Dissertation
submitted to the Combined Faculties
for the Natural Sciences and Mathematics
of the Ruperto-Carola University of Heidelberg, Germany
for the degree of Doctor of Natural Sciences

put forward by
Christoph Kamann
born in Regensburg, Germany
Date of oral exam:

ROBUST SEMANTIC SEGMENTATION WITH DEEP LEARNING

Advisor:
Prof. Dr. Carsten Rother

Acknowledgements

This dissertation results from a joint project between Bosch Research and Heidelberg University, and I highly appreciate the opportunity to be part of this. I especially want to thank Prof. Dr. Carsten Rother to be my advisor for the project at the Faculty of Mathematics and Computer science and for allowing me to be a member of his team. I highly appreciate every fruitful conversation, the scientific freedom given to me, and all the reviews that taught me how to write papers. I also want to thank Prof. Dr. Jähne for agreeing to be the second advisor.

Further, I want to particularly express my gratitude to Hartmut Spennemann for allowing working in his team at Bosch. From Hartmut, I always received support in every aspect during the past three years.

It is extraordinary gratitude goes to Burkhard Güssefeld, who always took time to support me, especially during the paper periods. His technical advice and experience helped me essentially.

I want to express my gratitude to Marzena Franek for her continuous encouragement during my time at Bosch, which has always been supportive.

My gratitude also goes to Jan Hendrik Metzen. Even though from a different department at Bosch, the discussions with Jan helped me in many aspects.

Special thanks also to my colleagues Johannes Hofmann and Sebastian Kotzur for motivation and out-of-topic conversations.

I want to thank my brother Alexander for numerous reviews and my parents, Christine and Klaus, for their continuous support and encouragement.

Finally, I want to thank Johanna for all the support in the last years.

Zusammenfassung

Die semantische Bildsegmentierung ist ein weit verbreitetes Gebiet des Computersehens mit vielfältigen Anwendungsmöglichkeiten in der medizinischen Diagnostik oder der Umgebungserfassung bei autonomen Transport. Der Stand der Technik von Bildverarbeitungsalgorithmen für die semantische Segmentierung wird durch künstliche tiefe neuronale Netze festgelegt. Beim Entwurf eines Segmentierungsnetzwerks ist es wichtig, die Robustheit des Moduls in Bezug auf eine Vielzahl von Bilddegradationen zu verstehen. In dieser Arbeit präsentieren wir eine umfassende Studie zur Netzwerkrobustheit für die semantische Bildsegmentierung. Unsere Studie besteht aus zwei Teilen. Im ersten Teil steht das Robustheitsverhalten neuronaler Netzwerkarchitekturen und Netzwerkeigenschaften im Fokus. Wir verwenden eine Datenbank mit fast 400.000 Bildern, die aus den Datensätzen Cityscapes, PASCAL VOC 2012 und ADE20K generiert wurde, um die Performanz neuronaler Architekturen zu bewerten. Wir vergleichen vollständige Architekturen und auch bestimmte architektonische Eigenschaften, die in der semantischen Segmentierung etabliert sind. Wir betrachten auch den Dateninput, auf der wir uns mit den Generalisierungsperformanz von neuronalen Netzwerken beschäftigen. Die Robustheitserhöung ist der Fokus des zweiten Teils dieser Arbeit. Wir bauen auf einer Erkenntnis aus der Bildklassifizierung auf, mit welcher die Netzwerkrobustheit verbessert werden kann, indem die Netzwerkneigung gegenüber Objektformen erhöht wird. Wir präsentieren ein neues Trainingsschema, das diese Netzwerkneigung erhöht. Unsere Grundidee besteht darin, einen Teil der RGB-Trainingsbilder mit gefälschten Bildern zu mischen, bei dem jedem Klassenetikett eine feste, zufällig ausgewählte Farbe zugewiesen wird, die in realen Bildern nicht erscheint. Dies zwingt das Netzwerk stärker Objektformen des Bildinhaltes zur Segmentierungsentscheidung heranzuziehen. Wir nennen diese Datenerweiterungstechnik “Painting-by-Numbers”. Wir demonstrieren die Effektivität dieses Schemas mit einer umfangreichen experimentellen Auswertung und stellen eine Methode zur Validierung solcher Techniken vor.

Abstract

Semantic image segmentation is a widely studied field in computer vision with a diverse set of applications in medical diagnostics or autonomous transportation. Neural networks set the state-of-the-art of vision algorithms for semantic segmentation. Understanding the robustness of the network's module with respect to a diverse set of image corruptions is essential when a segmentation network is developed. In this thesis, we present an exhaustive study of network robustness for semantic segmentation. Our study is separated into two parts. Firstly, understanding the robustness of neural networks. We utilize a database of almost 400,000 images created from PASCAL VOC 2012, ADE20K, and the Cityscapes dataset for evaluating the performance of neural architectures. We benchmark entire neural network architectures as well as particular architectural properties established for semantic segmentation. We also view the data-driven side, where we take on a look at such networks' generalization capabilities. Based on the first part, we focus on increasing robustness in the second part of this thesis. We build upon an insight from image classification that output robustness can be improved by increasing the network-bias towards object shapes. We present a new training schema that increases this shape bias. Our basic idea is to alpha-blend a portion of the RGB training images with faked images, where each class-label is given a fixed, randomly chosen color that is not likely to appear in real imagery. This forces the network to rely more strongly on shape cues instead of texture cues. We call this data augmentation technique "Painting-by-Numbers", and we provide extensive experimental evaluation and propose a method to validate such shape-based techniques.

