INFO8010: Knowledge Distillation

Nicolas THOU,[1] **Bilel GUETARNI**,[2] and **Robin ROEMER**[3]

[1] *nicolas.thou@student.uliege.be (s198327)*
[2] *bilel.guetarni@student.uliege.be (s198372)*
[3] *robin.roemer@rwth-aachen.de (s198394)*

I. INTRODUCTION

Running Neural Networks on mobile devices enables a wide field of new opportunities. With scene analysis and object detection on mobile devices the Augmented Reality technology could become cheap and affordable. Furthermore, in the surveillance industry and for Smart Cities edge computing is gaining traction. This is mainly due to privacy issues, since the videos themselves should not be send via a network but directly processed on the camera device. Both applications share the necessity to make networks smaller and less computation heavy during run time. One approach to reduce a Deep Neural Network in size is known as Knowledge Distiallation (KD).

For the object recognition task, a large amount of highly redundant data has to be processed to learn the model structure. Huge amount of computational resources are essential to train Deep Neural Networks. Training however, does not need to operate in real time and we have enough computational resources. For the deployment on an mobile device afterwards, we need to reduce the network size to make inference with fewer resources possible. To achieve that, the acquired knowledge from the deep model needs to be transferred to a smaller network.

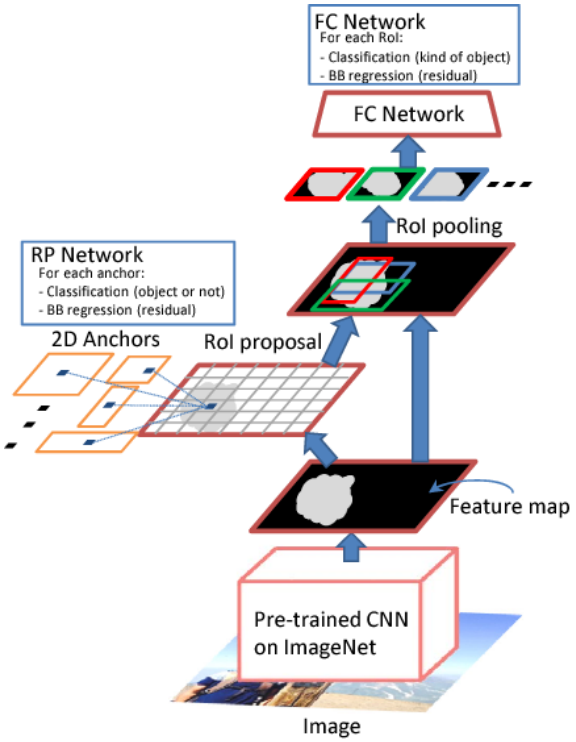


FIG. 1: Faster R-CNN [2]

II. RELATED WORK

In this chapter we present the related work. We will summarize the architecture of Faster R-CNN, we add an additional explanation for Feature Pyramid Network and explain the idea of KD for the image classification problem. Finally, we will explain an improvement, which is possible due to hints and the intermediate representation and make KD usable for object detection.

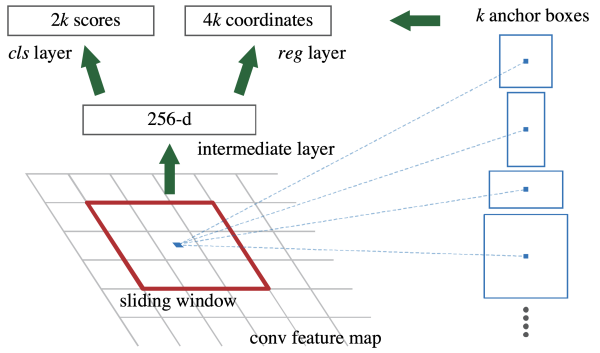
A. Faster R-CNN

Faster R-CNN is an object detection system composed of three parts. First, we feed the entire image (input) into a Convolutional Neural Network (CNN) similar to the Zeiler and Fergus model or VGG to obtain a high resolution feature map, this is used as a feature extractor. Secondly, we feed this feature map to the Region Proposal Network (RPN). The RPN outputs a set of rectangular object proposals, each with an objectness score [1]. Each object proposal is then fed into the Regression Classification Network (RCN), which is a Fully Connected Network (FCN) after passing a Region of Interest (RoI) pooling layer.

