Structural Knowledge Distillation for Object Detection

by

Philip J.T.M. de Rijk

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday June 22th, 2022 at 15:00.

Student number:	4375017	
Thesis committee:	Prof. Dr. ir. Dariu M. Gavrila	TU Delft, Academic supervisor
	Dr. Julian F.P. Kooij	TU Delft
	Dr. Lukas Schneider	Mercedes-Benz AG

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Any sufficiently advanced technology is indistinguishable from magic.

ARTHUR C. CLARKE

Structural Knowledge Distillation for Object Detection

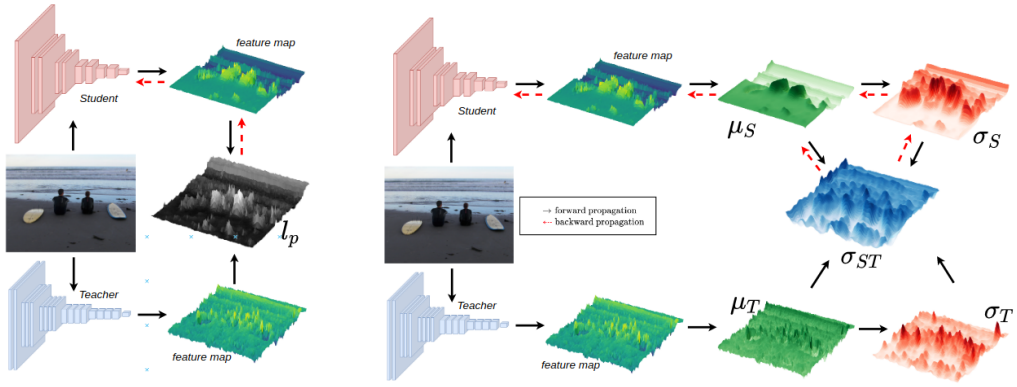
Abstract

Knowledge Distillation (KD) is a well-known training paradigm in deep neural networks where knowledge acquired by a large teacher model is transferred to a small student. KD has proven to be an effective technique to significantly improve the student's performance for various tasks including object detection. As such, KD techniques mostly rely on guidance at the intermediate feature level, which is typically implemented by minimizing an ℓ_p-norm distance between teacher and student activations during training. In this paper, we propose a replacement for the pixel-wise independent ℓ_p-norm based on structural similarity (SSIM) [30]. By taking into account additional contrast and structural cues, more knowledge within intermediate feature maps can be preserved. Extensive experiments on MSCOCO [17] demonstrate the effectiveness of our method across different training schemes and architectures. Our method adds only little computational overhead, is straightforward to implement and at the same time it significantly outperforms the standard ℓ_p-norms. Moreover, more complex state-of-the-art KD methods [14, 35] using attention-based sampling mechanisms are outperformed, including a +3.5 AP gain using a Faster R-CNN R-50 [23] compared to a vanilla model.

1 Introduction

Over the last decade, Deep Neural Networks (DNNs) have shown to be a very effective tool in solving fundamental computer vision tasks [15]. One major application of DNNs includes real-time perception systems found in *e.g*. autonomous vehicles, where object detection is often a task of major importance. Deployment of DNNs into real-time applications, however, introduces strict limitations on memory and latency. On the other hand, increased performance of state-of-the-art detectors typically comes with an increase in memory requirements and inference time [13]. Thus, the choice of network model and it's according detection performance is strictly limited. Several techniques have been proposed to tackle this problem, *e.g*. pruning [9], weight quantization [10], parameter prediction [6] and Knowledge Distillation (KD) [12]. In this work, we are particularly interested in the latter, as it provides an intuitive way of performance improvement without the need for architectural modifications to existing networks.

With KD, the knowledge acquired by a computationally expensive teacher model is transferred to a smaller student model during training. KD has proven to be very effective in tasks such as classification [12], segmentation [20], and in particular has seen considerable progress in object detection very recently [5, 8, 14, 35, 37]. Due to the complexity of the output space of a typical detection model, it is necessary to apply KD at the intermediate feature level, as solely relying on output-based KD has proven ineffective [3, 5, 8, 14, 16, 26, 35, 37]. In feature-based KD, a training objective is introduced in addition to existing objectives, which minimizes the *error* between teacher and student activations and is de-facto standard defined by the ℓ_p-norm distance [5, 8, 14, 26, 35, 37]. The ℓ_p-norm however ignores three important pieces of information present in the feature maps: (i) spatial relationships between features, (ii) the correlation between the teacher and student features and (iii) importance of individual features. We notice recent work has (implicitly) focused on bypassing the latter point through mechanisms that sample features by assuming that object regions are more "knowledge-dense" [5, 14, 26, 37]. However, as shown by [8], even distilling only background can improve performance, therefore it cannot be assumed that only the object regions contain



(a) Previous methods [26, 35, 37, 8, 5]. After sampling features an ℓ_p-norm is applied between feature maps.

(b) Our proposed method. We distill relational knowledge in the form of local mean μ, variance σ^2 and furthermore cross-correlation σ_{ST} between feature spaces.

Figure 1: Feature-based Knowledge Distillation.

useful knowledge. These sampling mechanisms furthermore introduce additional drawbacks which may limit their broader implementation into real-world applications, *e.g*. the need for labeled data [8, 14, 26].

In this work, we propose "Structural Knowledge Distillation", which aims to improve the downsides associated with the ℓ_p-norm as a central driver for KD methods, rather than designing an ever more sophisticated sampling mechanism. Our key insight is that similar to images, the feature space can be locally decomposed into luminance, contrast and structure components, a strategy that has seen successful application in the image domain in the form of structural similarity SSIM [30]. The new training objective becomes to minimize local difference in mean and variance, and maximize local zero-normalized cross-correlation between the teacher and student activations. We find that particularly the latter has great influence on distillation performance, as the underlying structure of relevant objects and background regions is propagated from the image throughout the CNN. Overall our contribution is as follows:

- We propose *structural distillation* which takes into account additional contrast and structure information in feature-based KD, based on SSIM [30].
- We illustrate through an analysis of the feature space that our method focuses on different areas than ℓ_p-norms and that solely sampling from foreground regions is sub-optimal as the entire feature space contains useful knowledge.
- We demonstrate the effectiveness of our method by performing extensive experiments using various detection architectures and training schemes. ℓ_{ssim} performs on par or outperforms carefully tuned state-of-the-art object sampling mechanisms [14, 35], fundamentally by replacing one line of code:

```
from kornia.losses import ssim_loss
def kd_loss(student_feats, teacher_feats):
    ### inputs have shape [B, C, H, W] ###
    # kd_feat_loss = torch.mse_loss(student_feats,teacher_feats) Changed!
    kd_feat_loss = ssim_loss(student_feats, teacher_feats, window_size=11)
    return kd_feat_loss
```

2 Related Work

Knowledge Distillation KD aims to transfer knowledge acquired by a cumbersome teacher model to a smaller student model. Bucilă et al. [1] (2006) demonstrate that the knowledge acquired by a large ensemble of models can be transferred to a single small model. Hinton et al. [12] (2015) provide a more general solution applied in a DNN in which they raise the temperature of the final softmax until the large model produces a suitably soft set of targets. Most KD research in the computer vision domain focuses on the classification task [25]. However, as our main interest lies in the real-time domain we focus on the more relevant object detection task.

Object Detection Object detection is one of the fundamental computer vision tasks, where speed and accuracy are often two key requirements. Object detectors can be classified into one-stage and two-stage methods, in this work we investigate our approach for both variations. The main meta-architecture within the one-stage domain is RetinaNet [18], with extensions including anchor-free modules and Reppoints [39, 34]. In the two-stage domain Faster R-CNN [23] is regarded as the most widely used meta-architecture, where a widely used iteration includes Cascade R-CNN [2]. Furthermore, regardless of architecture, ResNet [11] backbones are often used to extract features, which are furthermore fused at multiple scales using *e.g*. a FPN [18].

Knowledge Distillation for Object Detection Several methods have been proposed that use KD for object detectors, where it has been found that typically guidance at the intermediate feature level rather than the output is critical due to the complex nature of the output space in detection models [3, 5, 8, 14, 16, 26, 35, 37]. As the detection task requires the identification of multiple objects at different locations, a major complexity introduced is the imbalance between foreground and background, which manifests itself in the intermediate features. Typically, the assumption is made that object regions are "knowledge-dense", and background regions less so. As a result, recent work has implicitly focused on designing mechanisms which sample object-relevant features to distill knowledge from [5, 14, 16, 26, 37].

Li et al. [16] (2017) mimic the features sampled from the region proposals in a two-stage detector. Wang et al. [26] (2019) propose imitation masks which locate knowledge dense feature locations based on the annotated boxes. Dai et al. [5] (2021) propose a module which distills based on distance between classification scores. Similarly, Zhixing et al. [37] (2021) use output class probability to determine feature object probability. Recently Kang et al. [14] (2021) proposed a method in which they encode instance annotations in an attention mechanism [24] to locate "knowledge-dense" regions. Contrary to aforementioned methods, Zhang and Ma [35] (2021) propose a purely feature-based method in which they aim to both mimic the attention maps [38] as a sampling mechanism, and furthermore distill through non-local modules [27]. Regardless of the sampling technique, it has been demonstrated by Guo et al. [8] (2021) that it is not necessarily the case that background features are less important for distillation.