Contents

1

Introduction

1.1 Overview

Motivation. The research in computer vision started more than 50 years ago [56]. In the beginning, researchers pursued the goal of extracting geometrical information of two-dimensional blocks [98]. The goal of understanding scenes from real images followed in subsequent years in which low-level vision algorithms extracted image features such as edges and corners [82]. Nowadays, half a century later, remarkable progress has been made in computer vision. The diversity of computer vision tasks is immense, and for most, the state-of-the-art is set by deep convolutional neural networks.

Object recognition is a field of computer vision that aims to localize and detect objects in an image. In the simplest case, the visual recognition algorithm has to recognize one central object of an image contains *e.g.* an animal. This task is referred to as full-image classification [43], and we refer to the semantic category the object belongs to as a class. In practice, it is often desired to localize the object in an image by framing it with a bounding box, *i.e.*, a 2D rectangle, or 3D cuboid. Whereas this exercise is rather trivial for humans, it is extraordinarily challenging for machines. The task complexity increases with scene complexity. When images contain not just exclusively a single object but many, more sophisticated algorithms are required to solve the task at hand, represented by the task of object detection. Another widely studied computer vision field is image segmentation, which aims to extract the semantic content of an image. Concretely, image segmentation pursues the goal of assigning a class label to each pixel, for example, to differentiate between the foreground and the background of an image [99]. In this thesis, we centrally focus on one of the most widespread image segmentation variants, which is semantic segmentation. Semantic segmentation considers multiple instances of a class (*e.g.*, a single person of a group of persons) as a related collective and does not differentiate them.

The range of applications of image segmentation is diverse. In medical diagnostics, image segmentation can assist specialists in analyzing magnetic resonance and computed tomography scans [107]. Image segmentation can detect plant leaf diseases [85], and classify and differentiate crop [78] in agronomy. Semantic segmentation is also applied to aerial images, segmenting them into roads, buildings, or vegetation, which is essential for navigation applications or for recognizing natural hazards such as floods [96]. Autonomous driving is probably one of the most prominent candidates for computer vision applications. Segmenting the vehicle's environment into cars and traffic lights and further understanding the semantic context is of great importance, particularly for "vulnerable" classes like pedestrians or cyclists, for that the autonomous vehicle's software must pay special attention.

In the last decade, the de-facto state-of-the-art for these scene understanding and parsing tasks are set by neural networks and have surpassed human performance on particular categories of a visual recognition benchmark [103, 45]. A neural network learns a function that maps an input, here, the pixels of an image, to an output space such as class probabilities [71]. The emergence of fully-convolutional neural networks [79] started the deep learning-based development of semantic segmentation networks. Unlike convolutional networks for image classification, this type of network applies almost exclusively convolution operations, enabling to predict dense segmentation maps. The hundred of millions of learnable weights that can make up a neural network can be trained end-to-end. Neural networks are due to their success applied to steadily more academic and industrial disciplines.

Challenges. Deep networks excel on image data distributions they have been trained on but are known to struggle for data having a different distribution, for example, training a neural network on images captured during the daytime and deploying the trained neural network during nighttime. In the real world, images can be degraded in plenty of ways that, in turn, alter the image data distribution and might cause a network to predict wrongly. Concretely, environmental influences such as adverse weather conditions, *e.g.*, snowfall, rain, flare light, or fog, are naturally in severity and occurrence very diverse and further depend on the climatic region. Such environmental influences directly affect the optical and electrical properties of a camera and thus the image formation process. Additionally, fast-moving objects or camera motion causes image blur and the degree of smearing increases with decreasing scene illumination. Such external and internal influences affect and corrupt the image quality. A challenge in computer vision is the fact that these sources of image corruptions cannot be fully suppressed by sensing technology. Therefore, it is essential for safety-critical applications that neural networks have a certain degree of robustness against such unavoidable and inherently present image corruptions.

1.2 Contributions

The term "robustness" refers in this thesis generally to measuring the performance of a deep convolutional neural network with respect to images affected by common real-world image corruptions, as mentioned previously, such as fog, blur, or noise. In this thesis, we provide an extensive experimental evaluation with respect to the following two experimental setups. The network training is either conducted on i) the original (*i.e.*, non-corrupted) training images or ii) on both, *i.e.*, non-corrupted, and corrupted training data. The first case allows us to learn about the robustness properties of different neural network architectures. The authors in [37] show that a GoogLeNet [108] is less robust to image noise than ResNet-152 [47]. Is the former architecture less robust to noise due to the shallower architecture, nonexistent residual connections, or other network-specific architectural design choices? It is essential to understand architectural robustness properties if it is desired to design robust neural network architectures, in particular, because it may be hard actually to identify all possible scenarios in the application. The second case enables us to learn about the generalization capabilities of neural networks. A particular aspect discussed in detail in a later chapter of this thesis is to understand if and how a neural network generalizes to diverse types of image corruptions when trained on a particular one. Concretely, building upon the previous findings, we address the task of robustness enhancement by proposing a training strategy that replaces the original, non-corrupted training data with fake images, modifying the network's internal decision-making for the benefit of higher robustness against real-world image corruptions.

1.3 List of Articles the Thesis Builds Upon

The remaining chapters of this thesis build upon the following three publications.

1. **Benchmarking the Robustness of Semantic Segmentation Models**
   Christoph Kamann, Carsten Rother
   Computer Vision and Pattern Recognition (CVPR) 2020

2. **Increasing the Robustness of Semantic Segmentation Models with Painting-by-Numbers**
   Christoph Kamann, Carsten Rother
   European Conference on Computer Vision (ECCV) 2020

3. **Benchmarking the Robustness of Semantic Segmentation Models with Respect to Common Corruptions**
   Christoph Kamann, Carsten Rother
   International Journal of Computer Vision (IJCV) 2020

1.4 Outline

The central part of this thesis consists of the i) understanding and ii) increasing the robustness of deep convolutional neural networks against real-world image corruptions.

