Combining Denoising and Object Detection

An analysis to provide insights in combining denoising with object detection

MSc Thesis Computer Science
Rick Huizer

TU Delft

BOSCH

Combining Denoising and Object Detection

An analysis to provide insights in combining denoising with object detection

by

Rick Huizer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 21, 2023 at 13:00 PM.

Student number:	4847652	
Project duration:	November 14, 2022 – August 21, 2023	
Thesis committee:	Assoc. Prof. dr. J.C. van Gemert,	TU Delft
	Assoc. Prof. dr. N. Yorke-Smith,	TU Delft
	Dr. O.S. Kayhan,	TU Delft, Bosch Security Systems

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

TU Delft

Preface

This report describes the work done for my thesis for the Master of Computer Science at Delft University of Technology.

I want to give a special thanks to my supervisor Osman Kayhan. His supervision has been very helpful and has made this thesis a very educational process. He has provided me with a lot of helpful insights and tips and we could also have good laughs together. I want to thank Thomas Markhorst and Dajt Mullaj for being very nice, helpful peer students during my thesis. We could always have a laugh and ask each other any questions. Another thanks goes out to the team of Bosch Security Systems B.V. for providing me with the unique opportunity to conduct my thesis in their advanced development team. A big thanks to Jan van Gemert for his critical comments and thinking, which helped me focus my research on the right areas. Finally I want to thank my family and friends for their support.

Rick Huizer
Delft, August 2023

Contents

1 Introduction

Automated imaging systems play a crucial role in numerous domains, such as medical imaging, autonomous driving, and securing perimeters. Images from for example video cameras are processed automatically, saving time when compared to manual inspection of a human. In practice, these systems suffer from noise caused by sensor measurements and electronic circuits, especially in bad lighting conditions or dark scenes. Some tasks that are effected by this noise are image classification [1], instance segmentation [2], and object detection [3].

This thesis focuses on object detection: The problem concerned with localizing and classifying objects present in an image. Object detection has many applications, such as autonomous driving, systems related to security [4], and highway traffic monitoring. The context of this thesis is object detection in security cameras. Even though these systems are automated, the ability to verify the systems' actions is important. The system should provide clear images and accurate object detection to facilitate this. When we consider a building security scenario, a guard has to monitor a certain property using an automated system to detect intruders. High-quality images allow the guard to accurately double-check the alerts from the system. Reliable object detection ensures that the system does not miss any intruder that tries to break into the property. Excellent performance in denoising and detection is therefore essential.

The limitation of other approaches [5]–[7] that combine denoising with object detection, is that they focus only on the final detection performance. The image quality is not one of the final objectives that is optimized for. Having noisy images with detection is insufficient, as verification of the detections and manual inspection of the images is difficult when noise is present.

Motivated by the importance of both image quality and detection, and the limitations of related literature, this thesis provides an analysis that is focused on optimizing for both image quality and detection performance, in the context of security cameras. This thesis compares a number of strategies to combine denoising and object detection, providing an in-depth understanding of each strategy's performance, and what strategy performs best.

This report is structured as follows: Chapter 2 is a scientific article of the thesis, presenting and discussing all the experiments and results. Chapter 3 contains background information on technical aspects discussed in the scientific article, which assists the reader in understanding the research.

2

Scientific article

DODa: Denoising-Object Detection Analysis

Rick Huizer[1,2] Osman Semih Kayhan[1,2] Jan C.van Gemert[2]
[1] Bosch Security Systems [2] Delft University of Technology

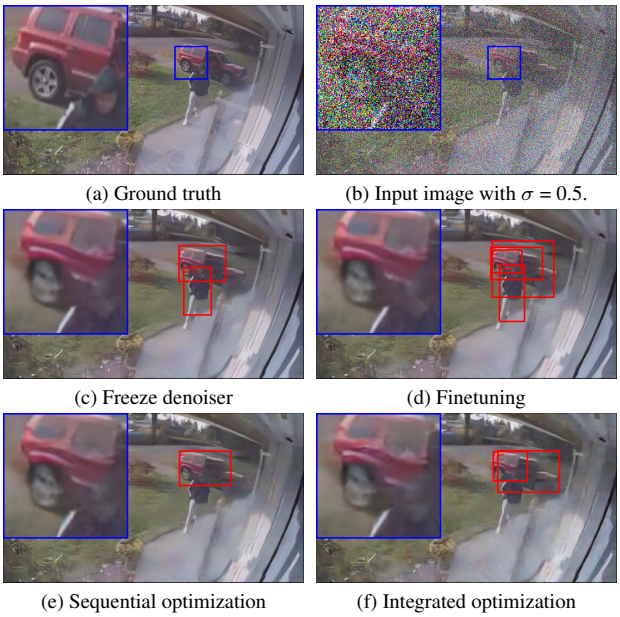
Abstract

Automated imaging systems, critical in domains like medical imaging, autonomous driving, and security, experience noise from camera sensors and electronic circuits in bad or dark lighting conditions. This impacts downstream tasks, including object detection. However, an analysis of strategies combining denoising and object detection is lacking. This study addresses this gap by analyzing diverse strategies for optimizing both image quality and detection performance. Results reveal that isolating denoiser network optimization and training a detector on its outputs yields the best overall performance. Combining detection and denoising enhances detection outcomes. The results offer valuable insights to make educated decisions on how to combine denoising and detection in modern imaging systems.

1. Introduction

Automated imaging systems play a crucial role in numerous domains, such as medical imaging, autonomous driving, and securing perimeters. By using automated image analysis, these systems provide valuable and important assistance to humans. In practice, these systems suffer from noise caused by sensor measurements and electronic circuits, especially in bad lighting conditions. Video cameras are a common imaging system that enables automatic image analysis like image classification [26], instance segmentation [4], and object detection [24], yet noise affects these downstream tasks severely [10].

Object detection systems are used in autonomous driving, security scenarios [17], and highway traffic monitoring. Even though these systems are automated, the ability to verify the systems' actions is important. The system should provide clear images and accurate object detection to facilitate this. When we consider a building security scenario, a guard has to monitor a certain property using an automated system to detect intruders. High-quality images allow the guard to accurately double-check the alerts from the system. Reliable object detection ensures that the system does not miss any intruder that tries to break into the property. Excellent performance in denoising and detection is therefore essential.



(a) Ground truth (b) Input image with $\sigma = 0.5$.
(c) Freeze denoiser (d) Finetuning
(e) Sequential optimization (f) Integrated optimization

Figure 1. Denoised images with detections by different combination strategies for denoising and detection (Zoom in to see details). (c) has better detection and fewer denoising artifacts compared to other strategies, and is therefore preferred over other optimization strategies.

The limitation of other approaches [16, 25, 32] that combine denoising with object detection, is that they focus only on the final detection performance. The image quality is not one of the final objectives that is optimized for. Having noisy images with detection is insufficient, as verification of the detections and manual inspection of the images is difficult when noise is present.

Motivated by the importance of both image quality and detection, and the limitations of related literature, we provide an analysis in this paper that is focused on optimizing for both image quality and detection performance, in the context of security cameras. We compare naïve, existing, and novel strategies to combine denoising and object detection, providing an in-depth

understanding of each strategy's performance. The main contributions are:

1. We provide multiple in-depth analyses of various strategies in combining denoising and object detection
2. We propose and analyze a novel optimization strategy to combine denoising and object detection

2. Related work

Denoising. The work of Burger et al. [1] shows that a simple Multi-Layer Perceptron (MLP) network is able to outperform the popular hand-crafted BM3D [6] algorithm. Following this, more complex CNN-based denoising networks were introduced like DnCNN [41] and MVCNN [21]. Recently, transformer-based denoisers like SwinIR [18] and Uformer [37] were presented. Interestingly, many of these proposed denoising networks utilize an encoder-decoder structure, such as RED-Net [23], MVCNN and UNet [31]. UNet in particular is an architecture that inspired many follow-up works with competitive performance, for example SUNet [12], RDUNet [40], and Uformer. This popularity of UNet is supported by the work by Ulyanov et al. [34], which shows that the UNet architecture is suited to suppress noise and reconstruct natural images. This analysis uses UNet for denoising for these reasons.

Object detection. RCNN [14] was one of the first object detector networks that outperformed algorithms using handcrafted features like HOG [8] or the Viola-Jones [35] detector. Followup works of this two-stage detector such as FastRCNN [13] and FasterRCNN [29] improved results significantly. FasterRCNN is used as inspiration for many subsequent research, including R-FCN [7], Mask RCNN [15], PANet [22] and DetectoRS [27]. Next to the two-stage family, single-stage detectors including YOLO [28], RetinaNet [20] and CenterNet [11] were proposed. Single-stage models are faster, but at the cost of performance [39]. State-of-the-art detection performance is achieved by transformer-based networks like Group-DETR v2 [5] and Co-DETR [42]. These networks are based on DETR [2]. The authors show that FasterRCNN combined with Feature Pyramid Networks [19] and GIoU [30] loss can compete with DETR, even outperforming on detecting small objects. A major downside of transformer-based networks is the complex and long training process. Because of its competitive performance, wide adaptation, and simple and relatively quick training process, we use FasterRCNN in this analysis.