Objective Functions The objective in feature-based KD is to minimize the *error* between teacher and student feature spaces during training, typically in addition to existing objectives. The most widely used objective function in feature-based KD is the ℓ_p-norm with $p = 2$ [3, 5, 8, 16, 26, 35, 37], and less commonly $p = 1$ [3]. The ℓ_p-norm however, ignores the spatial relationships between features, the correlation between teacher and student and the importance of individual features, of which the latter has been the main focus of previous work. To take into account spatial dependencies, we need to furthermore compare features locally rather than pointwise. SSIM provides an elegant way to take into account spatial dependencies by making local comparisons of intensity and contrast, rather than just pointwise. It is further able to take into account the relationship between the teacher and the student by integrating zero-normalized cross-correlation. Contrary to alternative image signal quality metrics such as VIF [28], GMSD [28] and FSIM [33], SSIM is less complex to formulate mathematically and is differentiable, making it suitable as an objective function.

3 Method

3.1 Overview

We start off by defining the general form of feature-based distillation loss. For the purposes of this work, we divide a detector into three components: (i) the backbone, used for extracting features, (ii) the neck, for fusing features at different scales, and (iii) the head, for generating regression and classification scores. For feature-based KD, we select intermediate representations $\mathcal{T} \in \mathbb{R}^{C,H,W}$ and $\mathcal{S} \in \mathbb{R}^{C,H,W}$ from the teacher and student respectively at the output of the neck. The feature-based distillation loss between $\mathcal{T}$ and $\mathcal{S}$ can subsequently be formulated as:

$$\mathcal{L}_{feat} = \sum_{r=1}^{R} \frac{1}{N_r} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} \mathcal{L}_{\varepsilon} \left(\nu \left(\phi \left(\mathcal{S}_{r,h,w,c} \right) \right), \nu \left(\mathcal{T}_{r,h,w,c} \right) \right) \tag{1}$$

where H, W, C, R are the height, width, number of channels and number of neck outputs respectively, $N_r = HWC$ the total number of elements for the r-th layer. Additionally we define $\nu(\cdot)$ as a *normalization* function which maps the values of $\mathcal{T}$ and $\mathcal{S}$ to $[0, 1]$, and $\phi(\cdot)$ as an optional *adaptation layer* [3] which matches the dimensionality of $\mathcal{T}$ and $\mathcal{S}$, in our case a 1×1 convolutional layer. We introduce the shortened notation $\mathcal{L}_{\varepsilon}$ which represents the choice of *difference* measurement function at a single feature position r, h, w, c on normalized features and including the adaptation layer, *i.e.* $\mathcal{L}_{\varepsilon} = \mathcal{L}_{\varepsilon} \left(\nu \left(\phi \left(\mathcal{S}_{r,h,w,c} \right) \right), \nu \left(\mathcal{T}_{r,h,w,c} \right) \right)$. Accordingly, we use $\mathcal{S}$ and $\mathcal{T}$ to denote normalized and adapted student and normalized teacher activations respectively, e.g. $\mathcal{S} = \nu \left(\phi \left(\mathcal{S}_{r,h,w,c} \right) \right)$.

3.2 Measuring Difference

As ascertained, the de-facto standard choice for $\mathcal{L}_{\varepsilon}$ is the ℓ_p norm. $p = 2$ penalizes large errors, but is more tolerable to smaller errors. On the other hand, $p = 1$ does not over-penalize large errors, but smaller errors are penalized more harshly. The ℓ_p norm in its general form is given by:

$$\ell_p : \mathcal{L}_{\varepsilon} = \left(|\mathcal{S} - \mathcal{T}|^p \right)^{1/p} \tag{2}$$

Clearly such a function is not able to capture spatial relationships between features. In order to capture second-order information we need to involve at least two feature positions, we therefore change the problem statement from a *point-wise* comparison to a local *patch-wise* comparison. For each such patch, we extract three fundamental properties: the mean μ, the variance σ^2, and the cross-correlation $\sigma_{\mathcal{ST}}$ which captures the relationship between $\mathcal{S}$ and $\mathcal{T}$. We follow [30] and compute these quantities using a Gaussian-weighted patch F_{σ_F} of size 11×11 and $\sigma_F = 1.5$. The proposed SSIM framework [30] compares each of the properties, and is therefore composed of three components: luminance l, contrast c and structure s, which are defined as follows:

$$l = \frac{2\mu_{\mathcal{S}}\mu_{\mathcal{T}} + C_1}{\mu_{\mathcal{S}}^2 + \mu_{\mathcal{T}}^2 + C_1} \tag{3a}$$

$$c = \frac{2\sigma_{\mathcal{S}}\sigma_{\mathcal{T}} + C_2}{\sigma_{\mathcal{S}}^2 + \sigma_{\mathcal{T}}^2 + C_2} \tag{3b}$$

$$s = \frac{\sigma_{\mathcal{ST}} + C_3}{\sigma_{\mathcal{S}}\sigma_{\mathcal{T}} + C_3} \tag{3c}$$

where $\mu_{\mathcal{S}}$, $\mu_{\mathcal{T}}$ refer to the mean, $\sigma_{\mathcal{S}}$, $\sigma_{\mathcal{T}}$ refer to the variance and $\sigma_{\mathcal{ST}}$ refers to the covariance within the patch. Furthermore, to prevent instability $C_1 = (K_1 L)^2$, $C_2 = (K_2 L)^2$, $C_3 = C_2/2$, where L is the dynamic range of the feature map and $K_1 = 0.01$, $K_2 = 0.03$. An important property of SSIM is that it assigns more importance to *relative* changes in l and c due to the quadratic terms in the denominator. Furthermore, s is a direct measurement of the zero-normalized correlation coefficient between $\mathcal{S}$ and $\mathcal{T}$, and hence is formulated as the ratio between their covariance and product of standard deviations. As the range of SSIM is $[-1, 1]$, combining these three components results in the following objective:

$$\ell_{\text{SSIM}} : \mathcal{L}_{\varepsilon} = (1 - \text{SSIM})/2 = \left(1 - \left(l^{\alpha} \cdot c^{\beta} \cdot s^{\gamma}\right)\right)/2 \,, \tag{4}$$

where the prevalence of each function can be tuned, with $\alpha = \beta = \gamma = 1.0$ as a default. As our method is purely feature-based and therefore independent of the type of head or bounding box labels, we simply add $\mathcal{L}_{feat}$ to the existing detection objective function $\mathcal{L}_{det}$ (typically $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$) using weighting factor λ, which results in the following overall training objective:

$$\mathcal{L} = \lambda \mathcal{L}_{feat} + \mathcal{L}_{det} \tag{5}$$

4 Experiments

4.1 Experiment Settings

Following literature [8, 14, 26, 35, 37], we assess the performance of ℓ_{SSIM} on the MSCOCO [17] validation dataset. We report mean Average Precision (AP) as the main evaluation metric, and additionally report AP at specified IoU thresholds AP_{50}, AP_{75} and object sizes AP_S, AP_M, AP_L. Our central points of comparison are the two most widely used one- and two-stage meta-architectures, RetinaNet [19] and Faster-RCNN [23]. We respectively use ResNet/ResNeXt-101 backbones [11, 32] for the teachers and R-50 [11] backbones for all students, with $\lambda = 4, 2$. We conduct our experiments in Pytorch [22] using the MMDetection2 [4] framework on a Nvidia RTX8000 GPU with 48GB of memory. Each model is trained using SGD optimization with momentum 0.9, weight decay 1e-4 and batch size 8. The learning rate is set at 0.01 for RetinaNet [19] and 0.02 for Faster R-CNN [23], and decreased tenfold at step 8 and 11 for a total of 12 epochs. We additionally implement batch normalization layers after each convolutional layer, and use focal loss [19] with $\gamma_{fl} = 2.0$ and $\alpha_{fl} = 0.25$. The input images are resized to minimum spatial dimensions of 800 while retaining the original ratios, and we add padding to both fulfill the stride requirements and retain equal dimensionality across each batch. Finally, the images are randomly flipped with $p = 0.5$ and normalized.

4.2 Comparison with ℓ_p-norms

In this first set of experiments we compare the performance of ℓ_p-norms to ℓ_{SSIM}. Table 1 shows the results of the main experiments comparing the best performance of each $\mathcal{L}_\varepsilon$. It can be observed that: (i) ℓ_{SSIM} outperforms ℓ_p-norms by a significant margin, boosting performance with up to +3.7AP. (ii) Adopting any form of feature-based distillation results in an improvement over the vanilla network, except in Faster R-CNN [23]. (iii) Even though previous work uses ℓ_2, ℓ_1 outperforms ℓ_2 with AP improvements of 2.3 vs. 0.4 and 1.2 vs. 0.0 respectively.

Table 1: Comparison of objective functions on MSCOCO [17].