Chapter 2 – Background We provide the necessary technical background of topics that are covered in this thesis. This chapter begins with the image formation process presentation, describing how photons fall onto an image sensor and are digitally converted to an image. It is followed by the theoretical background of deep learning and image segmentation using deep convolutional neural networks.

Chapter 3 – Benchmarking the Robustness of Convolutional Networks We demonstrate an extensive investigation of the robustness capability of both semantic segmentation architectures and respective architectural properties against real-world image corruptions, and we introduce model design rules for robust segmentation. Each network is evaluated on approximately 400.000 corrupted images, including the following realistically modeled image corruptions: image sensor noise, geometric distortion, and camera optics blur.

Chapter 4 –Increasing the Robustness of Convolutional Networks We introduce an approach to generically increase the robustness of semantic segmentation models based on modifying the network's internal decision-making by trading-off the network's biases between object texture and object shape. We demonstrate a method for validating and measuring the trade-off between these network biases.

Chapter 5 – Conclusions and Outlook We finalize the thesis with a conclusion and discuss the remaining challenges.

2

Background

In this chapter, we cover the principles behind the image formation process and the background of deep learning. We then present the computer vision task of image segmentation, a fundamental task of this thesis, and address several semantic segmentation algorithms based on deep learning.

2.1 Image Formation Process

In this section, we cover the principles behind the image formation based on the pinhole camera model. As the pinhole camera model is a very simplistic camera model, we expand it by camera optics and an image sensor covering a description of a conventional cameras' image formation process. The work in [110, 44] are the main references for this section. When no other references are explicitly cited, we kindly refer to these works for more details.

2.1.1 Pinhole Camera Model

A camera maps a 3D point in space to a 2D point on the image plane. In the following, we discuss the simplest camera model, *i.e.*, the pinhole camera model for image formation. A point of an object's surface in the real world reflects light rays from a light source spatially. In more detail, a pinhole camera projects the 2D image onto an image film, and the image is oriented upside-down relative to the 3D scene. A barrier with an infinitesimally small aperture (pinhole) limits the number of light rays of the scene falling onto the film.

Figure 2.1 depicts a more formal representation of the pinhole camera model. Under the center of projection (or, optical center) O, a 3D point in space $P = (x, y, z)$ (where $P \in \mathbb{R}^3$, spanned by the Euclidean coordinate system $\hat{x}, \hat{y}, \hat{z}$, which is discussed later), is projected to the point $p = (x', y')$ (where $p \in \mathbb{R}^2$) on the image plane. Under consideration of the focal

length f, the mapping of a 3D point in space to a 2D point in the image plane is given by

$$(x', y')^T = \frac{f}{z}(x, y), \tag{2.1}$$

assuming that the image plane is in front of the optical center O.

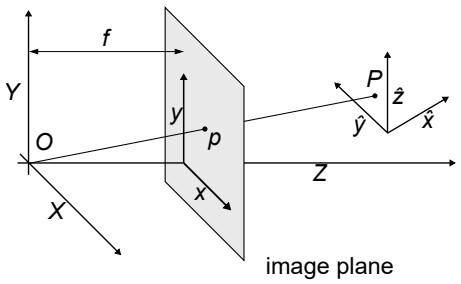


Figure 2.1: Illustration of the pinhole camera model; reproduced from [44].

Homogeneous coordinates are preferably used in projective geometry. A point $p \in \mathbb{R}^2$ that is expressed in homogeneous coordinates contains an extra coordinate, *e.g.*, $p = (x, y, \mathbf{1})$. When homogeneous coordinates are used, points are represented as equivalence classes, where two or more points are equivalent when they differ by common linear factors. Thus, a point with the homogeneous coordinates $(2, 2, 1)$ is equivalent to the point with homogeneous coordinates $(4, 4, 2)$, because both points represent the euclidean coordinate pair $(2, 2)$. The advantage of homogeneous coordinates is the possibility of representing points at infinity. The extra coordinate of a point in infinity is set to equals zero, *i.e.*, $p_{inf} = (x, y, 0)$.

Using homogeneous coordinates enables us to re-formulate eq. 2.1 as a general matrix-vector multiplication

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{camera matrix C}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{2.2}$$

where the camera matrix C is a 3×4 matrix, describing the projection from 3D to 2D. Incorporating the two constraints that i) the origin of coordinates in the image plane is not necessarily at the intersection of the image plane with the optical axis, the camera matrix C can be extended by the origin coordinates o_x and o_y, which is referred to as camera calibration matrix K. Moreover, ii), that the camera coordinate system is differently oriented than the euclidean coordinate system containing a point in space P (as illustrated in Figure 2.1), where P lies in the coordinate system $\hat{x}, \hat{y}, \hat{z}$. A euclidean rotation and

transformation are required to align both coordinates systems. The general mapping of a pinhole camera described by the camera matrix C is

$$C = K \cdot [R|t] =$$
$$\begin{pmatrix} f & 0 & o_x & 0 \\ 0 & f & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
$$\begin{pmatrix} \cos\alpha \cdot \cos\beta & \cos\alpha \cdot \sin\beta + \sin\gamma - \sin\alpha \cdot \cos\gamma & \cos\alpha \cdot \sin\beta \cdot \cos\gamma + \sin\alpha \sin\gamma & t_1 \\ \sin\alpha \cdot \cos\beta & \sin\alpha \cdot \sin\beta + \sin\gamma + \cos\alpha \cdot \cos\gamma & \sin\alpha \cdot \sin\beta \cdot \cos\gamma - \cos\alpha \sin\gamma & t_2 \\ -\sin\beta & \cos\beta \cdot \sin\gamma & \cos\beta \cdot \cos\gamma & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

where f, o_x, o_y are referred to as the internal camera parameters, R is the general 3D rotation matrix, α, β, γ are the spatial angles, and t is the translation vector [73].