Combining denoising and object detection. Some existing approaches [24, 33] fuse denoising and object detection using algorithms only optimized for one task. They are combined in a sequential configuration: First, the denoising algorithm is applied, after which detection takes place. Other works [25, 32, 38] concatenate two networks into one network that is optimized end-to-end. This network consists of part (i): responsible for denoising, and part (ii): performing detection. The exact setup varies between publications: sometimes parts of the network are frozen and pretrained [25], or authors target detection performance [16]. The work of Hong et al. [16] blends the networks of denoising and object detection into one, resulting in a shared feature representation. They have a shared encoder for both denoising and detection, followed by distinct output heads for each task. Their work focuses on reaching high performance in detection, not caring about denoising results. An important common factor is that these methods do not focus on achieving good performance in both tasks, which is something we do focus on in this analysis.

3. Optimization of Denoising and Object Detection

In this section, we explain various strategies to combine denoising and object detection, which are used in the analysis in the paper. Figure 2 visualizes and explains the strategies used.

Isolated optimization. The first strategy to combine the two tasks is to optimize each task in isolation and concatenate the detector network after the denoiser network. The denoiser is trained with denoising loss, and the detector is trained with detector loss. The denoiser first removes noise from input images. The cleaned image is then fed into the detector.

Freeze denoiser. One can also first optimize the denoiser in isolation with denoising loss, and then freeze the weights. Afterward, the detector can be trained on the output of the denoiser with detection loss. This approach allows the detector to adapt to the outputs of the denoiser.

Finetune both networks. Instead of training the detector from scratch on the output of the denoiser, one can also first train both networks in isolation with separate denoising and detection loss. Next, finetune the outputs and inputs to each other by concatenating the detector after the denoiser network. The network is then trained end-to-end with a weighted combination of denoising and detection loss resulting in the following loss function:

$$\mathcal{L} = \lambda_{\text{Den}} * \mathcal{L}_{\text{Denoising}} + \lambda_{\text{Det}} * \mathcal{L}_{\text{Detection}} \tag{1}$$

To finetune, the learning rate is set to a smaller value than is used during training from scratch. The detector can adapt to the output from the denoiser, and the denoiser can adjust its outputs to help the detector's performance.

Sequential optimization. Optimizing both networks with random initialization in a sequential setup is another strategy that can be used to combine the tasks. The detector

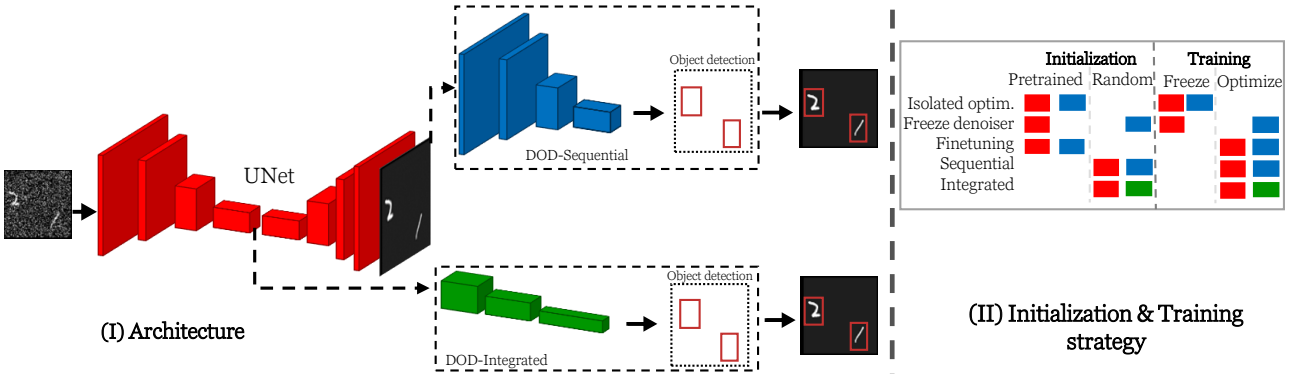


Figure 2. (i): Architectures used to denoise and detect objects. The red network is a UNet architecture for denoising. The blue network is an object detector network, using the denoised output images of UNet. The green network is a different object detector, using shared features of the red network as input. The denoised image and detections are combined together to achieve the final image. (ii): Explanations for each combination strategy, what parts are trained, and how each of them is initialized. *Pretrained* describes that the strategy is using pretrained weights, from a task optimized in isolation. *Random* means that random network initialization is used. *Freeze* indicates what parts of the network are frozen during training. Which parts of the network are optimized is shown by the *Optimize* column. If no blocks are optimized, the network can immediately be evaluated on a test set.

is concatenated after the denoiser. This strategy combines both loss functions, shown in Equation (1).

Integrated optimization. The two networks can also be combined into one single network. This network has a shared encoder, and two distinct output heads for (i): denoising and (ii): detection. The network is randomly initialized and optimized end-to-end using loss from Equation (1).

Definitions In the rest of the paper, we refer to the general combination of denoising and object detection using DOD. We refer to finetuning using Finetuning of Denoising and Object Detection (FDOD). Configurations where the detector network is concatenated after the denoising network, which are Freezing the denoiser, finetuning and sequential optimization, are indicated by *Seq* in tables. Optimizing a network in isolation is indicated by *Single*. Similarly, integrated configurations are indicated by *Int*. *Frozen* indicates that a method is using frozen denoiser weights. *Pretrained* indicates that a method is using pretrained denoiser weights.

4. Fully controlled experiments

In this section, we provide an analysis of all aforementioned strategies on a toy setup. This toy setup allows for fully controlled experiments.

Dataset. We use MNIST [9] to generate the toy dataset. A black square image of size 96x96 is created, and a few random MNIST digits are placed at a random location with a random size between 0.5 and 2x the original digit size, an example is shown in Figure 4a. Additionally, a random constant background shift is applied to the background. This is a randomly sampled value between 0.1 and 0.3, to prevent the denoiser from shortcutting the denoising task. To introduce noise to the images, Gaussian noise is added with $0 \leq \sigma \leq 2$. Increasing σ above 2 is not relevant, as images are like white noise at $\sigma = 2$.

Methods are evaluated by taking the average performance across a range of models that are each trained for a specific $\sigma \in \{.1, .2, .4, .8, 1.2, 1.6, 2.0\}$.

Network setup. The depth of the U-NET is set to 4 and the base feature channel width to 32. A tanh nonlinearity is used to force the final outputs to be between 0 and 1.

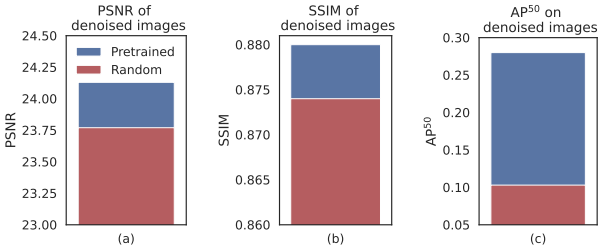


Figure 3. Training an integrated network with a frozen encoder using either random or pretrained weights initialization. In (a) and (b), the network is optimized for denoising, whilst the pretrained encoder is taken from a detection network. In the case of (c), The network is optimized for detection, with an encoder trained for denoising. Subfigures (a) and (b) show that performance using pretrained encoders is better than that of random ones. Results of (c) show that the performance of pretrained encoder features is more than 2.5 times better compared to randomly encoded features.

Motivation of integrating DOD. We propose the integrated optimization strategy under the assumption that denoising and object detection have subtasks in common, as this network uses a shared encoder. We test this hypothesis with two experiments: First, we train an integrated network for denoising with (i) a randomly initialized frozen encoder and (ii) a frozen encoder pretrained for detection. The results in Figure 3(a) and 3(b) show that performance using pretrained encoders is better than that of random ones. Second, we train an integrated network for detection with the difference of the first experiment being that (ii) is a frozen encoder trained for denoising instead of detection. The result in (c) shows that the pretrained encoder features can boost detection performance by more than 2.5 times compared to training on randomly encoded features. These results show that denoising and object detection have overlapping parts.

4.1. Denoising and Classification

To analyze the strategies on just denoising and classification, the ground-truth bounding box data is used to extract the relevant image areas for classification. We presume perfect localization this way. The classifier network is the encoder of UNet, followed by a global max pooling and a fully connected layer. This classifier allows for a fair comparison of all the different strategies. The number of digits in the images is fixed to two and the original sizes are kept. Because we use the ground truth bounding boxes to extract the patches containing the digits, adding more or resizing digits does not affect the classifier input much.

An additional two experiments are performed to establish a baseline. For the first experiment, we take a classification network trained on clean input images and evaluate it on noisy images. Secondly, we train a classifier on noisy images and evaluate them on noisy images.

Training settings. For denoising, a weighted combination of Charbonnier [3] and SSIM [36] loss was used. For classification, Cross Entropy loss was used. The total loss becomes:

$$\mathcal{L} = (\alpha_{\text{Char}} * \mathcal{L}_{\text{Char}} + \alpha_{\text{SSIM}} * \mathcal{L}_{\text{SSIM}}) * \lambda_{\text{Den}} + \mathcal{L}_{\text{CE}} * \lambda_{\text{Cls}} \quad (2)$$

α_{Char} and α_{SSIM} are empirically set to 0.8 and 0.2 respectively. λ_{Den} is set to 0.9 and λ_{Cls} is set to 0.1. These weights are also chosen from experimental results. For optimization, the Adam optimizer with a learning rate of 0.001 is used. For finetuning, a 10x smaller learning rate is used. A plateau learning rate scheduler is used with patience 20 and a multiplier of 0.1. Also, early stopping is used with patience 25. The maximum number of epochs was set to 100.