Backbone	$\mathcal{L}_\varepsilon$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet [19]							
Teacher R101		41.0	60.3	44.0	24.1	45.3	53.8
Vanilla R50	-	*36.4*	*55.6*	*38.7*	*21.1*	*40.3*	*46.6*
R50	ℓ_2	36.8 (+0.4)	55.7	39.1	20.6	40.5	47.3
R50	ℓ_1	38.7 (+2.3)	57.6	41.6	22.7	42.7	50.5
R50	$\boldsymbol{\ell}_{\text{SSIM}}$	**40.1 (+3.7)**	**59.2**	**43.1**	**23.1**	**44.6**	**53.2**
Faster R-CNN [23]							
Teacher X-101		45.6	64.1	49.7	26.2	49.6	60.0
Vanilla R50		*37.4*	*58.1*	*40.4*	*21.2*	*41.0*	*48.1*
R50	ℓ_2	37.4 (+0.0)	57.6	40.9	21.2	41.3	48.1
R50	ℓ_1	38.6 (+1.2)	58.8	42.1	21.8	42.1	49.9
R-50	$\boldsymbol{\ell}_{\text{SSIM}}$	**40.9 (+3.5)**	**61.0**	**44.9**	**23.7**	**44.5**	**53.5**

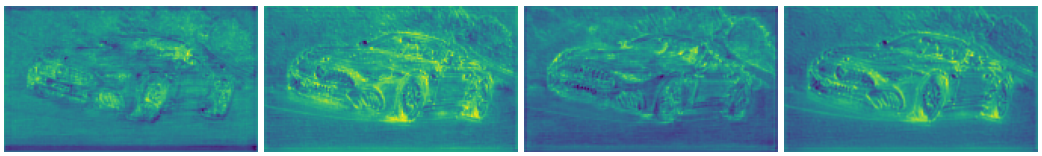
To further expand on what this improvement in performance can be attributed to, we analyze the distribution of the training stimulus in the feature space. Figure 2 illustrates the comparison between ℓ_{ssim} and ℓ_2 for the magnitude of the loss, averaged over all channels and 12 training epochs in neck $r = 1$. This tells us something about which regions in the feature space are focused on more. It can be observed that with ℓ_2, high loss is assigned to object regions in particular, and furthermore regions with high brightness, such as the sky in fig. 2a or the window in fig. 2b. ℓ_{ssim} however assigns the loss differently, where not only object regions are focused, but additionally more diverse background regions are targeted, while little importance is given to low-contrast background regions.

One of the issues highlighted by [8] is that losses are higher in object regions than in background regions. As can be seen in fig. 2, the loss applied by ℓ_{ssim} is much more distributed over the feature space than ℓ_2, which as a direct result causes a more distributed application of the gradient in the

(a) (b) (c) (d)

Figure 2: Distribution of the magnitude of the loss in the feature space $r = 1$ for various images, averaged over all channels and 12 epochs of training on MSCOCO [17]. From left to right: Image, student trained with ℓ_2, student trained with ℓ_{ssim}. A darker color indicates a higher loss, object regions have been highlighted with bounding boxes, and feature maps have been normalized.

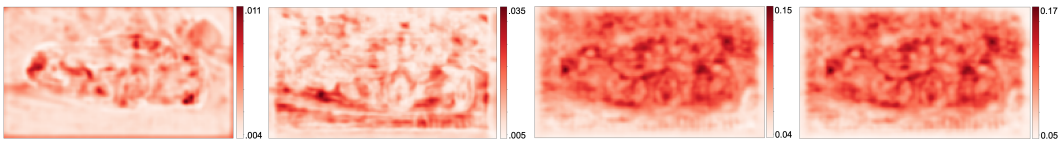
feature space. As a result, it can be observed that the feature map of a ℓ_{ssim} distilled model is much more similar to the teacher than an ℓ_p distilled model, as shown in fig. 3, which directly translates to the increase in performance.



(a) Vanilla (b) Teacher (c) ℓ_2 (d) ℓ_{ssim}

Figure 3: Qualitative comparison of channel sampled randomly from RetinaNet [19] intermediate neck features $r = 1$. Lighter colors indicate higher activations.

4.3 Influence of Luminance, Contrast and Structure



(a) Luminance (α) (b) Contrast (β) (c) Structure (γ) (d) ssim (α, β, γ)

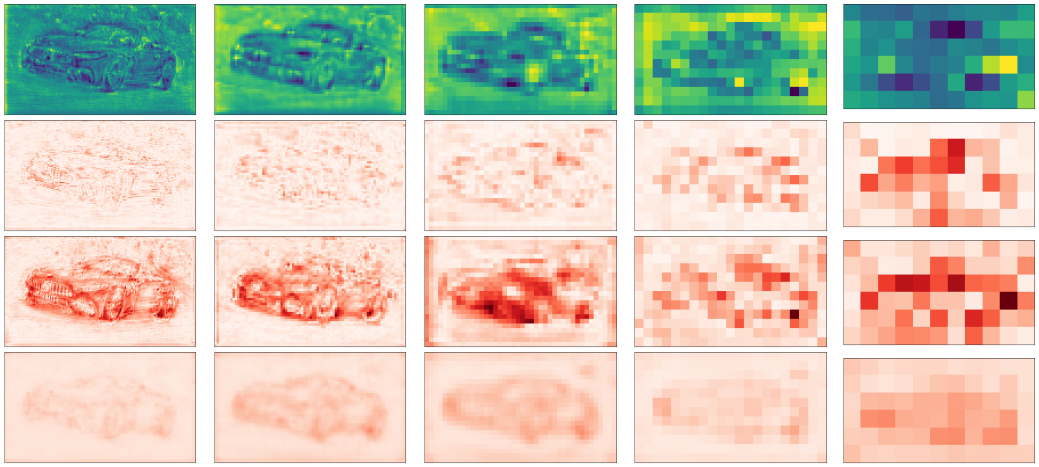
Figure 4: Distribution of the channel averaged magnitude of the loss for each individual component luminance, contrast and stucture, averaged over 12 epochs of training on MSCOCO [17].

Next we compare the influence of the luminance, contrast and structure components by tuning α, β and γ respectively (refer to eq.(4)), for this experiment we use the RetinaNet [19] teacher-student pair. The results are shown in table 2. The main observation that can be made here is that the influence of structure (γ) is more substantial than the other components, and even on its own provides an increase of +3.2 AP. Furthermore, the comparisons of luminance α and contrast β alone result in performance comparable or better than ℓ_p-norms (compare to table 1). In fig. 4 we furthermore compare the average magnitude of the loss during training. The luminance provides a similar training stimulus to the ℓ_p-norm (ref. to fig. 2c), but is more "smooth" due to the Gaussian kernel. Contrarily, contrast mostly targets background areas, as it is more sensitive to lower feature intensities. Finally, it can be noticed that structure has the most influence over the total loss, both in magnitude and spatial distribution.

Table 2: Comparison of objective functions for RetinaNet [19] on MSCOCO [17]. α tunes luminance, β tunes contrast and γ tunes structure.

Backbone	α	β	γ	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Teacher ResNet-101				41.0	60.3	44.0	24.1	45.3	53.8
Vanilla ResNet-50				*36.4*	*55.6*	*38.7*	*21.1*	*40.3*	*46.6*
ResNet-50	1	0	0	38.7 (+2.3)	57.9	41.6	21.8	42.8	50.8
ResNet-50	0	1	0	38.9 (+2.5)	57.7	41.6	21.7	42.6	51.3
ResNet-50	0	0	1	39.6 (+3.2)	58.6	42.7	22.5	44.0	52.5
ResNet-50	0	1	1	40.0 (+3.6)	59.0	42.8	22.4	44.4	**53.3**
ResNet-50	1	1	1	**40.1 (+3.7)**	**59.2**	**43.1**	**23.1**	**44.6**	53.2

Additionally, in fig. 5 we illustrate the differences between student and teacher activations for the differently trained models. It can be observed that the structure objective results in a feature space that is converged to a very similar local optimum as the teacher, with few noisy or large differences. Luminance contains more noisy differences, especially in the final layer. Although the contrast objective performs relatively well on its own, it appears to provide different activations as the teacher, particularly in the object area, as it converged to a different local minimum.



(a) $r = 1$ (b) $r = 2$ (c) $r = 3$ (d) $r = 4$ (e) $r = 5$

Figure 5: Top row: Channel averaged activations in the RetinaNet R101 [19] Teacher. Subsequent rows illustrate channel averaged differences in activations between teacher and students, distilled with: 2nd row: luminance (α). 3rd row: contrast (β). 4th row: structure (γ). Differences have been normalized, where darker color indicates a higher value.

4.4 Comparison to State-of-the-Art

Next, we compare to recent work, for which the following methods serve as baselines: (i) Zhang and Ma [35] (2021), a purely feature-based approach leveraging attention masks [38] and non-local modules [27], and (ii) Kang et al. [14] (2021), who encode labeled instance annotations in an attention mechanism [24] and report state-of-the-art for distillation methods for RetinaNet [19] and Faster R-CNN [23] on MSCOCO [17] at the time of writing. To further demonstrate the simplicity and versatility of our method, we use the original code and teachers as the authors to compare to our proposed method in the same experimental setup. [35] use the same MMDetection2 [4] framework, while [14] use Detectron2 [31]. For the comparison with [14] we furthermore adopt inheritance, a practice proposed by the authors in which the FPN [18] and head of the student are initialized with teacher parameters. This leads to faster training convergence, but may not be applicable when architectures differ between teacher and student. As the pre-trained teacher weights and exact configurations slightly vary between the two methods, we split up the comparison into two parts, as shown in table 3.

Table 3: Comparison to state-of-the-art methods on MSCOCO [17]. † denotes inheritance.