Using the camera matrix C from eq. 2.3, we can now adequately describe the projective transformation of a point in space to a (2D) image in homogeneous coordinates. Images captured by a pinhole camera are degraded in several aspects. For example, whereas a large aperture results in blurred images, a small aperture limits the number of light rays, creating dark images. To counter this, we extend the pinhole camera model by camera optics in the next section.

2.1.2 Camera Optics

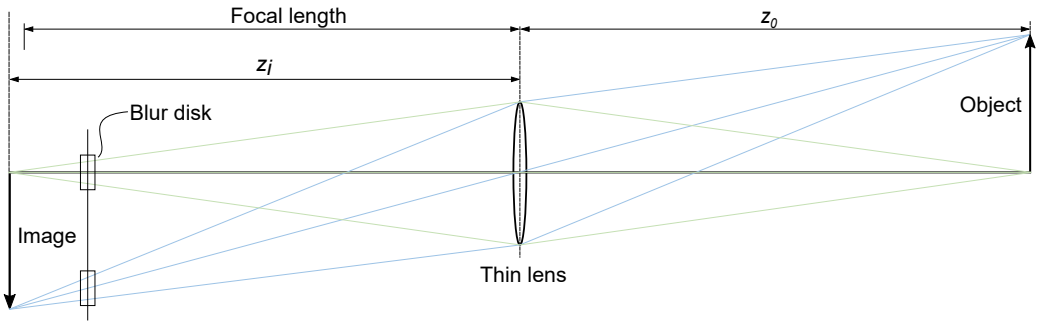


Figure 2.2: Optical path using camera optics; reproduced from [110].

Figure 2.2 illustrates the optical path when an object is projected onto an image plane using a thin lens between the object and image plane as the camera optics.[1] A point on the surface of an object reflects light spatially. The light ray passing the lens's spatial center

[1] Most optical aberrations are ignored for the thin lens approximation.

is not diffracted; light rays passing the lens at different positions are diffracted, hitting the image plane (or the image sensor) at the same spot as the center ray. The relation of the distance between object–lens and lens–projected object is defined by the lens law

$$\frac{1}{z_0} + \frac{1}{z_i} = \frac{1}{f}. \tag{2.4}$$

where f is the focal length of the lens. Moving the image plane (or the image sensor) towards the lens leads to an out of focus image, whose severity is measured by blur disk, or circle of confusion, since the light rays do not punctually hit the image sensor but are spatially distributed.

The camera optics enables us to accurately project a 3D scene through diffracting the reflected light on the image plane, without the need to limit the size of the aperture of a pinhole camera as described previously. However, real lenses are not infinitely thin, and, further, ray diffraction is not ideal, *e.g.*, due to manufacturing tolerances. Moreover, so far, we ignored that light consists of a spectrum of waves and described light solely with geometrical optics, which indicates that light propagates as rays. The diffraction properties of a lens are wavelength-dependent resulting in other out-of-focus scenarios. In the following, we discuss such unwanted optic-induced influences of image formation, which impacts are examined in the next chapter for modern image processing algorithms.

Optical Aberrations

Geometric Distortions. Geometric distortions, such as radial distortions, cause straight lines in the real-world to be curved in an image. Whereas this effect can be desired as an artistic aspect in photography when wide-angle lenses, such as fish-eye lenses, are used, they are mandatory in other applications, such as for driving assistance systems, where wide-angle lenses capture a large Field-of-View (FoV).[2] Barrel distortion and pincushion distortion are the two underlying distortions of radial distortion. With respect to barrel distortion, an object of scene mapped onto a 2D image is moved away from the image center, leading to a compression (enlargement) of the object's size the further (nearer) it is located at the image edge. With respect to pincushion distortion, image coordinates are moved towards the image center. Since geometric distortion may induce image artifacts, it is generally desired to compensate them, for example, by radial distortion models using polynomials

$$\begin{aligned} x_u &= x_c + L(r)(x_d - x_c) \\ y_u &= x_c + L(r)(y_d - y_c), \end{aligned} \tag{2.5}$$

where (x_c, y_c) is the distortion center, (x_u, y_u) are the undistorted (*i.e.*, corrected) coordinates, and (x_d, y_d) are the distorted (*i.e.*, measured) coordinates. The function $L(r)$ may

[2]The FoV depends on the focal length of a lens system and the image sensor's size. Since large image sensors are expensive, it is preferred to utilize appropriate lens systems to achieve the desired FoV.

be $L(r) = 1 + \kappa_1 r_d^2 + \kappa_2 r_d^4$ and the radius r_d is $r_d^2 = (x_d - x_c)^2 + (y_d - y_c)^2$, where κ_1, κ_2, x_c, y_c are referred to as the radial distortion parameters.[3]

It is important to note that an optical system's geometric distortion parameters are affected by environmental influences, alter over time, and vary from initial calibration stages. Consequently, radial distortion may not be fully neutralized. Further, when the image is warped for un-distorting the image, the warping process itself may introduce re-sampling artifacts (*e.g.*, aliasing effects), which degrades the informational content of an image. Therefore, one might prefer to use the geometrically distorted (*i.e.*, the original) image for tasks such as feature detection [44, p. 192f].

Spherical aberration Figure 2.2 illustrates the diffraction of light rays for a thin lens, *i.e.*, the rays focus on the focal point. Real lenses are finitely thick and have curved surfaces, causing light rays to be typically more strongly diffracted for the rays that are further distanced from the optical axis.