Results. Models are trained on an RTX 3090. The results of the experiments can be found in Table 1. We refer to Denoising and Classification using the acronym DC. Results show that the best denoising performance is achieved by optimizing the denoiser in isolation, and by training according to the DC-Int setup. These methods outperform other methods by up to 0.66 dB PSNR. Best classification accuracy is obtained when training a sequential DC configuration from scratch by a difference of 1% to the second-best strategy. However, denoising performance is the worst out of the DC configurations. Also, classification accuracy improves by almost 3% when adding denoising loss to the classification network, indicating that denoising the images is beneficial for classification performance. A visual comparison is shown in Figure 4.

Experiment					PSNR (dB) ↑	SSIM ↑	Acc (%) ↑
Name	Config	Train data	Frozen	Pretrained			
Classifier	Single	Clean	✗	✗	-	-	46.8
Classifier	Single	Noisy	✗	✗	-	-	76.7
Isolated	Seq	Both	✓	✓	**26.71**	0.943	63.9
DC	Seq	Noisy	✓	✓	**26.71**	0.943	76.0
FDC	Seq	Noisy	✗	✓	26.58	0.940	78.5
DC	Seq	Noisy	✗	✗	26.03	0.938	**79.4**
DC	Int	Noisy	✗	✗	26.69	**0.944**	73.0

Table 1. Results of combining classification and denoising on the toy dataset. *FDC* refers to Finetuning DC. In training data, *both* means the denoiser is trained using noisy data, and the classifier uses only clean data. The Isolated and DC-Int strategies achieve the best denoising performance, outperforming sequential optimization with random initialization by 0.66 dB PSNR. Randomly initialized sequential optimization achieves the best classification accuracy, beating others by at least 1%. However, it also has worst denoising compared to the other strategies.

(a) GT (b) σ = 1.2 (c) Frozen (d) FDC (e) DC-seq (f) DC-int

Figure 4. Denoised images from different Denoising and Classification (DC) strategies. Freezing the denoiser and integrated optimization perform best denoising, Sequential optimization is not able to reconstruct the digit, and finetuning is more blurry. (c) and (f) are therefore preferred.

4.2. Denoising and Localization

Denoising and localization experiments use the same training settings as the DC analysis, with a different loss function and network for localization. We use FasterRCNN [29], ignoring the classification loss. We set the backbone to be a UNet encoder, to ensure fair comparisons across all strategies. Anchor boxes are configured to have sizes of 10, 28, or 56, with an aspect ratio of 1. We use the losses from

Experiment			PSNR (dB) ↑	SSIM ↑	Mean IOU ↑
Config	Frozen	Pretrained			
Seq	✓	✓	**24.91**	**0.903**	0.538
Seq	✗	✗	24.88	0.902	**0.540**
Int	✗	✗	24.89	0.902	0.539

Table 2. Results of various strategies of combining denoising and localization. For both denoising and localization, all strategies have similar performance.

Experiment			PSNR (dB) ↑	SSIM ↑	AP^{50} ↑
Config	Frozen	Pretrained			
Seq	✓	✓	**24.91**	**0.903**	0.539
Seq	✗	✗	24.82	0.902	**0.543**
Int	✗	✗	24.85	**0.903**	0.538

Table 3. Denoising and detection results on the toy setup of various combining strategies. The different strategies show no significant difference.

FasterRCNN to localize digits, ignoring the classification part of the loss. The learning rate is set to 0.0001. As an evaluation metric, the mean Intersection over Union (IOU) for the predicted bounding boxes is used. We analyze three different strategies inspired by the classification experiments: freezing the denoiser and training a detector on these outputs, sequential optimization, and integrated optimization.

Results. Results can be found in Table 2. The results show that there is no significant difference between the different strategies, contrary to the classification experiments. It is interesting to note that the denoising performance of the sequential optimization from scratch is now the same as that of the frozen denoiser and integrated optimization.

4.3. Denoising and Detection

The analysis of denoising and detection uses the same networks and training settings as denoising and localization, no longer discarding outputs of FasterRCNN. To measure the performance of the detection, we use the AP^{50} metric.

Results. The results from Table 3 show that there is no significant difference between the different strategies. An interesting observation is that in both localization and detection experiments, sequential optimization from scratch performs similarly in denoising as other strategies. Some additional experiments were done to analyze the influence of including the detection loss in each strategy.

Influence of detection loss. An analysis of the influence of the detection loss is done by training the sequential optimization strategy using only classification loss on the exact same dataset as used for localization and detection. This dataset is the toy dataset including between 1 and 5 digits of varying sizes. The dataset used in classification experiments uses a fixed digit size and a fixed amount of digits. Additionally, the detector is trained on the dataset used in the classification experiments.

Experiment		PSNR (dB) ↑	SSIM ↑
Loss	Dataset		
Classification	Classification	26.03	0.938
Detection	Classification	**26.50**	**0.940**
Classification	Detection	24.19	0.895
Detection	Detection	**24.82**	**0.902**

Table 4. Effect of adding detection loss compared to just classification loss on the classification and detection datasets. Denoising loss is the same. The difference between these datasets is the number of digits and digits of varying sizes. The denoising networks are exactly the same, the only difference is in the classification/detection networks. Denoising performance is always better when detection loss is included.

Results. Results are shown in table Table 4. It shows that adding the detection loss improves the denoising quality of sequential optimization from scratch by 0.47 to 0.63 dB PSNR on both datasets.

5. Experiments on real data

We perform experiments with the various strategies on the PascalVOC dataset. As security cameras are the context of our analysis, we choose the following five relevant classes from PascalVOC: person, bike, truck, car and cat. For training, we use the trainval2012 set. For testing, we use the 2007 test set. FasterRCNN is used with a pretrained Resnet50-FPN backbone on Imagenet. We use the same denoiser as in the toy setup.

Training settings. The shortest side of images is resized to 800 pixels, to preserve aspect ratios. The longer side is capped at 1333 pixels. After this, image dimensions are rounded to the nearest multiple of 32, to ensure matching spatial resolutions for the skip connections of UNet. For each strategy, a single model is trained on a uniform range of σ between $[0, 1]$. Each image samples a σ from this interval. σ is capped at 1 because this noise is already distorting the input image significantly, as shown in Figure 9a. Adam optimizer is used with a base learning rate of $1e{-}4$. This learning rate is adapted per strategy for optimal performance, as explained in the following sections.

How to integrate denoising and detection? The integrated network used in the toy setup, attached the detector to the bottom of the UNet network, shown in Figure 2. Another approach to integrating the two networks is by attaching a denoising head to the ResNet50 backbone, visualized in Figure 5. An experiment is done to compare the performance of both approaches.

Experiment	PSNR (dB) ↑	SSIM ↑	AP^{50} ↑
UNet w/ detector	28.87	**0.814**	0.313
ResNet50 w/ denoiser	**28.96**	0.813	**0.631**

Table 5. Results show that attaching a denoiser head to a detector backbone results in over 2x better detection performance, whilst denoising performance is similar. This configuration is therefore preferred for subsequent experiments.

Results. Results shown in Table 5 show that attaching a denoising head to the backbone achieves more than 2x better detection performance, whilst denoising performance is similar. This configuration is therefore also used in subsequent experiments.

Denoising and Object Detection We optimize the different strategies including baseline experiments for denoising and detection. To ensure a fair comparison between all strategies, the denoiser network used in other strategies consists of a part of the ResNet-50 backbone up until *Stage 3*, followed by the denoising head shown in Figure 5.

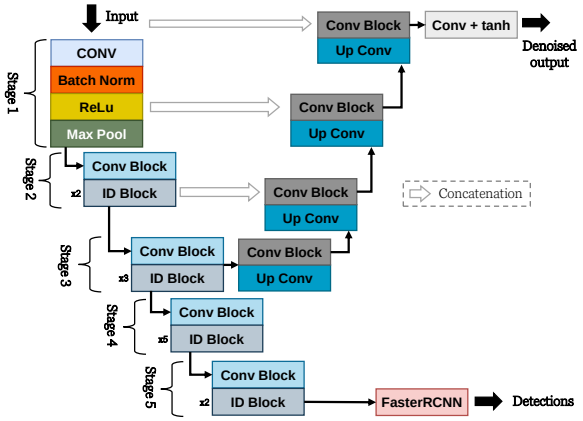


Figure 5. Figure illustrating how the denoising head is attached to the ResNet-50 backbone. The denoising head is the same as a UNet decoder. The FPN is left out as it is not used for the denoising head.

Results. The performance of all the different strategies is shown in Table 6. A visual comparison is shown in Figure 1 and 6. For each strategy, results are obtained using the best settings as found by ablation experiments. Training FasterRCNN only for detection on clean images achieves the best performance with 0.769 AP^{50}. In noisy images, DOD-Seq with frozen denoiser weights achieves the best overall performance. FDOD achieves slightly better denoising metrics, but 3.6 worse AP^{50}. Also, sequential optimization from scratch is better than integrated optimization, in both denoising and detection. Notably, results show that a combination of denoising and detection can outperform isolated detector optimization on noisy data by 12.8 AP^{50}.