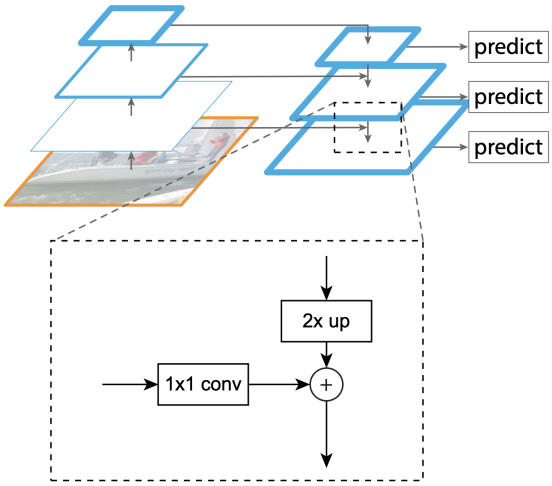
1. *Region Proposal Network (RPN)*

For the generation of regions of interest (or region proposals), we slide a window (kernel) over the feature map (output of the CNN). This sliding window is implemented by a $n \times n$ convolutional layer and is followed by two 1×1 convolutional layers for regression and classification, respectively.

An anchor box is considered to be a "positive" sample, if it has the highest Intersection of Union (IoU) with a ground truth box or has an IoU greater than 0.7 with any ground truth box. The same ground truth box can cause multiple anchors to be assigned positive labels. On the other hand, an anchor is labeled "negative" if its IoU with all ground truth boxes is less than 0.3. The other anchors (neither positive nor negative) are disregarded



(a) Sliding window for the RPN



(b) A building block illustrating the lateral connection and the top-down pathway, merged by addition. [3]

FIG. 2

for RPN training.

For each sliding window location (represented by the red square in Figure 2a), we predict k region proposals with different scale and ratio. So the regression layer has $4 * k$ outputs which represent the coordinates of k boxes. The classification layer outputs $2 * k$ outputs which represent the binary decision, weather there is an object or not. The number of anchor boxes k is a hyper-parameter. Each anchor is centered at the sliding window in question. For a convolutional feature map of size $W \times H$, there are $W * H * k$ anchor boxes (typically around 2,400) in total. [1]

2. *Feature Pyramid Networks (FPN)*

FPNs generate feature maps in a pyramid framework to detect features at different scales. It generates multiple feature map layers (multi-scale feature maps). There are two parts in the FPN, the bottom-up path, which is a classical CNN for feature extraction where the feature map is down-sampled where the semantic value increase but the resolution decreases and a top-down path which reconstructs a high resolution layer from a semantic rich layer. Indeed, we use a nearest neighbor upsampling (upsampling is a technique for increasing the size of an image, here by copying the value from the nearest pixel) during the top-down path.

Because of the upsampling during the top-down path, the object location is not accurate at all. Then, a lateral connection is added between the corresponding feature map and the reconstructed layers in order to better locate objects. The merging is performed by a 1×1 convolution layer on the feature map in the bottom-up path, which values are added element wise during the top-down path. Finally, a 3×3 convolution filter is used to avoid aliasing effect due to the nearest neighbors upsampling before the output.

FPN is placed between the feature extractor and the RPN in the architecture of Faster R-CNN. The sliding window is used on each feature map, which is produced by the FPN. RoIs of width w and height h are chosen from specific levels k of the pyramid by:

$$k = k_0 + log_2 \left(\frac{\sqrt{wh}}{\text{pre-training size}} \right)$$

k_0 is the target level on which the RoIs of pre-training size should be evaluated at and has to be chosen as a hyper-parameter. [3]

3. *Non Max Suppresion (NMS)*

After the RPN process we need to select the best region proposals because it costs a lot of computational resources to treat them all. The input to the NMS is a list of region proposal. We select the region proposal, let's call it A, that has the highest confidence score and compare it to all the other boxes using IoU. We compute IoU between each anchor boxes and A. We remove all the anchors boxes that has a greater IoU than an arbitrary threshold. We save A in the output list of the NMS. We iterate this process until we remove all the anchors boxes.

4. *Region of Interest (RoI) - Pooling Layer*

The RoI pooling layer uses max pooling to convert the features of any valid region into a small feature map with

