipedia.org/wiki/PyTorch.

“Release of Places365 - CNNs”. In: *Github* (). URL: https://github.com/CSAILVision/places365.

"Televisió de Catalunya - CCMA". In: *website* (). URL: https://www.ccma.cat/tv3/.

Tsung-Yi Lin Michael Maire, et al. (2014). "CVPR". In: *website*. URL: https://arxiv.org/pdf/1405.0312.pdf.

"Very Deep Convolutional Networks for Large-Scale Image Recognition" (2014). In: *CVPR*. URL: https://arxiv.org/pdf/1409.1556v6.pdf.