Experiment						PSNR (dB) ↑	SSIM ↑	AP^{50} ↑
Name	Train data	Test data	Config	Frozen	Pretrained			
Detection	Clean	Clean	Single	✗	✗	-	-	**0.769**
Detection	Clean	Noisy	Single	✗	✗	12.02	0.107	0.162
Detection	Noisy	Noisy	Single	✗	✗	12.02	0.107	0.621
Isolated	Both	Noisy	Seq	✓	✓	29.73	0.830	0.474
DOD	Noisy	Noisy	Seq	✓	✓	29.73	0.830	0.749
FDOD	Noisy	Noisy	Seq	✗	✓	**29.80**	**0.831**	0.713
DOD	Noisy	Noisy	Seq	✗	✗	29.23	0.821	0.688
DOD	Noisy	Noisy	Int	✗	✗	28.96	0.813	0.629

Table 6. Comparison of the different strategies on a subset of the PascalVOC dataset. The DOD sequential setup with frozen and pretrained denoiser weights achieves the best overall performance in terms of denoising and detection. FDOD achieves slightly better denoising performance, but over 3 AP^{50} on detection. Adding a denoising loss to the detector network helps to improve detection performance by almost 13 AP^{50}.

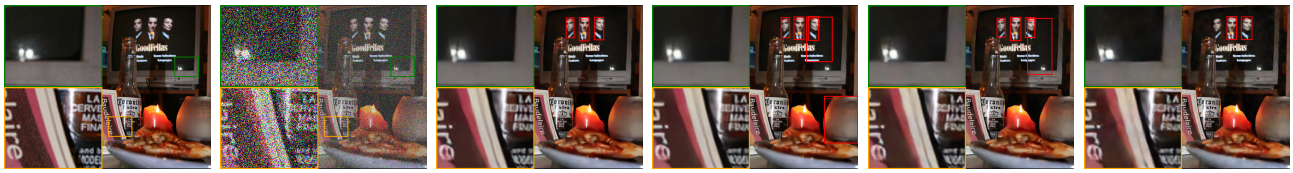
Experiment		PSNR (dB) ↑	SSIM ↑	AP^{50} ↑
Config	Learning rate			
Seq	Single	**29.18**	**0.821**	0.604
Seq	Task-specific	29.08	0.819	**0.684**
Int	Single	28.23	0.803	**0.635**
Int	Task-specific	**28.96**	**0.813**	0.629

Table 7. Results showcasing the effect of including task-specific learning rates, meaning that parts of the networks are optimized with different learning rates than other parts. Results show that task-specific learning rates boost one of the two tasks' accuracy, whilst not decreasing the other significantly.

Task-specific learning rate. An experiment is done to show the effect of using different learning rates for denoising and detection parts of the network. Learning rates used are $1e-4$ for denoising and $1e-5$ for detection. Results can be found in Table 7. They show that training with task-specific learning rates can boost the performance of a task, but sacrifices a bit of performance in the other. Sequential optimization from scratch improves 8 AP^{50} with denoising staying similar. Integrated optimization improves denoising quality by 0.74 dB PSNR and 0.01 SSIM, and detection performance only decreases by 0.06 AP^{50}.

Using pretrained weights. An ablation study is done to show the effect of initializing detector networks with pretrained weights. The weights are taken from a pretrained detector trained on clean images from the PascalVOC subsets. All experiments use task-specific learning rates. Results in Table 8 show that including a pretrained detector benefits denoising performance for all strategies. Detection performance does not always increase: sequential optimization with a frozen denoiser increases 5.6 AP^{50}, whilst integrated optimization decreases 2.2 AP^{50}. Optimal settings, therefore, vary per strategy.

Analyzing class performance. An in-depth analysis is done to investigate the performance of denoising and detection per class. Denoising metrics are calculated



(a) (b) (c) (d) (e) (f)

Figure 6. Denoised images with detections by different combination strategies for denoising and detection (Zoom in to see details). (a): GT. (b): Input image with $\sigma = 0.5$. (c): Freeze denoiser. (d): Finetuning. (e): Sequential optimization. (f): Integrated optimization. (c) has better detection and fewer denoising artifacts compared to other strategies.

Experiment				PSNR (dB) ↑	SSIM ↑	AP^{50} ↑
Config	Frozen	Pretrained denoiser	Pretrained detector			
Seq	✓	✓	✗	29.73	0.830	0.693
Seq	✓	✓	✓	**29.73**	**0.830**	**0.749**
Seq	✗	✗	✗	29.08	0.819	0.684
Seq	✗	✗	✓	**29.23**	**0.821**	**0.688**
Int	✗	✗	✗	28.96	0.813	**0.631**
Int	✗	✗	✓	**29.17**	**0.818**	0.609

Table 8. Effect of training various strategies starting from a pretrained detector network. The networks are pretrained on clean PascalVOC images. Task-specific learning rates are used. Initializing with pretrained networks boosts denoising performance up to 0.21 dB PSNR. However, detection performance decreases more than 2 AP^{50}, in the integrated configuration, compared to an increase of more than 5 AP^{50} when freezing the denoiser. Optimal settings, therefore, vary per strategy.

by extracting the ground truth bounding boxes out of the denoised and ground truth images. Results of this experiment can be found in Figure 7. Findings from Table 6 translate to class-specific results, with the finetuning strategy achieving the best denoising and freezing the denoiser achieving the best detection. Considering average performance on both tasks, freezing the denoiser is preferred as the detection performance is on average 3 AP^{50} better, with similar denoising performance.

Investigating noise levels. To provide more insight into the strategies' performance, methods are evaluated across a set of noise levels. Results are shown in Figure 8. In lower noise levels, strategies have similar denoising performance, but freezing the denoiser consistently achieves better detection. Freezing the denoiser is preferred as it achieves the best overall performance for each noise level.

Signal dependent noise. Gaussian noise is signal-independent, but many camera sensors experience signal-dependent noise. This noise distribution is different for different camera sensors. An experiment is done to investigate the behavior of the strategies with sensor noise applied to the images. A noise model of an OmniVision sensor is used for this experiment, with gain levels sampled uniformly from $[0, 1024]$. A gain level of 1024 is opted for, given that it is already a substantial amplification for this camera sensor. A comparison of Gaussian and sensor noise is shown in Figure 9. Results are found in Table 9. Freezing the denoiser and finetuning optimization strategies again achieve similar denoising performance, but FDOD is outperformed 3.0 AP^{50} by freezing the denoiser strategy. It is also interesting that the difference in performance between DOD-Seq without freezing the denoiser and DOD-Int is much smaller. This can be attributed to the fact that the images with the highest level of sensor noise that can be sampled are less noisy compared to the Gaussian noise counterpart, as shown in Figure 9.

Experiment					PSNR (dB) ↑	SSIM ↑	AP^{50} ↑
Name	Config	Frozen den	Pretrained den	Pretrained det			
DOD	Seq	✓	✓	✓	**32.62**	**0.894**	**0.803**
FDOD	Seq	✗	✓	✓	32.61	0.893	0.773
DOD	Seq	✗	✓	✓	31.88	0.884	0.749
DOD	Int	✗	✗	✗	31.88	0.883	0.729

Table 9. Results when applying signal-dependent sensor noise of an OmniVision sensor. Sequential and integrated optimization are much closer compared to Gaussian noise. This is because the highest level of sensor noise would be considered less noisy compared to the Gaussian noise, as shown in Figure 9. Freezing the denoiser is the preferred approach, reaching a similar denoising performance as FDOD, but outperforming over 3 AP^{50} on detection.

6. Discussion

The results from the toy setup show no clear preference between DOD strategies. When looking at DC results, Sequential optimization from scratch has the worst denoising of all configurations but achieves the best classification accuracy. Freezing the denoiser results in the best overall performance. When training on real-life images, FDOD achieves the best denoising performance but has more than 3 AP^{50} worse detection compared to freezing the denoiser. Freezing the denoiser is therefore preferred as the best approach for overall performance.

We show that denoising and detection can use features of the other task to extract information. Additionally, analysis of the toy setup indicates that the utilization of

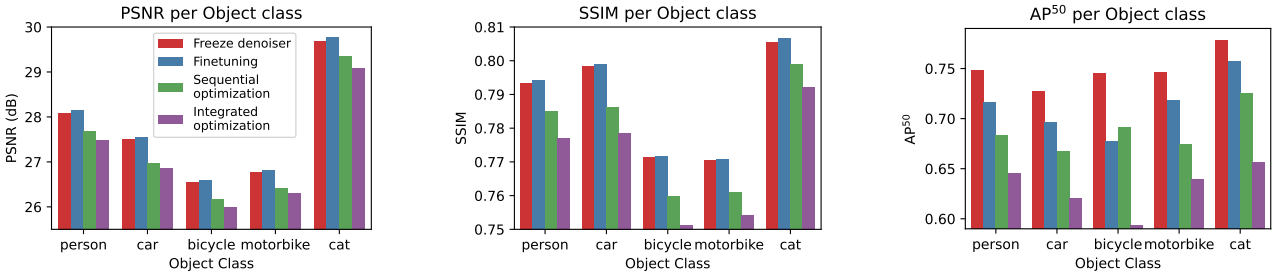


Figure 7. Class specific results on PascalVOC subset. Finetuning achieves the best denoising on all classes, but freezing the denoiser achieves the best detection performance. Overall, freezing the denoiser is preferred compared to the other strategies, as average performance is best.

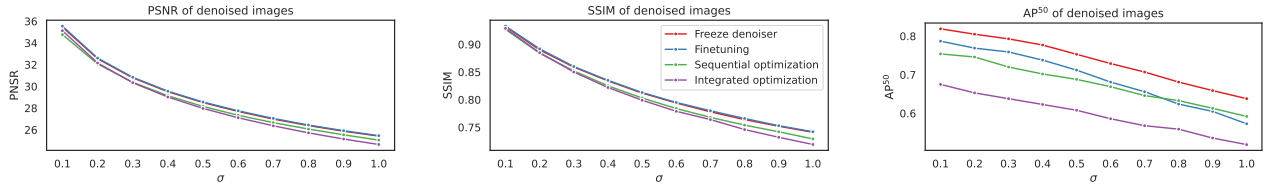


Figure 8. Performance of strategies across different noise levels. The x-axis represents the standard deviation of Gaussian noise. Finetuning and freezing the denoiser achieve similar denoising performance, but finetuning consistently has worse detection. Freezing the denoiser is therefore preferred considering all metrics.