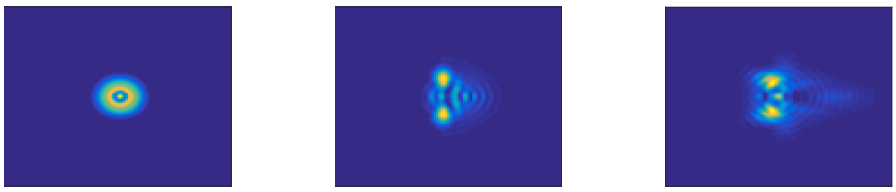
Chromatic aberration All materials have a refraction index, which indicates how fast light travels through a particular material. The refractive index determines the focal point of a lens. The refraction index of a material is always wavelength-dependent, such as for glass, where the index decreases with increasing wavelength. Hence, concerning camera optics, this wavelength-dependency affects the focus precision (*i.e.*, wavelength-dependent blur, referred to as longitudinal chromatic aberration) and wavelength-dependent magnification (referred to as transverse chromatic aberration). Whereas the latter type of chromatic aberration can be modeled and corrected through geometric distortion models, longitudinal chromatic aberration is due to the loss of information harder to correct.

Further optical aberrations. Other optical aberrations include "defocus", which results in image blur due to the image sensor being out of focus. In astrology, "coma" is an aberration that causes the light of an object which is not aligned with the optical axis are, similar to spherical aberration, reflected in different positions. "Astigmatism" causes image blur when different planes of a lens have a different focus, and "petzval field curvature" is an aberration due to the mismatch of shapes of a flat image plane and curved lenses of camera optics.

Point-Spread-Function. As discussed previously, optical aberrations occur in real lenses of any camera optics and degrade the image quality in terms of blur and color artifacts. A point spread function is the image of an ideal point light source captured by an imager's pixel sensor. Ideally, without aberrations, the point light source would be optimally mapped to a single pixel. Optical aberrations increase the point light source's spatial energy distribution, and hence the number of pixels it is mapped to [110, p. 78]. The PSF depends on the image sensor's spatial position, the distance between the lens and the object, and the wavelength. Figure 2.3 a) shows an example of a punctually centered real-world PSF caused

[3] For more complex lenses or fisheye lenses different models for correcting the distortion are more suitable.

by spherical aberrations. The energy distribution of light from a higher angle of incidences is significantly more distributed, and Figure 2.3 c) illustrates, among others, coma. More formally, the degradation caused by these aberrations can be described by a convolution with the PSF as the convolution kernel. When the PSF is known, it is hence possible to undo the aberrations mentioned here. In the next chapter of this thesis, we evaluate the effect of the image blur caused by real-world point-spread-functions with respect to modern image processing algorithms.



(a) Angle of incidence 0° (b) Angle of incidence 15° (c) Angle of incidence 30°

Figure 2.3: Several PSFs of a camera optics for varying angles of incidences. The PSF is more spatially distributed for higher angles, resulting in a more severe image blur.

2.1.3 Imaging Pipeline

In this section, we cover the principles of sensing technology in the imaging pipeline (see Figure 2.4). We first present image sensors used to capture impinging photons of light and convert them to digital color values. We also present common techniques of HDR imaging and possible sources of image noise. So far, we focused on the geometrical description *how* a point in space is mapped onto the image plane described by projective geometry using camera optics. Cameras used in the automotive section, in smartphones or digital cameras, are equipped with an image sensor accompanied by electrical signal processing.

CCD and CMOS sensor. An image sensor transforms light (refracted by the camera optics), falling onto the sensor into digital intensity values through counting the photons hitting the sensor in a particular time interval. There are two widespread types of image sensors. Firstly, the charge-coupled device (CCD) sensor. The CCD sensor absorbs photons that hit the sensor's active area, creating electrons whose energy is stored in an array of storage cells, which function as a capacitor. The stored charge is then sequentially moved to an output amplifier, transforming this charge into electrical voltage. The number of photons hitting the image sensor sets the electrical voltage's magnitude, which correlates with the light irradiance. In several post-processing steps, the final R, G, B values are computed. Secondly, the complementary metal-oxide on silicon (CMOS) sensor. Every CMOS' sensor's pixel is a photodiode containing an amplifier-transistor circuit. The photo-current discharges the photodiode's initial capacity when light falls on the sensor, decreasing the photodiode's

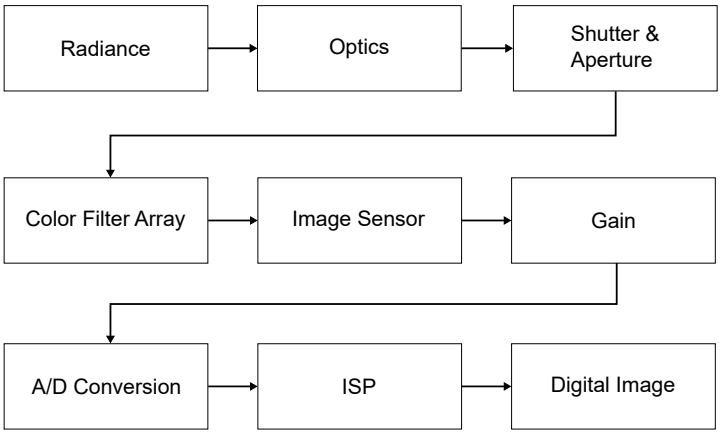


Figure 2.4: Overview of main components of the imaging pipeline. The camera optics refracts incidental light. The shutter opens during the exposure time, allowing light to travel through the aperture, where the color filter array is located in front of the image sensor. The gain amplifies the image sensor's analogous signal, which is converted to a digital signal. The image signal processing unit (ISP) eventually applies a series of post-processing steps to create the final image; reproduced from [110].

electrical voltage. Like the CCD sensor, the magnitude of electrical voltage correlates with irradiance, enabling analogous-digital post-processing to create the digital color values. The advantages of a CMOS sensor are, in general, both lower power consumption and price, on-chip functionality, high-level of integration (miniaturization), and high-speed imaging. In terms of sensitivity and noise, the created image quality used to be worse than for a CCD sensor before the 1990s. However, the CMOS sensor's image quality has been improved steadily [7].