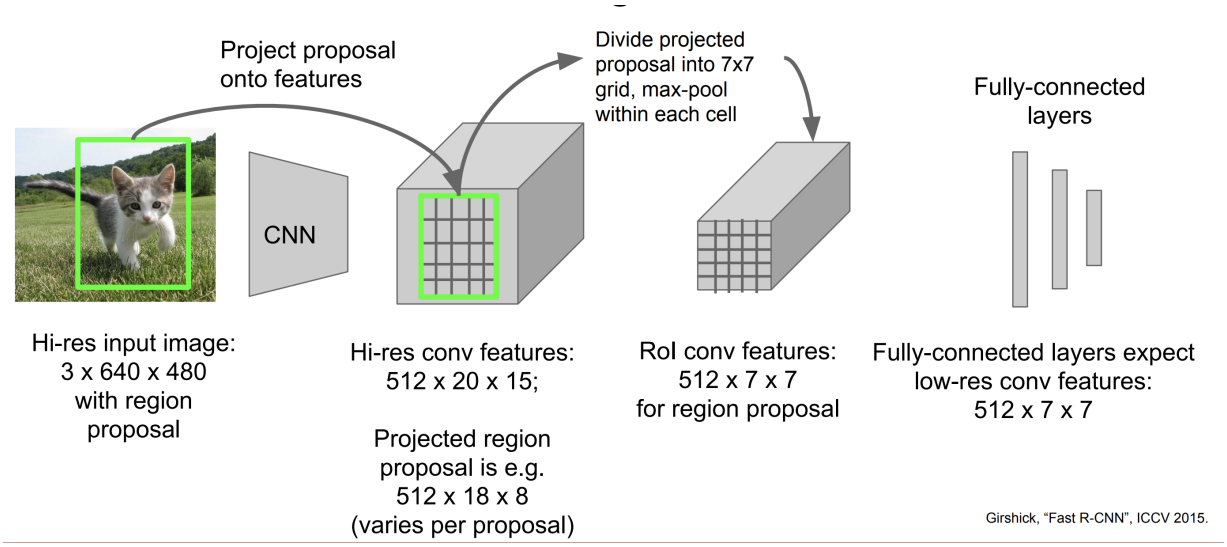


FIG. 3: Schema of the different steps before reaching the FCN [4]

a fixed spatial extent of $H \times W$ (e.g., 7×7). H and W are layer hyper-parameters.

RoI max pooling works by dividing the $h \times w$ region proposal window into an $H \times W$ grid of sub-windows of size $\frac{h}{H} \times \frac{w}{W}$ and then max-pooling the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map's channel, as in standard max pooling [5]. In figure 3 the region proposal (green) passes through an RoI max pooling layer in order to obtain a low resolution convolutional feature ($512 \times 7 \times 7$) just before the FCN

B. Knowledge Distiallation (KD)

The KD was introduced as a model compression framework, which eases the training of deep networks by following a student teacher paradigm, in which the student is penalized according to a softened version of the teacher's output.

For classification problems, we seek to maximize the average log probability of the correct answer. For wide and deep networks, they discriminate between a wide range of classes. However, the trained model assigns also a probability to wrong classes. This distribution over classes contains useful information, which can be forwarded to the student network. An image of a car for example has more common features with a truck, than with a flower. The distribution over classes can be seen as a rich similarity structure over the data. [6] We want to keep the information that the truck is similar to the car. Confidence score over the wrong classes can hide such information.

Even though we optimize an objective function in regards to our training data, the real aim is the optimization of the generalization for classification of future unseen data. KD tries to transfer the generalization knowledge to a smaller model (student). While it is hard to learn the generalization on a smaller model, transferring it from a large model (teacher) that has learned a good representation is a lot easier. This enables small models to perform almost the same, sometimes better as larger models. [6]

The relative probabilities are transferred into a soft target. This can be done by introducing a temperature in the final softmax scores - such temperature is a hyper-parameter usually set between 1 and 5. We can use the same training set, or a new set with unlabelled data to train the student network. We labelled the data with the prediction of the teacher. If the soft targets have high entropy, they represent the correlation in a rich way. Additionally, the soft targets induce less variance in the gradients between training data. Hence, the student model can be trained on less data than the teacher as well as using a higher learning rate. These properties are sometimes also used, to perform online training on edge devices and not only inference.

The training set of the student model can be unlabeled data or the original training set. The best performance can be achieved, if the objective is a combination of soft targets and the true label distribution. [6]

The higher the temperature is used, the more information is transferred. The temperature used by the teacher and the student has to be the same. However, after the training session, the temperature is set to 1 (classic softmax activation function). The loss function during the training of the student network consists of two parts. The first part is the cross entropy loss over the soft targets computed by the softmax temperature activation function. The second part is the cross entropy over the true label of the data. [6]

Finally, we summarize and formulate the concept of KD as below :

Let T be the teacher, a_T is the logits vector of the

teacher and $P_T = \sigma(a_T)$ the final output. Let S be the student with parameters W_S and output probability $P_S = \sigma(a_S)$, where a_S is the student's logits. Here, σ is the softmax function.