(a) Gaussian noise (σ=1.0) (b) Sensor noise (gain = 1024)

Figure 9. Example images show that the sensor noise image is less noisy than the Gaussian noise image.

a localization loss enhances the denoising performance of the sequential optimization from scratch. Analyzing the performance across various noise levels in the real setup shows that the denoising performance of the various strategies is similar at lower noise levels. The results when using sensor noise support this finding. Using pretrained weights and task-specific learning rates can help improve the performance of both tasks.

By tuning the weights of the losses, the performance between denoising and detection can be tweaked. The optimal weights depend on the importance of each task. The weights proposed in this paper aim to give equal importance to both tasks, which might not be optimal for different domains. For finetuning, the learning rate is also an important setting. Setting the learning rate too high causes the network to behave similarly to sequential optimization from scratch. Finetuning with a lower learning rate is therefore important to ensure that the networks cannot change their weights a lot.

The denoising in this paper aims to achieve full denoising of the input image. When images have a high noise level, it is difficult to obtain a clean version of the original image. High-frequency information is lost, which could result in loss of texture such as disappearing faces and unreadable text (See Figure 1 and 6).

To preserve more high-frequency information, future work could analyze the effect of partial denoising instead of full denoising. Additionally, it is interesting to investigate if the findings from our analysis also hold for different combinations of detection and denoiser networks, such as transformer-based approaches.

7. Conclusion

Modern imaging systems, pivotal in domains such as medical imaging, autonomous driving, and security, grapple with noise from sensor measurements and electronic circuits, impacting downstream tasks including object detection. Despite this, denoising and detection's combined potential remains underexplored. This work addresses this gap by analyzing different naïve, existing, and novel strategies that optimize both image quality and detection performance. The results show that the best overall performance is obtained by optimizing a denoiser network in isolation and training a detector on the outputs of the denoiser. Analysis shows that combining detection and denoising is beneficial for detection performance. Investigating the effect of doing partial denoising is an interesting direction for future works. Additionally, different detector and denoising networks can be analyzed to broaden this analysis.

References

[1] Harold C. Burger, Christian J. Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399, 2012. 2

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[3] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 168–172 vol.2, 1994. 4

[4] Linwei Chen, Ying Fu, Kaixuan Wei, Dezhi Zheng, and Felix Heide. Instance segmentation in the dark. *International Journal of Computer Vision*, pages 1–21, 2023. 1

[5] Qiang Chen, Jian Wang, Chuchu Han, Shan Zhang, Zexian Li, Xiaokang Chen, Jiahui Chen, Xiaodi Wang, Shuming Han, Gang Zhang, et al. Group detr v2: Strong object detector with encoder-decoder pretraining. *arXiv preprint arXiv:2211.03594*, 2022. 2

[6] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 2

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29, 2016. 2

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 2

[9] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 3

[10] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2016. 1

[11] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019. 2

[12] Chi-Mao Fan, Tsung-Jung Liu, and Kuan-Hsien Liu. Sunet: swin transformer unet for image denoising. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2333–2337. IEEE, 2022. 2

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[16] Yang Hong, Kaixuan Wei, Linwei Chen, and Ying Fu. Crafting object detection in very low light. In *BMVC*, volume 1, page 3, 2021. 1, 2

[17] Aliia Khasanova, Alisa Makhmutova, and Igor Anikin. Image denoising for video surveillance cameras based on deep learning techniques. In *2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 713–718. IEEE, 2021. 1

[18] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021. 2

[19] 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. 2

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[21] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-level wavelet-cnn for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 773–782, 2018. 2

[22] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018. 2

[23] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *Advances in neural information processing systems*, 29, 2016. 2

[24] S Milyaev and I Laptev. Towards reliable object detection in noisy images. *Pattern Recognition and Image Analysis*, 27:713–722, 2017. 1, 2

[25] Igor Morawski, Yu-An Chen, Yu-Sheng Lin, Shusil Dangi, Kai He, and Winston H Hsu. Genisp: neural isp for low-light machine cognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 630–639, 2022. 1, 2

[26] Tiago S Nazaré, Gabriel B Paranhos da Costa, Welinton A Contato, and Moacir Ponti. Deep convolutional neural networks and noisy images. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 22nd Iberoamerican Congress, CIARP 2017, Valparaíso, Chile, November 7–10, 2017, Proceedings 22*, pages 416–424. Springer, 2018. 1

[27] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10213–10224, 2021. 2

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2, 4

[30] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019. 2

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 2

[32] Sujin Shin, Youngjung Kim, Insu Hwang, Junhee Kim, and Sungho Kim. Coupling denoising to detection for sar imagery. *Applied Sciences*, 11(12), 2021. 1, 2

[33] Maheep Singh, Mahesh C Govil, Emmanuel S Pilli, and Santosh Kumar Vipparthi. Sod-ced: salient object detection for noisy images using convolution encoder–decoder. *IET Computer Vision*, 13(6):578–587, 2019. 2

[34] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 2

[35] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001. 2

[36] Zhou 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. 4

[37] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17683–17693, June 2022. 2

[38] Kushagra Yadav, Dakshit Mohan, and Anil Singh Parihar. Image detection in noisy images. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 917–923. IEEE, 2021. 2

[39] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, 126:103514, 2022. 2

[40] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6360–6376, 2021. 2

[41] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017. 2

[42] Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training. *arXiv preprint arXiv:2211.12860*, 2022. 2

3

Background

3.1. Deep learning background

This section explains some building blocks for deep learning that are more advanced and less well-known. Specifically, we will focus on GroupNorm [8] and transposed convolutions.

Group Normalization Including normalization layers like BatchNorm [9] in neural networks has been shown to improve training stability and speed up convergence. Santurkar et al. [10] propose that it helps to smoothen the loss function over the parameter space. The downside of BatchNorm is that performance starts to degrade when the batch sizes are reduced. Fitting large batch sizes in memory can be a challenge nowadays, with the amount of high-resolution training data available. A visualization of BatchNorm and other normalization techniques is shown in figure 3.1

Two normalization approaches that aim to overcome limitations of BatchNorm are LayerNorm [11] and InstanceNorm [12]. Instead of normalizing across the batch dimension, they normalize per entry in a batch. LayerNorm normalizes features across all channels, and InstanceNorm normalizes features per channel.

GroupNorm [8] can be seen as a tradeoff between LayerNorm and InstanceNorm. They divide the channels in G groups. The number of channels should be divisible by G. If $G = 1$, we get LayerNorm. Similarly, if $G = C$, we get InstanceNorm, as C represents the number of channels. The mean μ_G and standard deviation σ_G are calculated for each group.

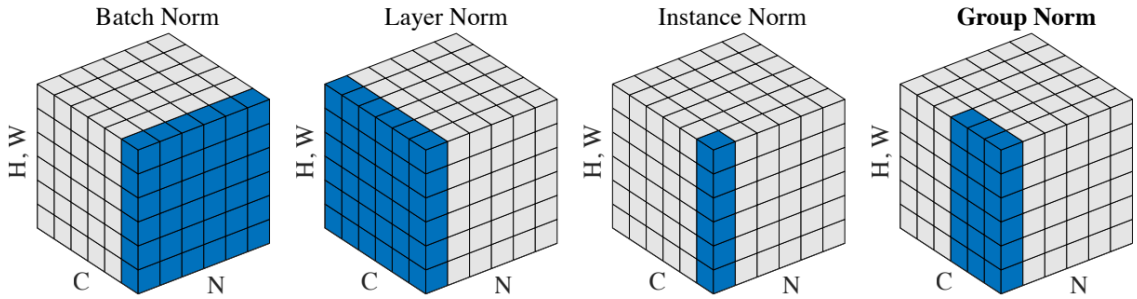


Figure 3.1: Illustration of various normalization methods used in neural networks. The blue part indicates across what dimensions normalization is applied. N represents the batch dimension, C represents the channel dimension, and H,W represent spatial dimensions. In this figure, GroupNorm parameter $G = 2$. Image taken from the GroupNorm [8] paper.

They are used to normalize the feature activations using the following formula:

$$\text{GroupNorm}(x, g) = \frac{x - \mu_g}{\sigma_g} * \gamma_g + \beta_g \tag{3.1}$$

Where:

x is the input tensor.
g is the group index or identifier.
μ_g is the mean of the elements in group g.
σ_g is the standard deviation of the elements in group g.
γ_g is a learnable scaling parameter for group g.
β_g is a learnable shifting parameter for group g.

The learnable parameters are included so that the network can learn to undo the normalization of the groups.

Transposed convolution

Normal convolutions are performed by sliding a kernel over an input feature map, performing element-wise multiplication and summing the results to produce an output feature map. Transposed convolutions do the opposite: Each value of the input is multiplied with the kernel. The resulting values of this multiplication are summed and inserted in an output grid. This process is visualized in figure 3.2. This process results in an output with increased spatial dimensions.