Exposure time. The exposure time or shutter speed describes the time interval in which the aperture is opened and enabling light to hit the image sensor. Regarding Figure 2.2, the light (irradiance) E hitting an pixel sensor is given by

$$E = L\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha \tag{2.6}$$

where d is the height of the lens, L is the scene radiance, and α the off-axis angle. While arbitrary exposure times are possible, it is typically chosen inverse proportional to the sensor plane's average irradiance. This corresponds to roughly $\frac{1}{100}$ second for dark scenes to $\frac{1}{1000}$ second for bright scenes. However, other aspects may influence the optimal exposure settings, *e.g.*, sensor properties, light filters, and the usage of advanced HDR imaging techniques. We

cover in section 2.1.4 a high dynamic range imaging technique based on combining the images taken in parallel during multiple exposures.

Gain/ISO. The gain is responsible for amplifying the image sensor's analogous signal, which can be controlled through a camera's ISO setting. The resulting amplified signal is then converted to a digital signal.

Color filter array and demosaicing. The color filter array, typically arranged in a Bayer pattern, is a two-dimensional array (ordered in a checkerboard scheme) that functions as a filter for incoming light placed after the aperture and before the image sensor (see Figure 2.5). Each cell of the Bayer pattern is a color filter for the red, green, or blue

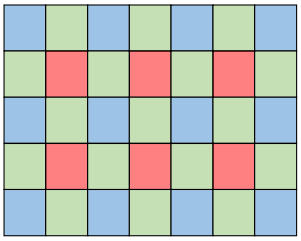


Figure 2.5: Illustration of the Bayer pattern, a two-dimensional array consisting of color filters ordered in a checkerboard scheme located right before the image sensor. Each cell corresponds to a pixel on the image sensor. A color value is created by interpolating the measurements of neighboring pixels; reproduced from [110].

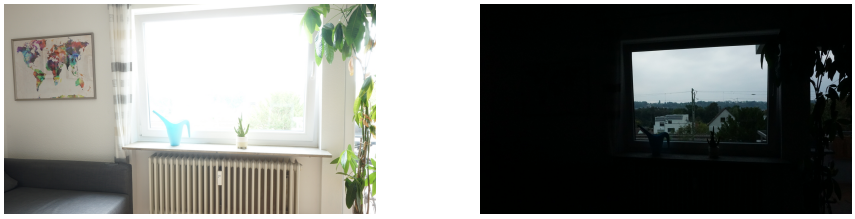
spectrum. A color value (for example, the red one) is determined by integrating light using the spectral response function of the red color sensor, given by

$$R = \int L(\lambda) S_R(\lambda) d\lambda, \tag{2.7}$$

where $L(\lambda)$ is the spectrum at a pixel, and S_R is the spectral sensitivity of the red sensor, describing the sensor's responsitivity for this color.

A blue color filter covers the top-left pixel of the image sensor in Figure 2.5. The questions arise about how the sensed color values for and around a pixel can be reliably interpolated to a final color for that particular pixel. This interpolation scheme is referred to as demosaicing. Interpolation algorithms such as a bilinear interpolation can be used to infer the color at homogeneous regions; however, they cause demosaicing artifacts (such as zippering) at object edges. Processing techniques that rely more on the pixel's gradient of the more densely available green channel reduce demosaicing artifacts.

2.1.4 HDR Imaging



(a) Saturation (b) Underexposure

Figure 2.6: Scenery images showing that the naturally occurring dynamic range of scene radiance is considerably high. A standard capturing technique can only capture either the indoor of the scene **(left)** or the outdoor scene **(right)**.

Figure 2.6 shows that the naturally occurring range of scene radiance is considerably higher than can be captured within the image sensor's limited range. This range (referred to as dynamic range) is expressed by integer exposure values (EVs). An increase in the dynamic range by 1 EV corresponds to doubling the scene radiance. High dynamic range (HDR) imaging allows capturing images of a greater dynamic range than with a single image, often referred to as low dynamic range (LDR) image. While there exist image sensors dedicated to HDR imaging, we discuss in the following an HDR-imaging technique referred to as "bracketing". The principle of bracketing is to capturing a scene for several exposure values and combining each of these low dynamic range images to a joint composite. Szeliski [110] organizes the bracketing process in three steps. 1) It is required to estimate the radiometric response function. The radiometric response function maps the light exposure/irradiance (see eq 2.6) to a digital RGB value. 2) Appropriate pixels from the low dynamic range images are "selected" or "blended" to estimate the HDR image, which must 3) be tone mapped to a regularly displayable/viewable color gamut (*e.g.*, to the standard RGB color gamut with range $[0, 255]$ per color channel).