The student will be trained such that its output P_S matches the teacher's output P_T, as well as the true label y_{true}. The temperature $\tau > 1$ is introduced to soften the signal arising from the output of the teacher and preventing a single very high probability erasing all other probabilities close to zero. The same temperature is applied to the output of the student (P_S^τ), when it is compared to the teacher's soft targets (P_T^τ).

$$P_S^\tau = \sigma\left(\frac{a_S}{\tau}\right) \tag{1}$$

$$P_T^\tau = \sigma\left(\frac{a_T}{\tau}\right) \tag{2}$$

The student is then trained to optimize the following KD loss function

$$L_{KD}(W_S) = H(y_{true}, P_S) + \lambda H(P_T^\tau, P_S^\tau), \tag{3}$$

such that λ is a tunable parameter to balance between the true labels and the soft targets and H is the cross entropy loss function.

The drawback for this approach is, that the width of the teacher and student has to be the same so the reduction in size for the student is limited by the width of the teacher.

C. Hints for thin Deep Networks

The *Hints idea* for the training of a student uses not only the output of the teacher but also intermediate representations. Results confirm that having deeper student models generalize better, while making them thin reduces the size and therefor the computational complexity. The resulting networks are called *FitNets* in [4].

Teacher hint layer

A hint is defined as the output of a teacher's hidden layer responsible for guiding the student's learning process. The so called guided layer of the student learns from these hidden layer.

1. Convolutional Neural Network (CNN) for Regression

The output dimensions of the hint layer has to match with the output dimensions of the guided layer. A convolutional regressor is added [7], which is a convolutional layer with a non linear activation function e.g. Relu. In figure 4 the convolutional regressor is blue.

Here, the convolutional regressor is added and the weights are updated according to the weights of the teacher's hint layer. We update the weights until $W_{regressor}$ but we only keep the W_{guided}. Then, we remove the convolutional regressor and proceed to the Distillation framework.

III. METHOD

The distilled Neural Network is based on [7] namely Faster R-CNN. Our approach to tackle the project consists of the following steps:

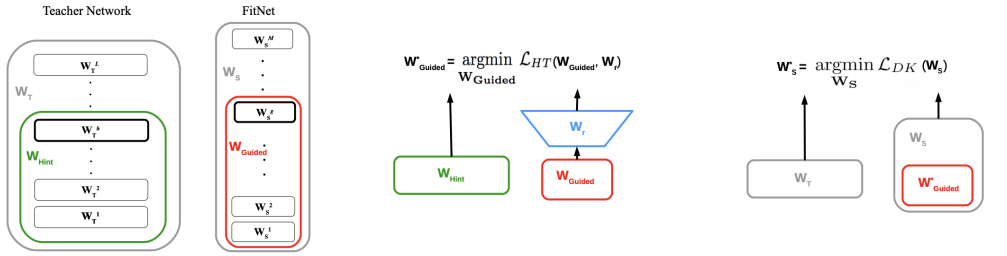
- As a teacher, a pre-trained Faster R-CNN (trained on COCO) with Resnet-101 as feature extractor is used and the argument of the activation function is adapted in order to have a SoftMax temperature as a final activation function and a cross entropy loss for the optimization.
- As a student a Faster R-CNN with a Resnet50 backbone is used.
- The three main loss functions are adapted to fit the presented KD to transfer knowledge from the teacher to the student. (Loss for classification (RPN, RCN), Loss for Regression (RPN, RCN), Hint Loss).
- We make predictions with the teacher's network (the prediction will be on the training set, we won't use unlabelled data as in [6]) and use those predictions for the student training.
- We use Stochastic Gradient Descent to optimize the student network (using the weight decay, momentum)

Using these, we train our student model using the MXNet python library.

A. Data

The student is trained on the COCO object detection data set with an additional class for the background resulting in 81 classes. During training, we use standard normalization for the pixels value and we use horizontal flip as data augmentation technique. We have to be careful on what is don to the training data because the teacher has to be able to make prediction on it.

There are three intermediate outputs at the RPN. First, we have targets. Those represent our region proposals and have shape (*batch size*, *# region proposals*). Secondly there is the target shape, which outputs the coordinates with shape (*batch size*, *# region proposals*, 4). The third output is a mask, which discards all region proposals for the background having the shape (*batch size*, *# region proposals*, 4).



(a) Teacher and Student Networks (b) Hints Training (c) Knowledge Distillation

FIG. 4: Training a student using hints [4]

Input: $\mathbf{W_S}, \mathbf{W_T}, g, h$
Output: $\mathbf{W_S^*}$
1: $\mathbf{W_{Hint}} \leftarrow \{\mathbf{W_T}^1, \ldots, \mathbf{W_T}^h\}$
2: $\mathbf{W_{Guided}} \leftarrow \{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\}$
3: Intialize $\mathbf{W_r}$ to small random values
4: $\mathbf{W^*_{Guided}} \leftarrow \underset{\mathbf{W_{Guided}}}{\text{argmin}}\ \mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r})$
5: $\{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\} \leftarrow \{\mathbf{W_{Guided}}^{*1}, \ldots, \mathbf{W_{Guided}}^{*g}\}$
6: $\mathbf{W_S^*} \leftarrow \underset{\mathbf{W_S}}{\text{argmin}}\, \mathcal{L}_{KD}(\mathbf{W_S})$