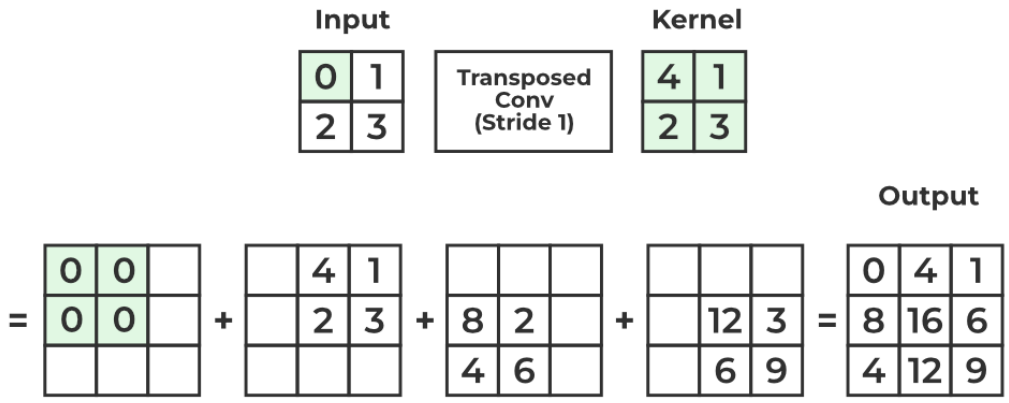


Figure 3.2: Visualization of performing a transposed convolution. The kernel is multiplied with each element of the input. The results of these calculations are placed on the output grid and summed. Stride and padding control the size of the output. Image taken from `geeksforgeeks.org`.

3.2. Denoising

Definition

Image denoising is the task of removing unwanted noise from an image. The noise can have various sources, such electronic measurement noise or camera sensor noise. Additive white Gaussian noise (AWGN) is a noise model that can be used to simulate random noise. *Additive* refers to the fact that this noise is added to the raw input signal. *White* noise means that the noise intensity is equivalent across all input signals. *Gaussian* means that the noise signal is drawn from a Gaussian distribution with zero mean. In other words, each image pixel is constructed of a raw input signal with some noise added on top. Mathematically this can be formulated as follows:

$$\mathbf{y} = \mathbf{x} + \mathbf{n} \tag{3.2}$$

Here $\mathbf{y} \in \mathbb{R}^N$ represents the noisy image, $\mathbf{x} \in \mathbb{R}^N$ is the ideal clean image and $\mathbf{n} \in \mathbb{R}^N$ is the independently and identically distributed noisy signal. Image denoising is concerned with recovering the clean signal $\mathbf{x}$ given $\mathbf{y}$.

Next to AWGN, there are other noise models that can be used. Camera sensors produce noisy signals caused by the randomness of photon arrival. Noise can also be introduced by the digital circuits present in cameras. These signals have different behaviour than the AWGN discussed above. They are signal dependent, and can be represented by a multiplication with the signal instead of addition.

Denoising networks

More traditional approaches try to denoise images by using hand-crafted methods, such as the works by Buades et al. [13], Dabov et al.[14] and Lan et al.[15]. The downside of these methods is that they often need hand-tuned parameters and are complex to optimize or slow in usage. With the development of the deep learning field and the availability of faster hardware, neural networks were used to perform image denoising. Instead of handcrafting the features as done in the more traditional approaches, by using deep learning the feature representations can be learned in an end-to-end fashion. These deep learning based denoisers outperform classical hand-crafted approaches, as is shown in Figure 3.3 found in the survey by Elad et al. [16].

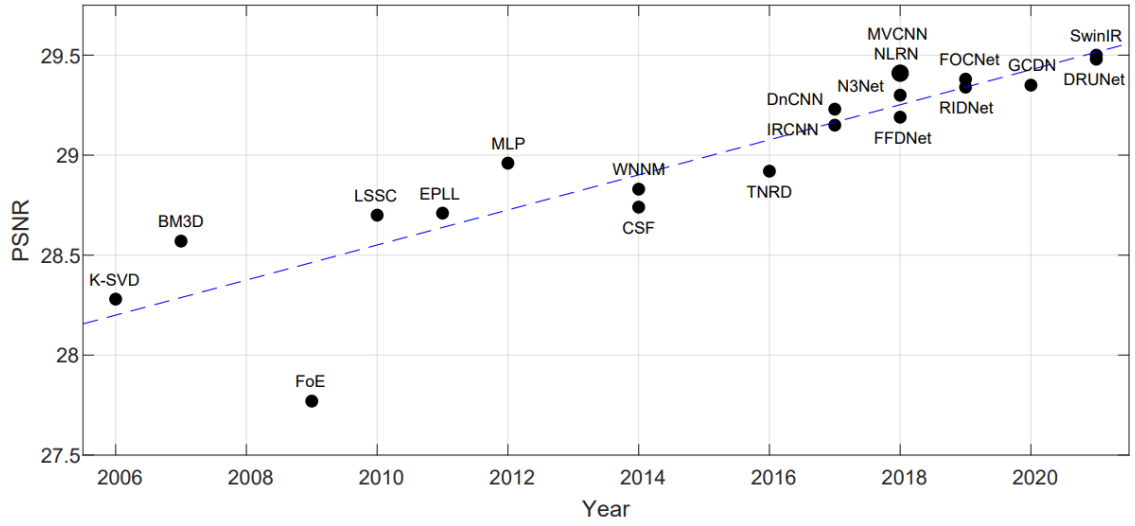


Figure 3.3: PSNR score of various denoising methods on the BSD86 dataset [17]. Recent deep learning based approaches outperform more traditional methods.

UNet

The denoising network used in this paper is UNet [18]. This network is well known and widely used. Many followup works are based upon this network. We also choose it because of its simplicity and its modular architecture. The network architecture can be seen in Figure 3.4. The network can be split up into two different parts: the encoder and the decoder. The encoder is responsible for encoding the image in an efficient representation and capturing the context of the image. The decoder is responsible for reconstructing the original clean image in the original resolution.

The encoder consists of multiple layers. Between each layer, a max pooling operation is applied which reduces the spatial resolution by a factor of 2. The decoder has the same amount of layers, and in each layer

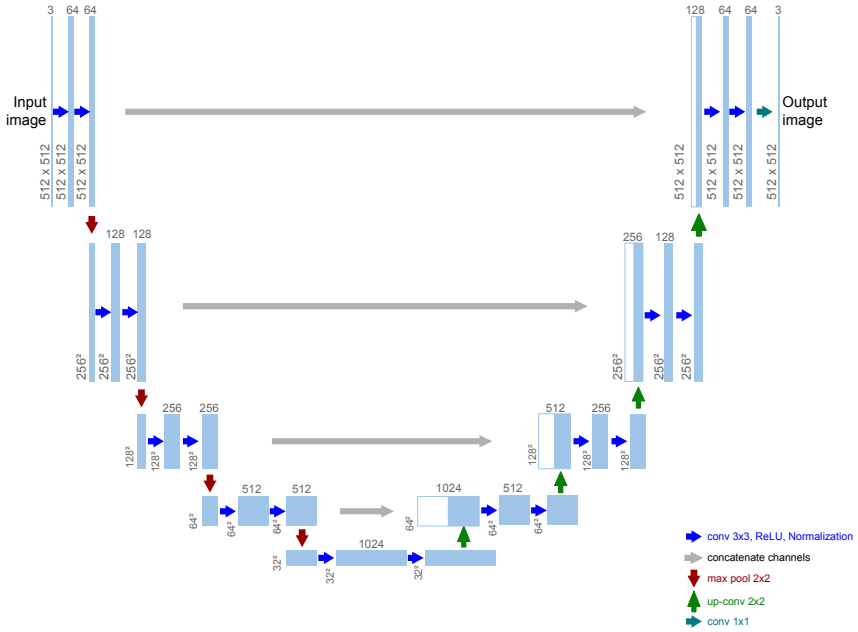


Figure 3.4: Visualization of UNet network architecture. The networks takes in an RGB image and outputs another RGB image. the concatenation of channels allows for efficient learning and preserves detailed information.

a transposed convolution is used to increase the feature resolution by a factor of 2, see section 3.1 for an explanation. The concatenation of the channels from the encoder in the blocks of the decoder allows the network to produce high resolution outputs with only limited training data.

The layers consist of two convolutional blocks, each block consisting of a ReLU followed by a normalization. The normalization used in the scientific article is GroupNorm, explained in section 3.1. The analysis in the scientific article adds an additional tanh nonlinearity after the final 1x1 convolution to force the outputs to be between 0 and 1.

Image quality

To be able to quantitatively analyze denoising performance on images, there exist various metrics to determine the quality of an image. These metrics compare a distorted image with an ideal ground truth image. The metrics used in the paper are Peak Signal to Noise Ratio (PNSR) and Structural Similarity Index Measure (SSIM).

PSNR The PSNR is a metric that compares pixel values between a reference image and a distorted image. The metric has a logarithmic scale, expressed in terms of dB. It uses the Mean Squared Error (MSE) to calculate its values.

The MSE is defined as follows:

$$\text{MSE} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - I'(i,j))^2 \tag{3.3}$$

Where:

M : Height of the image

N : Width of the image

$I(i,j)$: Pixel value of the original image at position (i,j)

$I'(i,j)$: Pixel value of the reconstructed image at position (i,j)

PSNR has two advantages over the MSE, namely (i) it is independent of the number of bits used to encode the images and (ii) it is a logarithmic scale resulting in values typically between 0 and 60, which is much easier

Figure 3.5: Leftmost image is a reference image, other images are images with various distortions applied. All these distorted images have the same MSE and PSNR value, even though perceptual quality is very different. The SSIM metric does distinguish between these images. Image taken from the work of Wang et al. [19].