Method	RetinaNet [19]				Faster R-CNN [23]			
	AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L
Teacher	41.0	24.1	45.3	53.8	45.6	26.2	49.6	60.0
Vanilla	*36.4*	*21.1*	*40.3*	*46.6*	*37.4*	*21.2*	*41.0*	*48.1*
Zhang and Ma [35]	38.5 (+2.1)	21.7	42.6	51.5	38.9 (+1.5)	21.9	42.1	51.5
Ours	**40.1 (+3.7)**	**23.1**	**44.6**	**53.2**	**40.9 (+3.5)**	**23.7**	**44.5**	**53.5**
Teacher	40.4	24.0	44.3	52.2	42.0	25.2	45.6	54.6
Vanilla	*37.4*	*23.1*	*41.6*	*48.3*	*37.9*	*22.4*	*41.1*	*49.1*
Kang et al. [14] †	**40.7 (+3.3)**	**24.6**	44.9	52.4	40.9 (+3.0)	**24.5**	44.2	53.3
Ours †	**40.7 (+3.3)**	24.0	**45.0**	**53.1**	**41.0 (+3.1)**	23.8	**44.5**	**53.7**

It can be observed that: (i) the adoption of SSIM as the distillation function results in an improvement of +3.7 AP, and outperforms [35] for all box sizes and IoU thresholds. (ii) Both SSIM and [14] result in an improvement of +3.3 AP over the vanilla network. In particular, our method scores high for AP_L, while [14] mainly show better performance in the small object AP_S category. Additionally it can be observed that the student is able to outperform the teacher with RetinaNet [19].

4.5 Qualitative Results

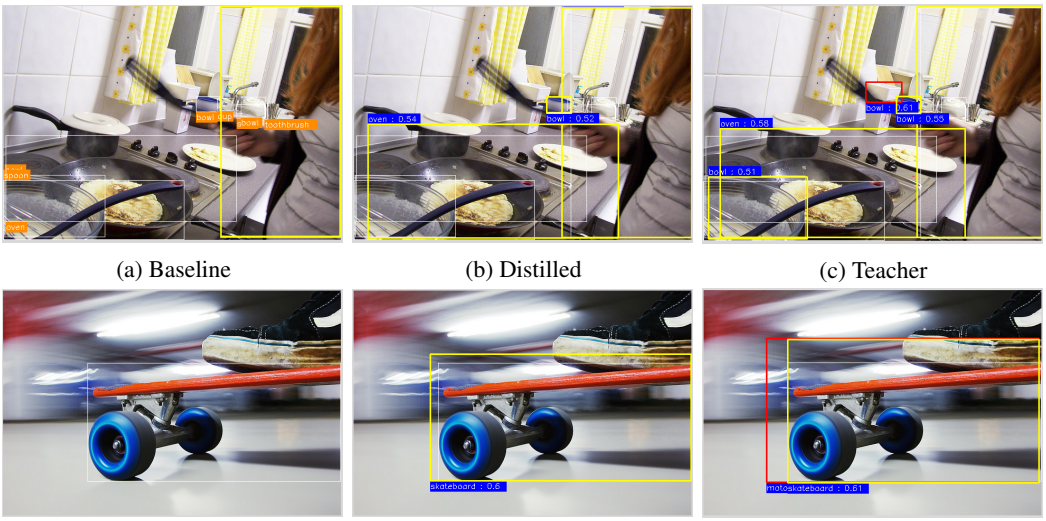
In order to verify the effectiveness of our method we analyze qualitative results in the form of several examples of detections, where we compare three models: (i) a vanilla RetinaNet-50 [19] trained without distillation which we refer to as *baseline*, (ii) a RetinaNet-50 [19]trained with our SSIM distillation method, which we refer to as *distilled*, and (iii) additionally we include the results produced by a teacher RetinaNet-101 [19], which we simply refer to as *teacher*. Throughout this section, yellow boxes denote correct predictions, red boxes denote incorrect predictions or localizations with false class predictions, and white boxes are ground truth bounding boxes.

First of all in fig. 6 we provide an example of a straightforward detection scenario, in order to obtain an indication of the overall performance. As can be expected, both the classification and localization across the board are very good. However, in this case the confidence with which our method predicts the classes is substantially higher than the baseline.

(a) Baseline (b) Distilled (c) Teacher

Figure 6: Straightforward detection scenario

Next, we provide some examples in order to verify the quantitative results which indicate that our method particularly excels in the AP_L category, which is a reflection of the performance on large objects. Figure 7a - 7c presents an example of a relatively complex scene containing multiple large objects. It can be observed that our method is able to detect additional large objects not detected by the baseline. The detections are still not as plentiful as the teacher, but the model does also not make a false positive detection. This phenomenon can also be observed in fig. 7d - 7f, where a detection is made on a close-up of a single object. The distilled model is able to detect the object, and furthermore does not make the false positive prediction made by the teacher.

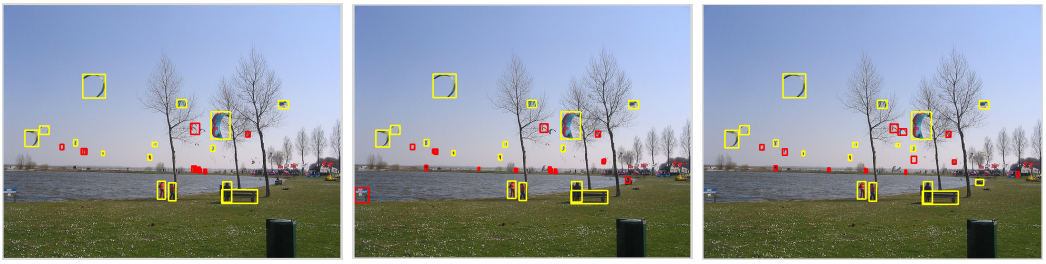


(a) Baseline (b) Distilled (c) Teacher

(d) Baseline (e) Distilled (f) Teacher

Figure 7: Detection scenarios with large objects

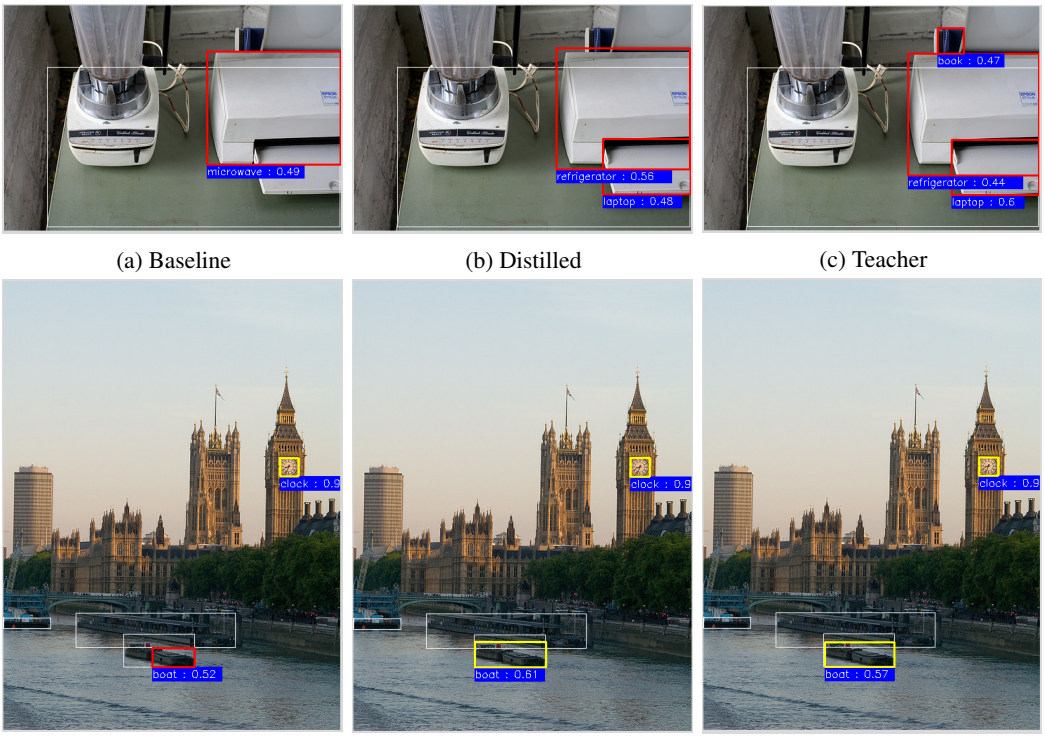
Next, we look at an example in which our method improves performance on detection of small objects. Although not as substantial as in large objects, the AP improvement over the baseline is still 2-3 AP across various evaluation settings, refer to fig. 11. Figure 8 illustrates an example of the distilled model's ability to detect objects that are tiny because of their large distance. Note that the ground truth annotations are not always perfectly accurate, in this case some clearly correct detections of persons in a distance are reported as incorrect.



(a) Baseline (b) Distilled (c) Teacher

Figure 8: Detection scenario containing many small objects

Finally, we analyze examples in which the qualitative results indicate that knowledge transferred from the teacher had impact. Figure 9a - 9c illustrate an example of incorrect predictions by each model. In contrast to the baseline, the distilled model mimics the teacher in making the same (incorrect) class prediction and an additional incorrect localization prediction. Furthermore in fig. 9d - 9f the distilled model produces improved localization compared to the baseline, where it can be observed that the teacher is mimicked.



(a) Baseline (b) Distilled (c) Teacher

(d) Baseline (e) Distilled (f) Teacher

Figure 9: Detection scenarios with information transfer. **a-c** example of classification transfer. **e-f** example of localization transfer.

Overall we have demonstrated several examples where the quantitative results are supported by the qualitative results. Particularly the detection of large objects is significantly improved, and in cases even surpasses the performance of the teacher. Additionally, both in confusing detection cases and more straightforward cases the knowledge transfer from teacher to student is manifested, both positively and negatively.

4.6 Error Type Analysis

Next, in order to gain insight in the overall strengths and weaknesses of our distillation method, we conduct an investigation of the types of errors made on the MSCOCO [17] validation dataset. We compare a vanilla RetinaNet-50 [19] trained without distillation which we refer to as the *baseline* in this analysis to our SSIM distillation method.