Algorithm 1: Algorithm for the Hint learning Framework

B. Classification Loss

$$L_{cls} = \mu L_{hard}(P_s, y) + (1-\mu) L_{soft}(P_s, P_t) \quad (4)$$

$$L_{hard}(P_s, y) = -E_y[log(P_s)] \quad (5)$$

(cross entropy loss),

$$L_{soft}(P_s, P_t) = -\sum w_c P_t log P_s \quad (6)$$

(class-weighted cross entropy),

$w_0 = 1.5$ for the background class and $w_i = 1$ for all the others classes.

$$P_s = \sigma\left(\frac{Z_s}{T}\right) \quad (7)$$

$$P_t = \sigma\left(\frac{Z_t}{T}\right) \quad (8)$$

Such that:

T is the temperature hyper-parameter.

μ is a hyper-parameter to balance the hard and soft losses.

P_s is the output of the student.

P_t are the teacher's soft targets.

Z_s is the final score output for the student.

Z_t is the final score output for the teacher.

C. Regression loss with teacher bounds

$$L_{reg} = L_{sL1}(R_s, y_{reg}) + \nu L_b(R_s, R_t, y_{reg}) \quad (9)$$

$$L_b(R_s, R_t, y_{reg} = \quad (10)$$

$$\begin{cases} \| R_s - y_{reg} \|_2^2, & \text{if } \| R_s - y_{reg} \|_2^2 + m > \| R_t - y_{reg} \|_2^2 \\ 0, & \text{otherwise} \end{cases}$$

$$L_{sL1}(t^u, y_{reg}) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - y_{reg,i}) \quad (11)$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1. \\ |x| - 0.5, & \text{otherwise.} \end{cases} \quad (12)$$

Such that:

m is the margin.

y_{reg} denotes the regression ground truth coordinates.

R_s is the regression output of the student.

R_t is the regression output of the teacher.

ν is a weight hyper-parameter.

Moreover, we have the bounding-box regression offsets $t^k = (t_x^k, t_y^k, t_z^k, t_t^k)$, for each of the K object classes, indexed by k. The tuple of true bounding-box regression targets for one class is given by $y_{reg} = (y_{reg,x}, y_{reg,y}, y_{reg,w}, y_{reg,v})$

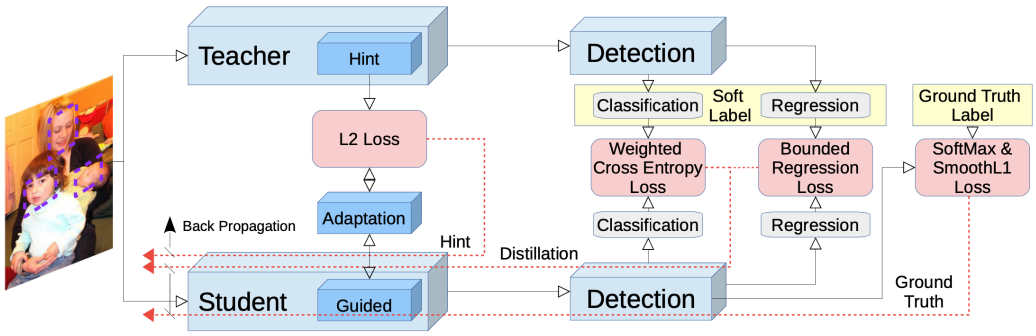


FIG. 5: Learning Framework according to [7] - there are 4 contributions to the loss. (i) from the hint layer, (ii) from the classification (iii) from the regression and from the ground truth. The backward pass is shown with a dotted red line.

Contrary to the KD for classification, the teacher's regression outputs can be misleading for the student if they are unbounded. The teacher's hint may lead in the opposite direction of the ground truth. In [7] it is proposed to use the teacher's regression output as upper bound for the the student. To better generalize, there is no additional loss for the student anymore after it performances are better than those of the teacher with a certain margin. [7]

D. Hint based learning loss

$$L_{hint}(V,Z) = \| V - Z \|_2^2 \tag{13}$$

or

$$L_{hint}(V,Z) = \| V - Z \|_1 \tag{14}$$

Such that :

Z represent the output of the teacher's hint layer.

V represent the output of the student's guided layer.

IV. DESCRIPTION OF THE ARCHITECTURE AND MAIN ALGORITHMS

This chapter describes the details of the implemented Faster R-CNN.