Figure 3.6: Example scenario where SSIM fails to indicate the better quality images. Leftmost image is the groundtruth image. The other images are distorted versions of this image. The SSIM value of the two right images is higher than that of the second image, even though visually the right images look worse. The PSNR metric does rank images as expected. Image taken from the work of Kotevski et al. [20]. This motivates the use of both SSIM and PSNR to quantitatively asses image quality.

to compare than the large values the MSE produces. PSNR can be seen as a normalized version of the MSE. PSNR is defined as follows:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right) \tag{3.4}$$

Where:

PSNR : Peak Signal-to-Noise Ratio
MAX : The maximum possible pixel value (usually 255 for an 8-bit image)
MSE : Mean Squared Error

SSIM The downside of metrics that are based on comparing pixel values, is that two very visually different images can have the same metric value. The work of Wang et al. [19] showcased these shortcomings and proposed a new image quality metric: Structural similarity index measure (SSIM). An illustration of this problem is given in figure 3.5. Instead of comparing pixel values, SSIM compares structural elements of the pixels. Images that have the same PSNR now get different values for SSIM, solving the shortcomings of pixel based image quality metrics. SSIM is calculated on image patches of size NxN, typically 11x11. The formula to calculate SSIM on an NXN window is as follows:

$$\text{SSIM}(I, I') = \frac{(2\mu_I\mu_{I'} + C_1)(2\sigma_{I,I'} + C_2)}{(\mu_I^2 + \mu_{I'}^2 + C_1)(\sigma_I^2 + \sigma_{I'}^2 + C_2)} \tag{3.5}$$

Where:

I, I' : Original and reconstructed image patch
$\mu_I, \mu_{I'}$: Mean pixel intensities of I and I'
$\sigma_I, \sigma_{I'}$: Standard deviations of I and I'
$\sigma_{I,I'}$: Cross-correlation between I and I'
C_1, C_2 : Constants to stabilize the division with weak denominator

To compute the SSIM of an entire image, the average is taken of all the SSIM values on the patches. SSIM is a value between -1 and 1, where -1 represents perfect anti-correlation, and 1 perfect similarity.

The SSIM metric however also has scenarios where high values do not necessarily correspond to high image quality. The work of Kotevski et al. [20] shows examples of cases where SSIM metric fails to rank images based on visual quality. This is shown in figure 3.6. The two right images in this figure have a higher SSIM score than the second image. PSNR does rank images as expected in this example.

These examples show that image quality metrics like PSNR and SSIM both have their shortcomings and advantages. To provide a more reliable quantitative analysis in Chapter 2, both SSIM and PSNR are used to assess image quality.

Loss function

PNSR is not a differentiable metric due to the *MAX* term in the equation (3.4). Therefore it cannot be used directly as a loss function. SSIM is differentiable, and can be used as a loss function. Because we want to minimize loss, we obtain the following SSIM loss:

$$\mathscr{L}_{\text{SSIM}} = 1 - \text{SSIM(I, I')} \tag{3.6}$$

Where:

I, I' : Original and reconstructed image patch

We subtract the SSIM metric from 1 to minimize the loss, and therefore maximizing the SSIM value. This loss is also used in the scientific article of Chapter 2.

Charbonnier loss As shown before, no single image metric always perfectly corresponds to to visual quality. Therefore, another loss is used in combination with SSIM loss to get the best of both worlds. This loss is a pixel-based loss that can be seen as a mix between L1 and L2 loss. The downside of L2 loss is that it is sensitive to outliers, and gradients of the L1 loss are always the same, even when coming close to the minimum. This can cause optimization to overshoot the minimum. Charbonnier loss has a smooth shape around the minimum, reducing the gradient magnitudes, but has smaller values when further away from the minimum than L2, adressing the outlier problem. It has the following formula:

$$\mathscr{L}_{\text{Charbonnier}} = \sqrt{(x-y)^2 + \epsilon^2} \tag{3.7}$$

x : Distorted image

y : Reference image

ϵ : Small constant to control the tradeoff between L1 and L2, usually 1e−3

3.3. Image classification and object detection

Image classification is the task concerned with assigning a label to a complete image based on its contents. Pictures of a dog would be labeled as 'dog' for example. Neural networks can solve this task by training on large datasets of labeled images. A popular dataset for this is called ImageNet [21]. This dataset has over 14 million annotated images with 1000 different classes. Convolutional neural networks are used to extract discriminative features out of images, which are fed into a linear layer that has 1000 output heads, where each head corresponds to a different label. To be able to properly extract discriminative features for good classification accuracy, deep networks are required. The problem with these deep networks is that the gradient vanishes when the number of layers becomes too large. The work of He et al. [22] solved this issue by introducing residual connections, preserving gradients even if networks consist of a lot of layers. Their work introduced ResNet, a deep network able to achieve state-of-the-art performance at the time on the ImageNet dataset. Concepts introduced in their work are the basis of many networks that achieve state-of-the-art performance today.

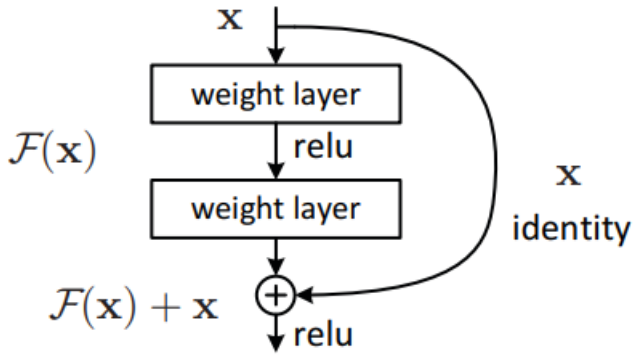


Figure 3.7: An example of a residual building block. The input features x are added to the output of several weight layers. This allows gradients to flow back into the earlier layers without vanishing. Image taken from the work of He et al. [22].

The paper introduces residual connections: A connection where the input features are added to the outcome of applying weight layers to the input. The advantage of this is that gradients can flow freely through this identity connection. A visual explanation is given in figure 3.7.

These residual building blocks are used in very deep networks like ResNet. A ResNet network is built by first applying a 7x7 convolution to increase the receptive field of subsequent layers. Then normalization, nonlinearity and pooling is applied. Next, many residual blocks are stacked on top of each other. Finally, a global average pooling and linear layer are applied to obtain predictions. The architecture is shown in figure 3.8.

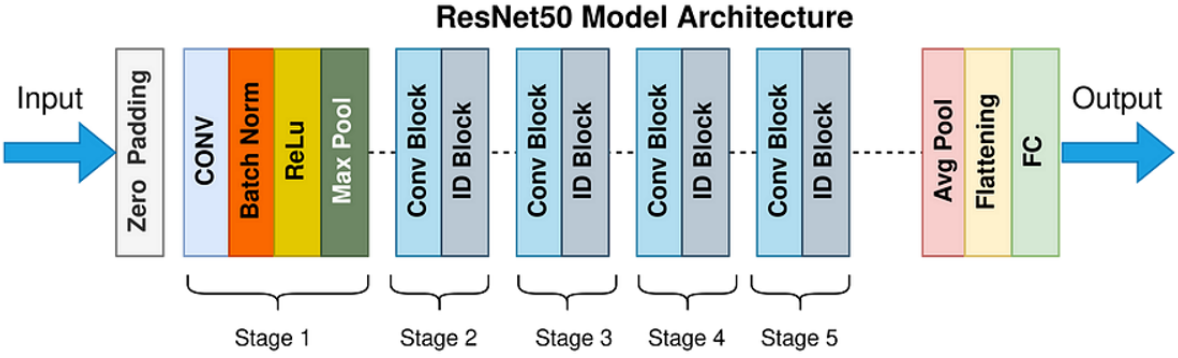


Figure 3.8: A visual representation of the ResNet architecture. ID Blocks refer to the residual blocks shown in figure 3.7. Different networks can be created depending on how many residual blocks are stacked in stages 2-5. Image by Suvaditya Mukherjee.

Detection Object detection is an extension of image classification. Instead of assigning only a label to the entire image, we now also want to localize the object. This is done by generating a bounding box around the

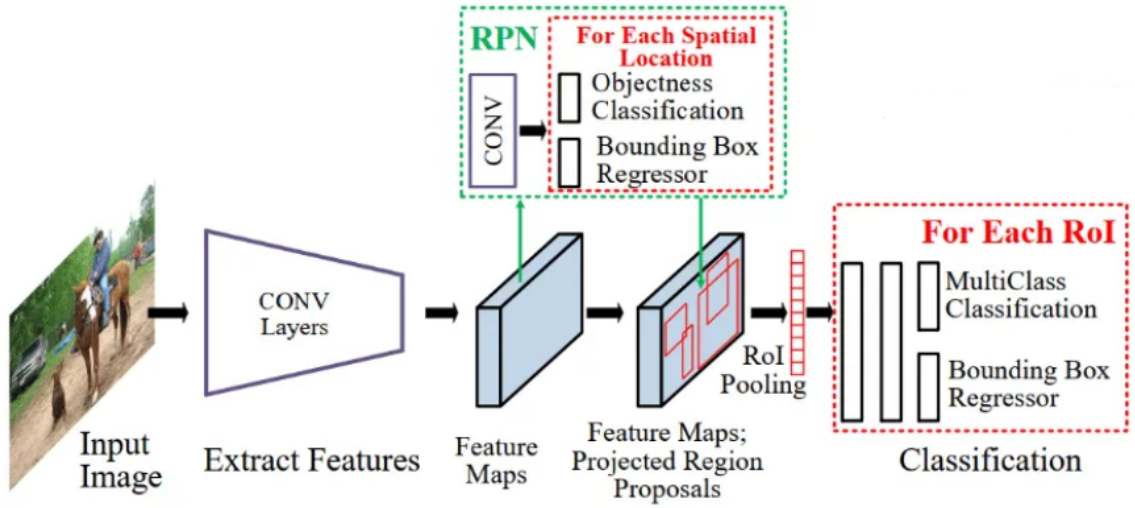


Figure 3.9: Illustration of the FasterRCNN architecture. Features are extracted using a deep convolutional network, like ResNet. A region proposal network proposes regions in the features that contain objects. All the proposals containing objects are projected onto the feature maps, and converted to a fixed-size feature vector using RoI pooling. These feature vectors are given to the detection module, that classifies them and regresses the bounding boxes. Image by Ashutosh Makone.

object. Another difference with image classification is that an image can contain multiple objects, of different labels. Object detection tries to localize and classify each of them.