Figure 2.7 shows in the left a radiometric response function [24, 86]. Unfortunately, in general, neither is the radiometric response function nor the exposure E_i for a pixel z_i known. However, it is possible to reconstruct the radiometric response function, as shown on the right, through determining E_i for each pixel z_i for several exposure times. One approach is therefore to estimate E_i simultaneously with the radiometric response function f for a measured pixel's intensity value z_{ij} and exposure time t_j as

$$z_{ij} = f(E_i; t_j). \tag{2.8}$$

Eq. 2.8 can be rearranged to

$$g(z_{ij}) = \log f^{-1}(z_{ij}) = \log E_i + \log t_j, \tag{2.9}$$

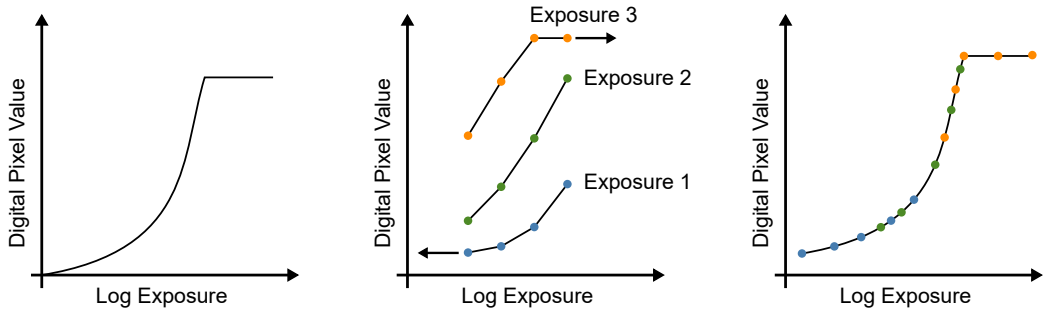


Figure 2.7: Calibration of the radiometric response function. **(left)** A typical but unknown radiometric response function for a color channel mapping incoming irradiance to a pixel's intensity value (*e.g.*, a value between 0 and 255). (middle) The individual exposures E for each pixel for several exposure times. **(right)** Through shifting and smoothing each exposure curve, the radiometric response function is approximated. Reproduced from [110].

where g is the function to be approximated, that maps each pixel's intensity value to log irradiance.[4] The exposures E_i for each pixel z_i with corresponding response values g_k (*e.g.*, 256) are the unknown parameters to be approximated; the exposure time t_j is given.

We have an inverse response function $E = g(z)$ after the previous approximation, which estimates the pixel's exposure z. The second step's goal is to select the *best* pixels from the available brackets to create a composite radiance map. Due to inaccurate approximation or noise, an appropriate radiance value for non-saturated pixels at the transition of two exposure times is not trivial and is, for example, determined by a weighted sum of E of the inverse radiometric response curves g_k. In the third and last step, the radiance map (which is, for example, a 32-bit float per channel) must be converted into a regular, displayable image (*e.g.*, 8 bit per channel). This conversion is referred to as tone mapping. Whereas many tone mapping algorithms have been proposed, a simple method is gamma compression, that globally maps a radiance map by

$$v = ar^{\gamma}, \tag{2.10}$$

where a is a scalar ($a > 0$), the exponent γ ($0 < \gamma < 1$) maps an entry of the radiance map r to a value v, where $v = [0, 1]$. More complex tone mapping algorithms apply pre-processing such as edge filtering, decomposition in log luminance and chrominance images, and contrast reduction to increase the tone mapping quality.

So far, we discussed bracketing in terms of photography. In real-world applications, such as in the automotive context, the appointed cameras for video recording are operated in demanding conditions. Automotive video cameras are calibrated during manufacturing,

[4] The authors of this approach formulated a least square problem to approximate E_i and g_k [24].

which does not require estimating a radiometric response function algorithmically, and the tone mapping function is set. If an automotive camera has the requirement to capture 30 images per second and use up to four brackets to composite an HDR image, then the longest exposure time that can be used for bracketing is limited. The development of single exposure sensors is an active field of research and aims to mitigate such limitations due to application-dependent circumstances.

2.1.5 Imaging Noise

In the previous sections, we mostly ignored image noise and possible noise sources. Noise, in general, is any deterioration of a particular signal. In fact, most imaging pipeline components, as illustrated in Figure 2.4, are a source for noise in the final image. In the following, we discuss the most dominant types of noise occurring. We kindly refer to [59, 32, 113, 7, 11, 88, 48, 119] for a more detailed description of types of image noise.

Shot noise. The average number of photons N hitting a pixel of an image sensor in a given time interval (exposure time) t_e is given by

$$\lambda = \frac{N}{t_e}. \tag{2.11}$$

Due to light's random nature, the number of photons counted in identical subsequent time intervals is not constant. This uncertainty is referred to as shot or photon noise. A Poisson-distribution describes shot noise since averaging the number of photons N by λt_e is referred to as a Poisson process. The Poisson distribution f_p is given by

$$f_p = \exp(-\lambda t_e)\frac{(\lambda t_e)^n}{n!}, \quad n >= 0 \tag{2.12}$$

with $\sigma^2 = \mu = \lambda t_e$. Since the standard deviation equals the number of counted photons' square root, shot noise is significantly dominant when N is small, *i.e.*, for poorly illuminated scenes.

Dark current. Not every recorded electron is generated by absorbed photons of light falling onto the image sensor, but also due to thermic processes. The (unwanted) electric current flowing due to such electrons is referred to as dark current, leading to dark current noise or thermal noise. Dark current occurs in most semiconductor devices, such as the photodiodes making up CMOS sensors or the silicon structure of a CCD sensor. Dark current f_{dc} is typically modeled by additive Gaussian noise for each pixel, *i.e.*,

$$f_{dc} = \frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right), \tag{2.13}$$

where σ, μ as standard deviation and mean of the Gaussian distribution.

Quantization noise. Quantization noise occurs when a continuous signal is discretized. In the imaging pipeline illustrated in Figure 2.4, quantization noise happens at the analog

to digital conversion and at the compression stage, *i.e.*, before the final image is created. Quantization noise is approximated by a uniform distribution.

Read noise. Read noise occurs during the readout of electronic and semiconductor devices and usually includes several types of noise such as reset noise, white noise, or flicker noise. The output amplifiers mostly determine the read noise. In a CCD sensor, such an amplifier operates on the sequentially moved charge (see section 2.1.3). An amplifier is part of each pixel in a CMOS sensor. Read noise is modeled by a Gaussian distribution.