We decide not to create an actual training set for the student, since the Train Loader shuffles the data and we do not want to store the original data twice. Therefore, the student is trained while generating the soft targets from the teacher.

In terms of structure, Faster R-CNN is composed of a feature extraction network, a RPN (including its own anchor system, proposal generator), region-aware pooling layers, class predictors and bounding box offset predictors.

A. Forward pass of Faster R-CNN

The forward pass is based on the Hybrid Forward of the library GluonCV. We included the softmax temperature setting for the teacher and adapt its output to include the label distribution for each bounding box predicted by the teacher. Further, the training for the student is adapted to include the guided layer and soft targets.

B. Custom loss function

We implemented the loss according to chapter III. Only the "Hint based learning loss" was not implemented. The hyper-parameter μ and ν are set to 0.5. Moreover, the hyper-parameter m for the margin in the *regression loss with teacher bounds* is set to 0.5. This implies, that most of the time we trust the teacher more than the student, and if the student is a lot better than the teacher, then the teacher loss is set to 0.

V. PROBLEMS FACED DURING THE PROJECT

In GluonCV, there are two modes, prediction mode and training mode. That separation made it difficult to create the predictions from the teacher while training the student:

- It was hard to get a hand on the label distributions for every bounding box. By default, only the top-1 is shown.

Algorithm 2 Training algorithm for the student

Result: trained student
student ← FasterRCNN with Resnet50 backbone
teacher ← FasterRCNN with Resnet101 backbone {pre-trained model}
for batch ← COCO dataset **do**
  for image, ground_truth, RPN_targets ← batch **do**
    bboxes_teacher, score_teacher, RPN_box_teacher, RPN_score_teacher ← teacher.prediction(image)
    ground_truth_label, ground_truth_bounding_boxes ← ground_truth
    RPN_class_target, RPN_box_target ← RPN_targets
    bboxes_student, score_student, RPN_box_student, RPN_score_student ← student.prediction(image)
    rpn_loss_reg ← L_{reg}^{RPN}(RPN_box_student, RPN_box_teacher RPN_box_target)
    rpn_loss_class ← L_{cls}^{RPN}(RPN_score_student, RPN_score_teacher, RPN_class_target)
    rcn_loss_class ← L_{reg}^{RCN}(score_student, score_teacher, ground_truth_label)
    rcn_loss_reg ← L_{cls}^{RCN}(bboxes_student, bboxes_teacher, ground_truth_bounding_boxes)
  end for
  Apply SGD on the batch losses
end for

- The existing code base for the NMS, the RPN and the RoI pooling layer was difficult to understand. Around 70% of the project was to understand the library functions used in Faster RCNN.

- Training the model was not possible on Google Colab. We had to train it on a virtual machine during more than 24h.

- Behind KD there is a basic idea that deep and thin models has to be chosen for good distillation. However we couldn't build our own model, because not enough much time to study how to do it, so we have to deal with the library predefined models.

VI. RESULTS AND QUANTITATIVE ANALYSIS

Something that we didn't think about before starting is how much training data should be used for an object detector to be accurate. However, we trained two models, one using distillation and one without, on a total of 1400 images. Indeed, 1400 images are clearly not enough to train an object detector, but we couldn't do more due to computation limitations. We reported the training plots in the appendix I and J to have an idea of how things started. Note that to train them on 1400 images, it took us around 24 hours.
We trained the models using the Stochastic Gradient Descent algorithm with a constant learning rate of 10^{-3} and using regularization techniques as weight decay with $\lambda = 5.10^{-4}$ and momentum with $\beta = 0.9$. After facing gradient exploding we also added gradient clipping with a norm set to 1.

We have to notice the fact that the distillation model doesn't train with the hint layer of the teacher. Moreover, the FasterRCNN model used for the student model doesn't include the Feature Pyramid Network to detect small object.

As we can see in the loss plots (Appendix I & J), both models seems to behave the same way for the first images. Indeed, the performance doesn't change a lot due to the poor quantity of data used. Even if the training was short, we decided to test them on the validation data set of COCO. We reported some examples of the test in the figure 8. We see that the distilled model can perform better on certain images compared to the standard model, in term of the object detected or the confidence score. But the standard model still better perform for other objects for example in the fourth image with the giraffes. We noticed that both model struggle for images containing other objects than person; maybe because the majority of images they were trained on contained mostly persons. We could also notice that the distilled model has taken the building in the second image for a train which is quite similar since it has windows and it's elongated. This shows that it has already understood that a train is an elongated object with windows and allows it to detect the train in the third image, contrary to the standard model. Considering that we didn't used hint learning; and this could fine-tuned the backbone for object detection; and we know from the ablation studies of [7] that hint learning reduces significantly under-fitting and improves generalization resulting in "improvement in both training and testing accuracy".