FasterRCNN [23] is a popular object detector, successor of RCNN [24] and FastRCNN [25]. This object detector consists of two stages: (i) Propose regions that contain objects, and (ii) classify these regions. This is shown in figure 3.9. ResNet is often used as a backbone to extract useful features from an input image. The final 3 blocks shown in figure 3.8 are left out, and the output of stage 5 is given to FasterRCNN.

Region proposal network The extracted features are given to a Region Proposal Network (RPN). This RPN generates so-called *anchor boxes* for each spatial location. Anchor boxes are boxes with different sizes and aspect ratios, defined in advance. Usually, a total of 9 anchor boxes are used, with 3 different scales and 3 different aspect ratios. These anchor boxes are then classified if they contain an object or not, and bounding box coordinates are regressed. This is illustrated in figure 3.10.

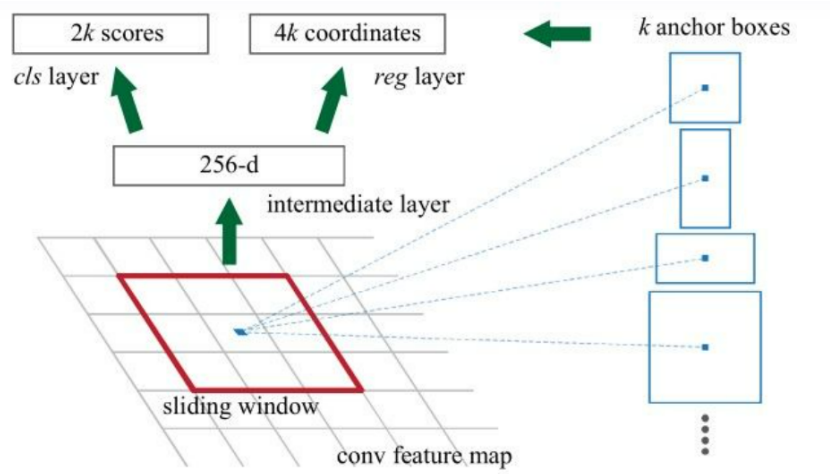


Figure 3.10: Illustration of anchor box generation. Anchor boxes of various scales and aspect ratios are projected onto each spatial location of the feature map. Each of these boxes is then classified to be either foreground (containing an object) or background, and bounding box coordinates are regressed.

Relevant anchor boxes are sampled from all of the boxes and projected onto the feature map. RoI pooling is applied to convert the projected anchor boxes to have a same-sized feature vector. This feature vector is

then given to the detection module, that assigns labels to each RoI and regresses the bounding box coordinates. Cross Entropy loss is used to train the classifiers of the RPN, and Smooth L1 Loss is used to train the regressors of the bounding boxes. The total loss is a weighted combination of these losses:

$$\mathscr{L}_{total} = (\mathscr{L}_{cls} + \lambda_{reg}\mathscr{L}_{reg}) + (\mathscr{L}_{rpn\_cls} + \lambda_{rpn\_reg}\mathscr{L}_{rpn\_reg}) \tag{3.8}$$

With:

$\mathscr{L}_{\text{total}}$: Total loss, combining object detection and RPN losses.
$\mathscr{L}_{\text{cls}}$: Cross Entropy loss in object detection.
$\mathscr{L}_{\text{reg}}$: Smooth L1 regression loss in object detection.
λ_{reg} : Balancing parameter for detection head. Usually set to 1.
$\mathscr{L}_{\text{rpn\_cls}}$: Cross Entropy loss in RPN.
$\mathscr{L}_{\text{rpn\_reg}}$: Smooth L1 regression loss in RPN.
$\lambda_{\text{rpn\_reg}}$: Balancing parameter for RPN. Usually set to 10.

Feature Pyramid Networks A Feature Pyramid Network (FPN) [26] is a network that takes a single scale image as input, and outputs feature maps of varying scales. This is useful for object detection, as smaller objects are difficult to detect when the spatial resolution becomes smaller in the ResNet backbone used by FasterRCNN. FPN can be merged with ResNet, providing FasterRCNN with different scales of feature maps. RoI's of larger objects can be projected onto smaller feature maps, and RoI's of smaller objects can be projected onto larger feature maps. This guides the network to let bigger feature maps focus on extracting the smaller objects, and the smaller feature maps are concerned with extracting larger objects. A visual of the new pipeline is given in figure 3.11.

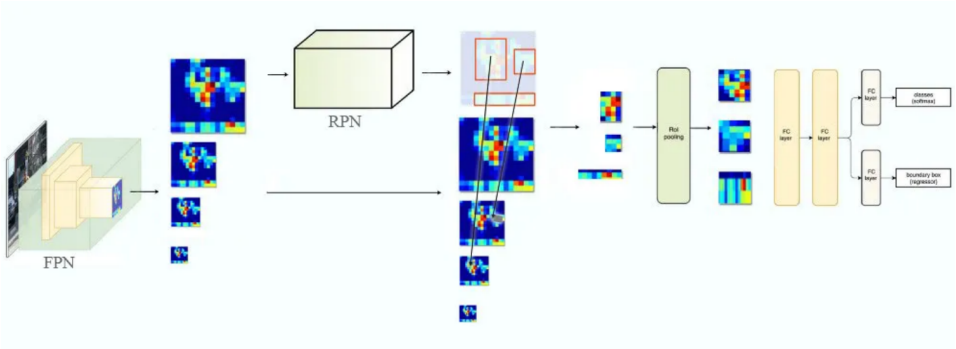


Figure 3.11: Pipeline of combining FPN with FasterRCNN. Features are extracted at different scales, and anchor boxes are projected on scales depending on the size of the anchor box. Smaller feature maps are responsible for detecting larger objects, whilst smaller objects are projected onto the bigger scale feature maps. Image by Jonathan Hui.

Bibliography

[1] T. S. Nazaré, G. B. P. da Costa, W. A. Contato, and M. Ponti, "Deep convolutional neural networks and noisy images," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 22nd Iberoamerican Congress, CIARP 2017, Valparaiso, Chile, November 7–10, 2017, Proceedings 22*, Springer, 2018, pp. 416–424.

[2] L. Chen, Y. Fu, K. Wei, D. Zheng, and F. Heide, "Instance segmentation in the dark," *International Journal of Computer Vision*, pp. 1–21, 2023.

[3] S. Milyaev and I. Laptev, "Towards reliable object detection in noisy images," *Pattern Recognition and Image Analysis*, vol. 27, pp. 713–722, 2017.

[4] A. Khasanova, A. Makhmutova, and I. Anikin, "Image denoising for video surveillance cameras based on deep learning techniques," in *2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, IEEE, 2021, pp. 713–718.

[5] I. Morawski, Y.-A. Chen, Y.-S. Lin, S. Dangi, K. He, and W. H. Hsu, "Genisp: Neural isp for low-light machine cognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 630–639.

[6] Y. Hong, K. Wei, L. Chen, and Y. Fu, "Crafting object detection in very low light," in *BMVC*, vol. 1, 2021, p. 3.

[7] S. Shin, Y. Kim, I. Hwang, J. Kim, and S. Kim, "Coupling denoising to detection for sar imagery," *Applied Sciences*, vol. 11, no. 12, 2021, ISSN: 2076-3417. DOI: 10.3390/app11125569. [Online]. Available: https://www.mdpi.com/2076-3417/11/12/5569.

[8] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[10] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[12] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[13] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Ieee, vol. 2, 2005, pp. 60–65.

[14] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[15] X. Lan, S. Roth, D. Huttenlocher, and M. J. Black, "Efficient belief propagation with learned higher-order markov random fields," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part II 9*, Springer, 2006, pp. 269–282.

[16] M. Elad, B. Kawar, and G. Vaksman, "Image denoising: The deep learning revolution and beyond–a survey paper–," *arXiv preprint arXiv:2301.03362*, 2023.

[17] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, IEEE, vol. 2, 2001, pp. 416–423.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[19] 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*, vol. 13, no. 4, pp. 600–612, 2004.

[20] Z. Kotevski and P. Mitrevski, "Experimental comparison of psnr and ssim metrics for video quality estimation," in *International Conference on ICT Innovations*, Springer, 2009, pp. 357–366.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database."

[22] 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*, 2016, pp. 770–778.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[25] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.