Figure 10 shows a curve averaged over all class categories for different types of errors for the baseline and for our method. Each plot consists of a series of precision-recall curves with each curve denoting a slightly more permissive evaluation setting. Overall AP_{75} is .431, 11.4% better than the baseline, and for a more permissive AP_{50} we obtain .591, a 6.3% improvement. If we furthermore assume perfect localization, the AP increases from .633 to .665, a 5.1% improvement. It can be observed that as we increase the permissiveness of the localization of our detector, the performance improvement is relatively less. Therefore we can conclude that our method is mainly effective in improving detection scenarios that require more precise localization.

If we furthermore move on to loosening the classification requirements, we can see that when equalizing similar categories the AP reaches 0.697, 3.9% better than the baseline. Removing all class confusions pushes AP to 0.776, 2.6% better than vanilla detection, and removing background FPs results in .878 AP, 1.3% better. Overall it can be observed that the types of errors made are quite diverse, but lean slightly to class confusions of other classes and background confusions.

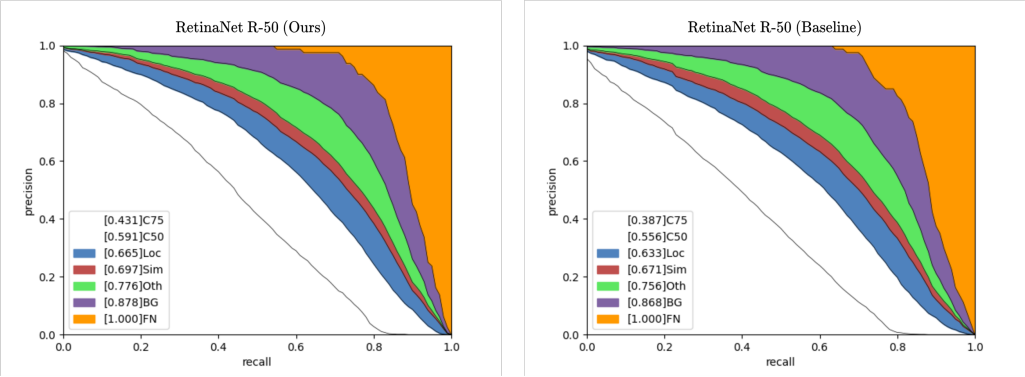


Figure 10: Distribution of error types on MSCOCO [17]. Area under Curve is provided in brackets in the legend: **C75** - at box IoU .75 (AP_{75}); **C50** - at at box IoU .50 (AP_{50}); **Loc** - at IoU .10 (localization ignored, no duplicates); **Sim** - after removal of supercategory false positives (FP's); **Oth** - after removal of all class confusions; **BG** - after removal of all background FP's; FN - False Negative predictions ($AP = 1.00$).

Furthermore, in fig. 11 we illustrate the types of error sorted by box size, where the comparison is split up in evaluation settings with increasing permissiveness. It can be noticed that the most substantial improvement in distillation performance is achieved in the large detection category. Furthermore, the most substantial improvements particularly in the medium and large category are achieved in the stricter evaluation settings (C75, C50), and decrease as permissiveness is increased.

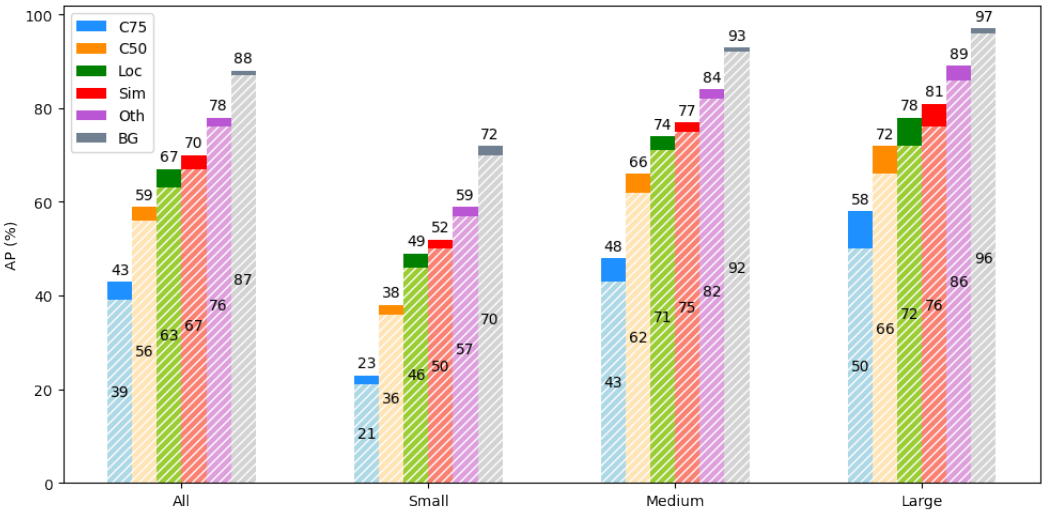


Figure 11: AP score for varying box sizes. Hatched areas represent the vanilla RetinaNet R-50 [19], solid areas represent the performance increase obtained through ℓ_{ssim} distillation.

Overall, it can be observed that the types of error that our distilled detector makes are relatively similar to that of a vanilla model. The similar pattern is that the types of errors made are fairly well distributed, with slightly more class and background confusions. Our method is furthermore particularly effective in improving performance in more strict localization metrics, and in the detection of large objects.

4.7 Ablation Studies

Generalizability to Detection Architectures and Schedules We perform additional studies on several different detection architectures to demonstrate the generalizability of our method. We evaluate our distillation method on the smaller ResNet-18 backbone, a ResNet-50 backbone with a 2x training schedule and two alternative one-stage architectures, Fsaf-RetinaNet [39], which extends RetinaNet [19] with an anchor-free module, and Reppoints [34], which replaces the regular bounding box representation of objects by a set of sample points. The results of our experiments are shown in Table 4. It can be observed that: (i) For each detection architecture our method significantly improves performance, with +3.5 AP for the ResNet18 backbone, +3.2 AP for the longer training schedule, +2.3 AP for Fsaf-RetinaNet [39], +3.3 AP for Reppoints [34]. (ii) In general, our method is modular and can significantly improve performance regardless of the detection backbone or model used.

Table 4: Investigation of several popular detection architectures on MSCOCO [17].

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet-R101 [19] (Teacher)	41.0	60.3	44.0	24.1	45.3	53.8
RetinaNet-R18 [19] (Vanilla)	32.6	50.6	34.6	17.8	35.2	43.5
RetinaNet-R18 [19] (Ours)	**36.1 (+3.5)**	**54.3**	**38.6**	**18.9**	**39.7**	**49.2**
RetinaNet-R101 [19] (Teacher)	41.0	60.3	44.0	24.1	45.3	53.8
RetinaNet-R50 [19] (Vanilla, 2x)	37.4	56.7	39.6	20.0	40.7	49.7
RetinaNet-R50 [19] (Ours, 2x)	**40.6 (+3.2)**	**59.7**	**43.7**	**23.6**	**44.8**	**53.9**
Fsaf-RetinaNeXt-X101 [39] (Teacher)	42.4	62.5	45.5	24.6	46.1	55.5
Fsaf-RetinaNet-R50 [39] (Vanilla)	37.4	56.8	39.8	20.4	41.1	48.8
Fsaf-RetinaNet-R50 [39] (Ours)	**39.7 (+2.3)**	**59.3**	**42.4**	**22.0**	**43.3**	**52.0**
Reppoints X-101[34] (Teacher)	44.2	65.5	47.8	26.2	48.4	58.5
Reppoints-R50 [34] (Vanilla)	37.0	56.7	39.7	20.4	41.0	49.0
Reppoints-R50 [34] (Ours)	**40.3 (+3.3)**	**60.3**	**43.5**	**22.6**	**44.4**	**53.9**

Effects of an Adaptation Layer Adaptation layers can be implemented when channel or spatial dimensions between teacher and student do not match, and have shown to generally improve performance previous methods [3, 26, 35]. We adopt the commonly used 1×1 convolution to investigate the influence on performance with our method. We adopt the RetinaNet R101-50 [19] teacher-student pair and the Cascade R-CNN X101 [2] - Faster R-CNN R50 [23] teacher-student pair. The results are shown in table 5. We notice that in table 5a there is no additional benefit of adopting an adaptation layer, while in table 5b the difference is significant, and implementing the adaptation layer is critical. Our method can therefore be used both with and without adaptation. However, when architectures and backbones differ between teacher and student, the adaptation layer should be implemented.