The student model tends to display more bounding boxes during prediction. However, sometimes those bounding boxes are not relevant. For instance, it can infer multiple bounding boxes while there is only one object. Moreover, we notice that the distilled model has higher confidence scores than the standard model. We insist on the fact that we trained the models over only 1400 images. The key point to takeaway is the trend of the distilled model to behave as well as the standard model, sometimes even better and unfortunately sometimes worst than the standard model. The goal was at the beginning to act like the teacher. We are still far from the results of the teacher. The main problem is that the training set doesn't include enough images.

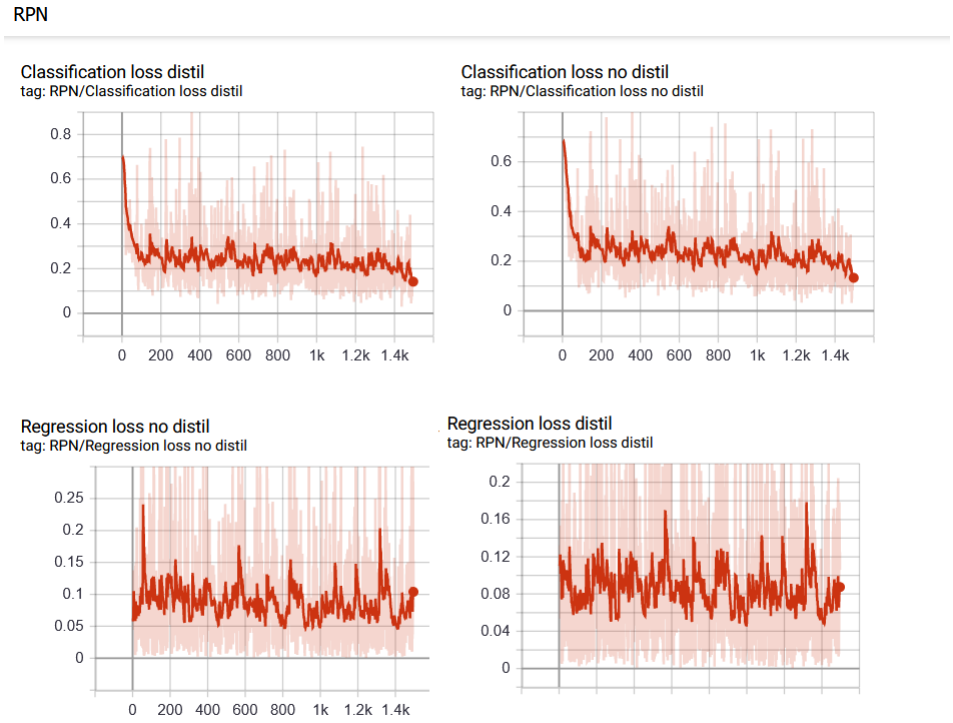


FIG. 6: Training loss of the RPNs part

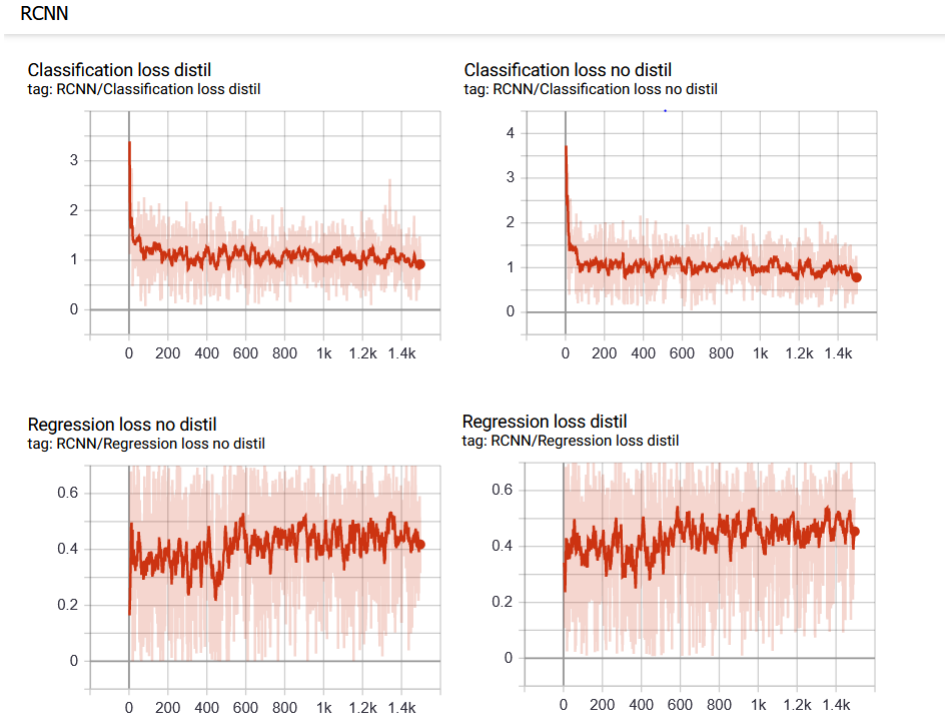


FIG. 7: Training loss of the RCNs part

FIG. 8: Predictions of distilled (left) and not distilled (right) models

VII. DISCUSSION

After the research and writing chapter III, the aim of the implementation was quite clear - implementing the correct loss function in order to train a student network. Still, while implementing the code, finding the right functions in the GluonCV library and defining proper hyper-parameters was challenging and time consuming. The first try, using the teacher's intermediate feature map directly for the loss function failed since the shape does not align. The second try, using this feature map as an argument in the forward pass and using this as ground truth did not work as well. So we didn't used the hint learning.

In further work we propose to add the hint learning in order to have a better accuracy from the student and use more data during training. We can also test different optimization algorithms and hyper-parameters. Furthermore, we can try to code Faster RCNN from scratch, then modify the width of the student model (make it thinner than the teacher) for the extract features, and also the depth (make it deeper for better generalization).

REFERENCES

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[2] Hirotaka Hachiya, Yuki Saito, Kazuma Iteya, Masaya Nomura, and Takayuki Nakamura. 2.5 d faster r-cnn for distance estimation. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3999–4004. IEEE, 2018.

[3] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[4] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[5] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[7] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pages 742–751, 2017.

[8] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *Unpublished draft. Retrieved*, 3:319, 2019.

APPENDIX

A. General notions

Inference is the use of a network to make prediction on unseen data normally after the training is completed.

Intersection of Union (IoU) is the ratio between the area of the overlap between a ground truth bounding box and the predicted bounding box and the are of the union so the sum of the area of both bounding boxes minus the overlap.

B. Squared Norm Regularization