Table 5: Investigation of the effect of adaptation layers

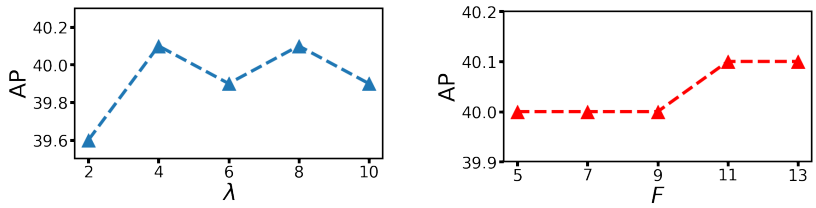
(a) RetinaNet R101 - R50 [19]

Adap. layer	AP	AP_S	AP_M	AP_L
none	**40.1**	**23.1**	**44.6**	53.2
1×1	**40.1**	**23.1**	44.4	**53.4**

(b) Cascade R-CNN X101 [2] - FRCNN R50 [23]

Adap. layer	AP	AP_S	AP_M	AP_L
none	39.8	22.6	43.4	52.1
1×1	**40.9**	**23.7**	**44.5**	**53.5**

Varying patch size F and loss prevalence λ We additionally investigate the two main hyperparameters introduced in this work, for which we use the RetinaNet R101-50 [19] teacher-student pair. The influence of the prevalence of $\mathcal{L}_{feat}$ tuned by λ is shown in fig. 12a, where it can be noticed that the choice of λ can cause a difference of up to +0.5 AP, with $\lambda = 2, 4$ providing the best performance. Additionally the influence of the local patch size F over which we calculate each component of ℓ_{ssim} is investigated, the results are shown in fig. 12b. It can be noticed that the choice of kernel size does not have significant influence on overall AP score. In table 6 it can however be observed that a smaller kernel size of $F = 5$ substantially outperforms other sizes in eh AP_S category, while a kernel size of $F = 9$ seems to be the optimal size for the best AP_L performance.



(a) Feature KD loss weight λ.

(b) Window size F.

Figure 12: Influence of Varying Hyperparameters λ and F.

Table 6: Comparison of varying window sizes F for RetinaNet [19] on MSCOCO [17].

Backbone	F	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Teacher ResNet-101		41.0	60.3	44.0	24.1	45.3	53.8
Vanilla ResNet-50		*36.4*	*55.6*	*38.7*	*21.1*	*40.3*	*46.6*
ResNet-50	5	40.0 (+3.6)	59.0	42.9	**23.8**	**44.5**	53.1
ResNet-50	7	40.0 (+3.6)	59.1	**43.1**	22.8	44.3	53.8
ResNet-50	9	40.0 (+3.6)	**59.2**	42.8	22.5	44.4	**54.0**
ResNet-50	11	**40.1 (+3.7)**	**59.2**	**43.1**	23.1	44.4	53.2
ResNet-50	13	**40.1 (+3.7)**	58.9	**43.1**	22.5	44.4	53.5

Comparison of Multiscale SSIM Loss Functions Multiscale SSIM (MS-SSIM) [29] is an adaptation of SSIM in which SSIM is calculated using varying window sizes F on the same feture map. MS-SSIM and combinations of MS-SSIM with ℓ_1 have proven successful in deep learning applications [36] in the image quality assessment domain. In this set of experiments we evaluate the performance of these various loss functions. Following [36], we test out smooth-ℓ_1, $\ell_{\text{MS-SSIM}}$, and a combination of $0.15 \cdot \ell_1 + 0.85 \cdot \ell_{\text{MS-SSIM}}$. The results are presented in table 7. It can be observed that: (i) smooth ℓ_1 boosts performance by 1.8 on its own. (ii) Adopting SSIM and the variations MS-SSIM and $\ell_{\ell_1+MS-SSIM}$ result in AP improvements of 3.5, 3.6, very similar to ℓ_{SSIM}. This demonstrates that adopting any form of SSIM is more beneficial than the pointwise ℓ_p norms.

Backbone	$\mathcal{L}_\varepsilon$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Teacher ResNet-101		41.0	60.3	44.0	24.1	45.3	53.8
Vanilla ResNet-50	-	*36.4*	*55.6*	*38.7*	*21.1*	*40.3*	*46.6*
ResNet-50	$\ell_{1,\text{smooth}}$	38.2 (+1.8)	57.2	40.7	21.5	41.9	49.9
ResNet-50	$\ell_{\text{MS-SSIM}}$	39.9 (+3.5)	59.1	42.7	22.6	44.0	**53.7**
ResNet-50	$\ell_1 + \ell_{\text{MS-SSIM}}$	40.0 (+3.6)	**59.2**	**43.2**	22.6	44.0	53.0
ResNet-50	ℓ_{SSIM}	**40.1 (+3.7)**	**59.2**	43.1	**23.1**	**44.6**	53.2

Table 7: Comparison of distillation functions using RetinaNet [19] on MSCOCO [17]

Training Resources We are additionally interested in the resources required to train a model, as this may be a critical factor in industrial applications. We compare our method to the ℓ_2 based on the required training time and memory in table 8. It can be observed that: (i) when training with distillation, the training time is significantly higher, as the forward pass through the teacher needs to be performed each iteration. (ii) The increase in memory requirement when using distillation is not as significant, as teacher weights do not need to be stored. (iii) Compared to ℓ_2 distillation, our method takes 0.14s more per iteration, and requires 1.9 GB more memory. Compared to the baseline, this difference is only 17% for both statistics respectively, and is attributed to the fact that we need to save each component of SSIM in memory during distillation.

	$t_{avg}(s) \downarrow$	Batch Size	Memory (GB) $\downarrow$
Vanilla	*0.87 ± 0.007*	*8*	*11.8*
ℓ_2	1.33 ± 0.012 (+52%)	8	12.7 (+8%)
ℓ_{SSIM}	1.47 ± 0.011 (+69%)	8	14.8 (+25%)

Table 8: RetinaNet R-50 [19] training statistics on MSCOCO [17]. t_{avg} indicates one for ward/backward training iteration.

5 Discussion

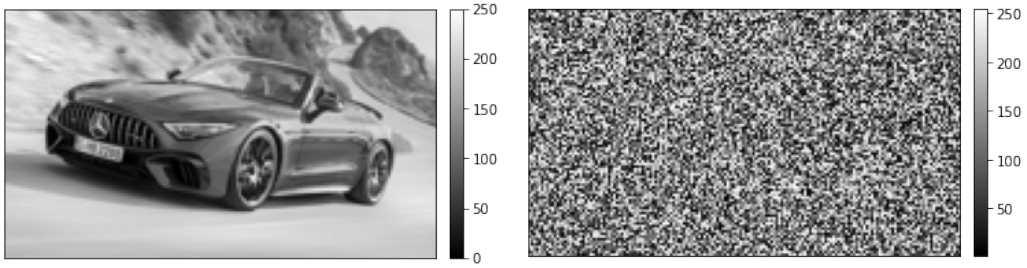
In this section we delve into a deeper analysis of the proposed method. We start off with providing interpretations for the obtained results. Next, we discuss how our findings are related to previous work. Subsequently, we discuss how the proposed method could be used in real-world applications, and provide specific examples. We additionally discuss potential future improvements, and finish off with the broader implications of this work.

5.1 Obtained Results

Among other considerations, we provide interpretations for where the majority of performance improvement comes from, what the student learns from the teacher, and how influential the choice of hyperparameters is.

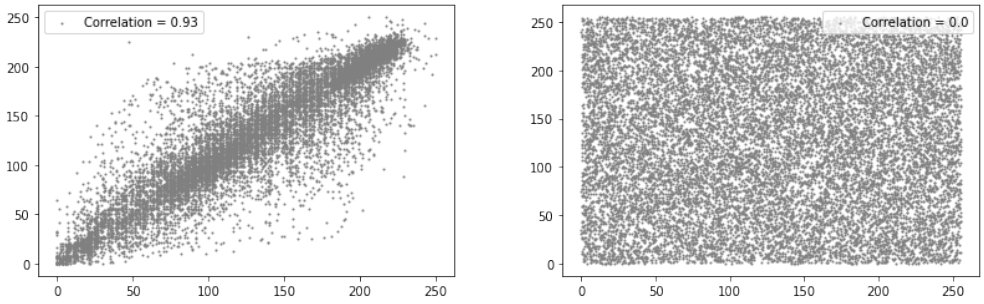
What is the main reason for performance improvement? Throughout the experiments, the overall observation we make is that ℓ_{SSIM} is able to outperform pointwise metrics in any scenario we ran experiments on. We hypothesize that one of the main reasons for the performance improvement is that ℓ_{SSIM} calculates similarity measures in a local region, are thereby is able to take into account additional knowledge contained in spatial relationships between features. This is contrary to pointwise metrics which, as can be observed in eq. (2), only compare the values of a single feature.

To substantiate this hypothesis, we demonstrate the presence of relational information in the feature space. With images, one can intuitively reason that pixels are not independent of each other; images are typically smooth, *i.e*. pixels that are spatially proximate will often have similar intensities. We can quantify this intuition by comparing an original image (fig. 13a) to an image containing random uniformly sampled pixels (fig. 13b). Subsequently, we plot the intensity of *neighbouring pixels* against each other for an entire image, as shown in fig. 13c, 13d. In fact, calculating the correlation coefficient results in ± 0.93 for the original image, and 0.0 for the random uniformly sampled image, confirming that indeed the pixels are spatially related in images.



(a) Original Image.

(b) Random Uniformly Sampled Pixels.



(c) Original Image.

(d) Random Uniformly Sampled Pixels.

Figure 13: Top: example images. Bottom: scatter-plot containing intensities of neighbouring pixel *i* on the x-axis and pixel *j* on the y-axis.

A similar pattern can be observed in feature maps, which we demonstrate by highlighting some characteristics of the feature maps generated by a Convolutional Neural Network (CNN). Specifically,

one key property of a CNN is that, due to its kernel-based nature, the connectivity of a feature is local in the spatial dimension $\mathbb{R}^{H,W}$, and global along the depth dimension $\mathbb{R}^{C}$. As a result, dependency between features is retained along the spatial dimension, but not along the depth dimension. We can again quantify this dependency retainment by calculating the correlation coefficients between neighbouring features in $\mathbb{R}^{H,W}$ throughout $\mathcal{T}$ and $\mathcal{S}$, the results of which are shown in table 9 for the original image. As can be observed, as we go deeper into the network and the receptive field increases, features become more abstract and spatial correlation decreases but remains strongly positive. We can therefore conclude that the features are not independent, and there is inherent knowledge present in the relationships between the features. Capturing these relationships in the distillation process subsequentely results in improved performance. The retainment of the spatial relationships is additionally qualitatively visible in fig. 3.

r	1	2	3	4	5
Teacher RetinaNet-R101 [19]	0.88	0.85	0.81	0.68	0.43
Ours RetinaNet-R50 [19]	0.87	0.85	0.81	0.69	0.44

Table 9: Average correlation coefficients between neighbouring features throughout the feature space, at each different feature map scale r.

What does the student learn from the teacher? It is difficult to exactly map out what the student learns from the teacher. What we can do however to gain some insight into this question is to evaluate the performance and analyze which metric has the most *relative* improvement compared to a vanilla network, and which types of errors are mainly reduced. Clearly we have seen in fig. 11 that the AP_L category has seen the most improvement. These detections are mostly influenced by the latter stages of the feature space ($r = 4, 5$), indicating that the proposed combination of kernel size and are particularly suited for these latter layers. This can also be qualitatively observed in fig. 5. Furthermore, the nature of the types of errors made by the distilled model are not significantly different than the vanilla model, as can be observed in fig. 10. We can therefore reason that distillation helps to reduce errors of all types, and is not limited to specific metrics.

One limitation of KD that has to be kept in mind that there is exists no "perfect teacher". Therefore finding the perfect distillation method as an objective function will not directly allow for perfect students. Furthermore, KD remains a mechanism for regularization, as initially shown by [12]. As such, the upside is that it allows the student to even outperform the teacher, as can be observed in table 3. This can be mainly attributed to the boost the student receives at the beginning of the training compared to a vanilla model, as illustrated in fig. 14. The training stimulus received early on from the cues of the teacher allow for faster convergence, and combined with the continual ground truth training stimulus it enables the student to fine-tune and outperform the teacher in the latter stages of the training.

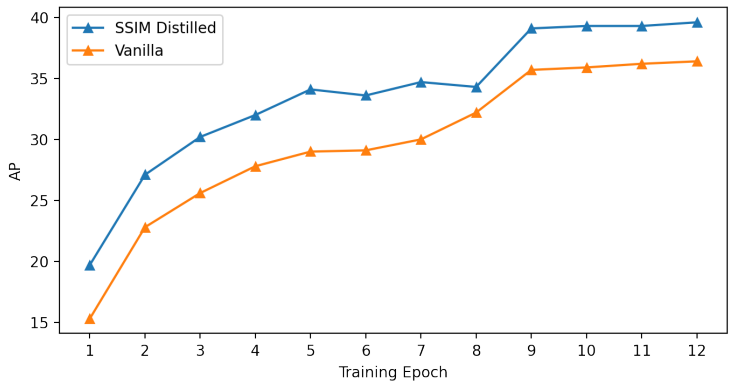


Figure 14: Progression of RetinaNet-R50 [19] AP score throughout training 12 epochs on MSCOCO [17].

How Influential are the Hyperparameters? Of interest are furthermore to what degree the hyperparameter choice in our method is of influence. Recall the hyperparameters λ, which measures the weight of the KD loss, and F, which determines the size of the window over which the SSIM is calculated. As we have observed in fig. 12a, λ has an influence of up to 0.5 AP, which is in the same order of magnitude as the influence of hyperparameters in previous methods [14, 35]. As the distillation loss drives ±10% of the overall performance, the choice of λ on distillation is quite substantial, and should be thoroughly evaluated when implemented in a custom system.

Contrary to intuition, as can be observed in fig. 12b the window size only impacts performance by 0.1 AP, which is not a significant difference. Interestingly, there are major performance differences to be found in the AP categories for the different box sizes, as illustrated in table 6. The smallest window size of 5 improves performance most on the AP_S category. We hypothesize due to the smaller window size, the gaussian weighting retains more detail in the feature space, which plays a particularly important role in the spatially larger earlier layers $r = 1, 2$ which are responsible for detecting smaller objects. Furthermore, not the largest window size 13, but 9 provides the most improvement in the AP_L category. As the latter feature maps are in the order of $H, W = 11$ (depending on the input size), this seems to be the sweet spot for the transfer of knowledge at these latter layers. Overall AP still remains most improved with the default window size of 11, which seems to strike a good balance for good performance across all metrics.

Finally, using multiple window sizes though MS-SSIM does also not provide additional benefits, as can be observed in table 7. In the image restoration domain this particular function provided improved performance through its ability to capture relationships in images at different scales [36]. As the feature maps inherently provide features at different scales, the effect of using MS-SSIM is therefore negated.

5.2 Relation to Other Work

Compatibility with sampling mechanisms In our proposed method we take into account each feature where we replace the ℓ_p-norms with ℓ_{ssim}, and as demonstrated in table 3 we even are able to outperform sophisticated state-of-the-art sampling mechanisms [14, 35]. As previous work has mainly focused on sampling object regions, we furthermore are interested if ℓ_{ssim} can be combined with existing sampling mechanisms that focus on these object regions. To that end we provide an initial implementation where we adopt the spatial attention maps proposed by [35], and replace he ℓ_s with ℓ_{ssim}, the results of which are shown in table 10.

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Spatial-SSIM	39.4 (+3.0)	58.3	42.2	21.8	43.6	52.4
SSIM	**40.1 (+3.7)**	**59.2**	**43.1**	**23.1**	**44.6**	**53.2**

Table 10: Comparison of Spatial Attention + SSIM to only SSIM, applied using RetinaNet [19] on MSCOCO [17]

These results indicate that combining existing sampling mechanisms might be counterproductive, and generally all features need to be accounted for when using ℓ_{ssim} as a distillation mechanism, as we have seen in section 4.2. It is also not possible to directly replace ℓ_2 in methods such as [14], as the features used for distillation are projected to a subspace where spatial relationships between the features are not present.

Can the transfer of relational knowledge be made learnable? An earlier attempt to capture relational knowledge in the feature-based KD domain has been made by Zhang and Ma [35] (2021), who propose non-local modules [27] as a learnable way to capture relational information between features. Non-local modules are learnable blocks that compute the response at a position as a weighted sum of the features at all positions. Compared to our method we identify two major disadvantages. First: as we can observe in table 3, performance is not as strong as with our method. Second: From an explainability standpoint it remains unclear what the relationship is between the non-local regions and KD relevant features.

5.3 Real World Applications

As our main research interest lies in improving real-time perception systems that operate in complex and changing environments, we discuss two scenarios of direct relevance to our method: training networks with unlabeled data and ensuring the method provides robust performance.

Unlabeled Data Throughout this work we compare our method to existing literature using ground truth annotations. A major hardship of annotated data is that is typically very expensive and labor intensive to obtain. Therefore, in order to simulate a scenario in which data annotations are not available, we furthermore investigate performance on MSCOCO [17] without ground truth annotations. This is achieved through *hard output distillation*, *i.e*. we use the outputs of the teacher model with confidence $p > 0.3$ as labels for the student. The results are shown in Table 11. Our ℓ_{ssim} method achieves a +1.9 AP improvement over the vanilla model, compared to +1.1 AP when using ℓ_2. This furthermore demonstrates the advantage ℓ_{ssim} distillation can bring when dealing with a scenario in which annotations are not available.

Backbone	$\mathcal{L}_\varepsilon$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Teacher ResNet-101		41.0	60.3	44.0	24.1	45.3	53.8
Vanilla ResNet-50		*34.6*	*53.8*	*36.7*	*20.3*	*38.4*	*43.7*
ResNet-50	ℓ_2	35.7 (+1.1)	54.9	38.1	20.5	39.5	44.8
ResNet-50	$\boldsymbol{\ell}_{\text{SSIM}}$	**36.5 (+1.9)**	**55.8**	**38.9**	**21.6**	**40.4**	**46.1**

Table 11: RetinaNet [19] experiments on MSCOCO [17] w/o annotations.

Robustness For use cases such as autonomous driving, it is of major importance that the detector functions regardless of image distortions or weather conditions. The Robust Detection Benchmark [21] introduces a way to evaluate detectors in which the performance of the algorithm is tested over 15 different types of distortions such as blur, noise, snow and fog conditions. Additionally, five different severity levels are introduced for each distortion, for a total of 75 different scenarios. Two metrics are introduced: mPC (mean Performance under Corruption) measures the average AP over each of the distortions, while rPC (relative Performance under Corruption) measures the performance on distortions relative to clean data. It can be observed in table 12 that our distillation method not only is more robust (+2.1 mPC), but also improves the relative robustness (+0.7 rPC). Our distillation method therefore not only demonstrates an absolute increase in performance, but also has improved generalization ability to scenarios in which the visual conditions are not as optimal as in a prepared dataset.

Backbone	$\mathcal{L}_\varepsilon$	AP	mPC	rPC
Vanilla ResNet-50		*36.5*	*18.0*	*49.4*
ResNet-50	ℓ_{ssim}	40.1 (+3.6)	20.1 (+2.1)	50.1 (+0.7)

Table 12: RetinaNet [19] robustness experiments on MSCOCO [17]

5.4 Future Improvements

Kernel Size One aspect that can be improved is the difference in performance on different object sizes. As can be noticed in *e.g*. table 3 and table 6, when using the "balanced" kernel size of 11 there is still a clear gap in performance compared to Kang et al. [14] (2021) and smaller kernel sizes in the AP_S category. One potential solution would be to investigate using the sizes of teacher prediction boxes in order to determine the kernel size which should be applied.