The regularization technique is used in the loss function by adding a penalty term. The most common method for ensuring a small weight vector is to add its norm as a penalty term to the loss. If the weight vector grows larger, the penalty $\| w \|^2$ grows and a further growth for higher accuracy is traded off with this penalty [8]. This trade-off is controlled by a hyper-parameter and has to be tuned to guarantee a good performance of the Stochastic Gradient Descent (SGD).

C. How to choose the guided and hint layer?

The guided layer should predict the output of the hint layer. Hints are a form of regularization and thus, the pair hint/guided layer can over-regularize the student's training. The deeper the guided layer is placed, the less flexible is the student. Therefore, *FitNets* are more likely to suffer from over-regularization. In our case, we choose the hint to be the middle layer of the teacher. Similarly, we choose the guided layer to be the middle layer of the student.

D. Class imbalance problem

The problem of imbalanced classes occurs when a data set contains more examples for a certain class. Many classification learning algorithms have low predictive accuracy for the infrequent class.

E. Saddle point

A saddle point is neither a local nor a global minima. Optimization might stall at the point, even though it is not a minimum and lead to the vanishing gradient problem. [8]

F. Resnet-50 and Resnet 101

Resnet is a feature extractor that has the particularity to be really deep and use skip connection (also called short cut) in order to avoid the vanishing gradient problem. The vanishing gradient problem occurs in deep networks, since with the chain rule during back propagation a lot of gradients are multiplied. If only one of those gradients is very close to zero, the whole product becomes very small. It is the contrary of the exploding gradient problem where the value of certain gradients is very large and gives the main effect on the update since it is multiple magnitudes larger. Using residual blocks allow to train much deeper neural networks and to back-propagated the values until the earlier layer without becoming too small. Resnet is a chain of residual blocks.

G. Entropy - information theory

Entropy is a measure of disorder or uncertainty and one of the goals of machine learning models in general is to reduce uncertainty. Entropy is a metric for random variables of continuous or discrete distributions. The context of the definition of *Information* has an principle axiom: The information we gain by observing two random variables is no more than the sum of the information we gain by observing them separately. If they are independent, then it is exactly the sum. Indeed, for a random variable X following the probability distribution P, the expected amount of information through entropy (or Shannon entropy) is

$$\begin{aligned} H(X) &= E_{x\sim P}[I(event_i)] \\ &= -E_{x\sim P}[log(p(X)] \\ &= -\sum_i p_i log(p_i) \end{aligned} \tag{15}$$

Such that $p_i = P(X)$ and $p(X)$ are the density functions.