Extension to other Tasks This work has focused around object detection, as a major aspect is the distillation of structural components of the feature space, which directly are associated with objects. It is of interest to further investigate the effectiveness of our method on additional object-centric tasks, and furthermore on multi-task systems that combine object-centric and dense prediction.

5.5 Implications

As ℓ_{ssim} is very straightforward to implement and does not require any labeled data we believe that it can be easily integrated into industry applications, some of which have already been demonstrated in this work. Furthermore, up to this point the research space has mainly focused on creating sampling mechanisms that focus on the object regions in the feature space. In this work we provided a contrasting approach that is in line with the findings of [8] in that not just object regions are important, but also the background regions. With our method, particular importance is additionally found in background regions that contain more structure, but do not directly contain objects for detection. Through its ability to outperform methods that have refined sampling mechanisms it has shown potential for future use. We believe that ℓ_{ssim} can serve as a good basis to build future KD mechanisms on, and provides an alternative direction for the research space, other than creating ever more sophisticated sampling mechanisms.

6 Conclusion

This work proposed ℓ_{ssim}, a replacement for the conventional ℓ_p-norm as a building block for feature-based KD in object detection. By taking into account additional contrast and structural cues, feature importance, correlation and spatial dependence are considered in the loss formulation. ℓ_{ssim} outperforms ℓ_p-norm by a great margin and is able to reach performance on par or even surpass state-of-the-art without the need for carefully designed and complex sampling mechanisms. Our method is simple and can be implemented by replacing one line of code. Future work consists of using ℓ_{ssim} as a building block for future KD methods, which includes applying it to other tasks.

References

[1] C. Bucilă, R. Caruana, and A. Niculescu-Mizil. Model compression. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006:535–541, 2006. doi: 10.1145/1150402.1150464.

[2] Z. Cai and N. Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[3] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker. Learning efficient object detection models with knowledge distillation, 2017. ISSN 10495258.

[4] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. {MMDetection}: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[5] X. Dai, Z. Jiang, Z. Wu, Y. Bao, Z. Wang, S. Liu, and E. Zhou. General Instance Distillation for Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7842–7851, 2021.

[6] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas. Predicting parameters in deep learning. *Advances in Neural Information Processing Systems*, pages 1–9, 2013. ISSN 10495258.

[7] D. P. E. R. E. Riba D. Mishkin and G. Bradski. Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL `https://arxiv.org/pdf/1910.02190.pdf`.

[8] J. Guo, K. Han, Y. Wang, H. Wu, X. Chen, C. Xu, and C. Xu. Distilling object detectors via decoupled features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2154–2164, 2021.

[9] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, 2015-Janua:1135–1143, 2015. ISSN 10495258.

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–14, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. pages 1–9, 2015. URL `http://arxiv.org/abs/1503.02531`.

[13] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and others. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[14] Z. Kang, P. Zhang, X. Zhang, J. Sun, and N. Zheng. Instance-Conditional Knowledge Distillation for Object Detection. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[16] Q. Li, S. Jin, and J. Yan. Mimicking very efficient network for object detection. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua: 7341–7349, 2017. doi: 10.1109/CVPR.2017.776.

[17] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10602-1{_}48.

[18] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:936–944, 2017. doi: 10.1109/CVPR.2017.106.

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[20] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang. Structured knowledge distillation for semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:2599–2608, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00271.

[21] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2015. URL `http://arxiv.org/abs/1506.01497`.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[25] L. Wang and K. J. Yoon. Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):1–40, 2021. ISSN 19393539. doi: 10.1109/TPAMI.2021.3055564.

[26] T. Wang, L. Yuan, X. Zhang, and J. Feng. Distilling object detectors with fine-grained feature imitation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4933–4942, 2019.

[27] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[28] Z. Wang and Q. Li. Information content weighting for perceptual image quality assessment. *IEEE Transactions on image processing*, 20(5):1185–1198, 2010.

[29] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems \& Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[31] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. \url{https://github.com/facebookresearch/detectron2}, 2019.

[32] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[33] W. Xue, L. Zhang, X. Mou, and A. C. Bovik. Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. *IEEE transactions on image processing*, 23(2): 684–695, 2013.

[34] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9657–9666, 2019.

[35] L. Zhang and K. Ma. Improve Object Detection with Feature-based Knowledge Distillation: Towards Accurate and Efficient Detectors. In *International Conference on Learning Representations*, 2021.

[36] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2016. ISSN 2573-0436. doi: 10.1109/tci.2016.2644865.

[37] D. Zhixing, R. Zhang, M. Chang, S. Liu, T. Chen, Y. Chen, and others. Distilling Object Detectors with Feature Richness. *Advances in Neural Information Processing Systems*, 35, 2021.

[38] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[39] C. Zhu, Y. He, and M. Savvides. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 840–849, 2019.

Appendix

A Reproducibility Statement

All our experiments are based on publicly available frameworks [4, 31] and datasets [17]. An example implementation of KD loss between teacher and student features is shown below. Omitting the import and using a library such as Kornia [7], a change from ℓ_2 to ℓ_{SSIM} only requires a change in **one line of code**.

l2 implementation

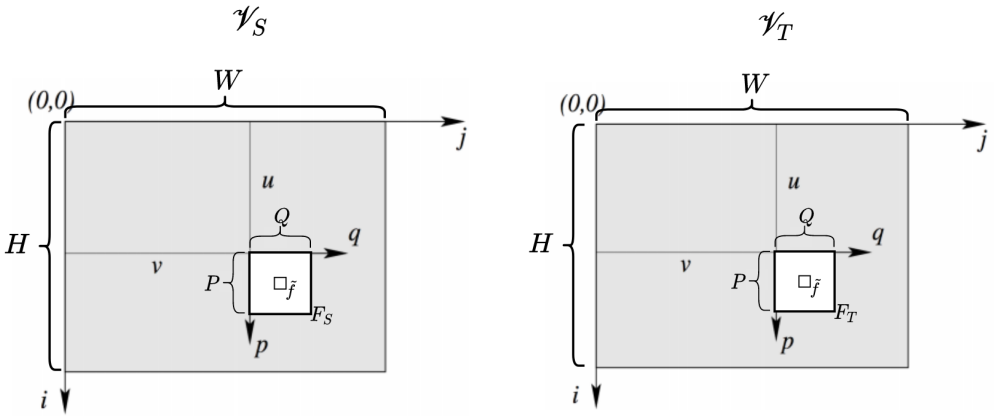
```
from torch.nn.functional import mse_loss
def kd_loss(student_feats, teacher_feats):
    # inputs have shape [B, C, H, W]
    kd_feat_loss = mse_loss(student_feats, teacher_feats)
    return kd_feat_loss
```

ssim implementation

```
from kornia.losses import ssim_loss
def kd_loss(student_feats, teacher_feats):
    # inputs have shape [B, C, H, W]
    kd_feat_loss = ssim_loss(student_feats, teacher_feats, window_size=11)
    return kd_feat_loss
```

B Derivations

In this section we demonstrate how to calculate each statistical property used in our KD method.



(a) Student (b) Teacher

Figure 15: Geometric illustration of intermediate feature maps. u and v are the location of the top left feature of patches F. The patches are subsequentely defined as follows: $F_{\mathcal{S}} = \mathcal{V}_{\mathcal{S}}([u, u+1, ..., u+P], [v, v+1, ..., v+Q])$ and $F_{\mathcal{T}} = \mathcal{V}_{\mathcal{T}}([u, u+1, ..., u+P], [v, v+1, ..., v+Q])$ with central feature $\tilde{f}$. Finally P and Q indicate the size of the patch.

Mean

$$\mu_{\mathcal{S}}(F_{\mathcal{S}}) = \frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} F_{\mathcal{S}}(p,q) \qquad (6a) \qquad \mu_{\mathcal{T}}(F_{\mathcal{T}}) = \frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} F_{\mathcal{T}}(p,q) \qquad (6b)$$

Variance

$$\sigma^2_{\mathcal{S}}(F_{\mathcal{S}}) = \frac{1}{PQ-1}\sum_{p=1}^{P}\sum_{q=1}^{Q}\left(F_{\mathcal{S}}(p,q) - \underbrace{\frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} F_{\mathcal{S}}(p,q)}_{\mu_{\mathcal{S}}(F_{\mathcal{S}})}\right)^2 \qquad (7)$$

$$\sigma^2_{\mathcal{T}}(F_{\mathcal{T}}) = \frac{1}{PQ-1}\sum_{p=1}^{P}\sum_{q=1}^{Q}\left(F_{\mathcal{T}}(p,q) - \underbrace{\frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} F_{\mathcal{T}}(p,q)}_{\mu_{\mathcal{T}}(F_{\mathcal{T}})}\right)^2 \qquad (8)$$

Covariance

$$\sigma_{\mathcal{ST}}(F_{\mathcal{S}}, F_{\mathcal{T}}) = \frac{1}{PQ-1}\sum_{p=1}^{P}\sum_{q=1}^{Q}\left[\left(F_{\mathcal{S}}(p,q) - \underbrace{\frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} F_{\mathcal{S}}(p,q)}_{\mu_{\mathcal{S}}(F_{\mathcal{S}})}\right)\cdot \left(F_{\mathcal{T}}(p,q) - \underbrace{\frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} F_{\mathcal{T}}(p,q)}_{\mu_{\mathcal{T}}(F_{\mathcal{T}})}\right)\right] \qquad (9)$$