Fixed Pattern Noise (FPN). Due to fluctuations in the manufacturing process of an image sensor (charge wells in a CCD sensor, or transistors in a CMOS sensor), environmental influences (*e.g.*, ambient temperature, humidity), and also wear and tear of the device, each pixel sensor responses slightly differently to an identical input signal. Applying a uniform signal to an image sensor results in an output image with a non-homogenous pattern, referred to as Fixed-Pattern Noise (FPN). However, FPN can be relatively reliable eliminated through appropriate signal processing.

Impulse noise. Impulse noise, or salt and pepper noise, is a type of noise that causes some pixels in the image to be noisy. Crossover distortions can disturb the signal by flipping bits when an image is transmitted through a digital process. Since the crossover probability is relatively low for most bits, only a few pixels are affected. However, the degree of noise for affected pixels is considerably high.

Speckle noise. Speckle is a type of noise that is produced by mutual interference of coherent light. For example, when a light wave hits another medium's interface, a portion of the wavefront is reflected. A medium's surface is microscopically viewed, rough, causing the reflected light to scatter, which leads to mutual destructive and constructive interference. Due to the interferences, the resulting signal hitting a pixel of the image sensor can underly substantial variations in amplitude and phase, which is referred to as "speckle". A typical case is that a camera's optics cannot resolve a surface's roughness, but the roughness is considerable when viewed on the wavelength scale. In this case, speckle noise is modeled as a multiplicative noise model where bright areas' pixels are noisier than pixels of dark areas. We refer to [11] for more details.

Image Noise in HDR imaging. Figure 2.8 schematically illustrates the signal to noise ratio (SNR) as a function of log irradiance for an HDR image capturing using three exposure times [8]. Each SNR-exposure segment's curved shape is due to the dominating shot noise since its variance is proportional to the square root of the mean. The pixels of the first exposure (the longest exposure due to low irradiance) saturates at a certain point. At this point, pixels of the second exposure is taken to create an HDR image through bracketing. The transition from second exposure to the third behaves similarly. Between each transition of exposures segments, an unavoidable SNR drop or SNR discontinuity is observed. However, the drop can be diminished when the ratio of longest to shortest exposure is decreased or

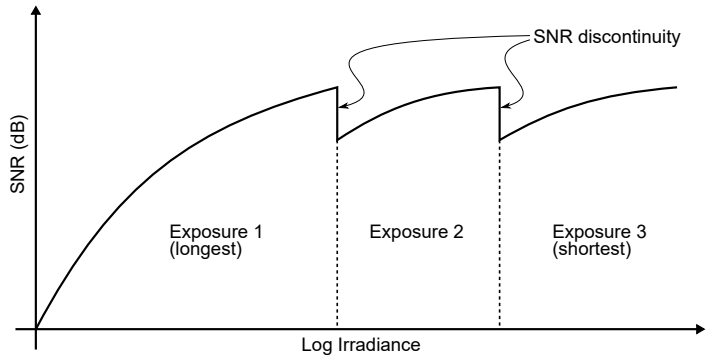


Figure 2.8: Signal to noise ratio (SNR) as a function of log irradiance for a three-shot exposure bracketing. The pixels of each exposure time saturate for an absolute irradiance, resulting in an SNR drop/discontinuity at the transition to the subsequent bracket. These kinds of SNR drops cause a special kind of image noise characteristic that is not present in regular, single-exposure low dynamic range images; reproduced from [8].

when more exposure times are used. These SNR discontinuities and other influences may cause special artifacts in the final HDR image [8, 2, 40, 75, 6]. Please see Appendix A.1 for a multi-exposure noise model.

2.2 Deep Learning

In the last years, deep artificial neural networks have set the state-of-the-art on many computer vision tasks. In the following, we address the basics of neural networks, regularization, optimization, and convolutional neural networks. The work in [39, 66] are the main references for this section. When no other references are explicitly cited, we kindly refer to these works for more details.

2.2.1 Feedforward Neural Networks

Artificial neural networks are used in a diverse set of applications. For the task of supervised full-image classification, a neural network represents a function $\hat{f}$ that maps an input X to an output Y. Assuming the task of a neural network is to classify an image x of a set of images X as either *dog*, *plane*, or *boat*, the neural network assigns then the image an integer class label y, representing one of these categories, *i.e.*, $Y = \{y_{dog}, y_{plane}, y_{boat}\} = \{0, 1, 2\}$. The image-label pairs represent a dataset consisting of n examples $\{(x_0, y_0), ..., (x_n, y_n)\}$. The mapping of $\hat{f}$ between X and Y is controlled by a set of learnable parameters or weights that are summarized in θ. These parameters are determined through the training process.

In the remaining chapters, we use the terms "parameters" and "weights" interchangeably. The neural network learns hence the mapping $y = \hat{f}(\theta, x)$, where $x \in X$, $y \in Y$.

It is commonly assumed that the samples of a dataset are independent and identically distributed (i.i.d assumption), meaning that the samples of a dataset are mutually independent and that train and test sets are generated by the identical probability distribution p_{data}. However, in practice, the true data-generating process p_{data} is unknown, and, instead, it is attempted to find a distribution p_{model}, representing a reasonable approximation using the available training examples. Formally, the neural network's objective is to find a function $\hat{f}$ which minimizes the expected loss on the training data that were sampled from the data-generating process $\hat{p}_{data}$

$$\hat{f} = \arg\min \mathbb{E}_{x,y\sim\hat{p}_{data}} L(y, f(x;\, \boldsymbol{W,b})), \tag{2.14}$$

where $f(x;\, \boldsymbol{W,b})$ is the score function of the neural network, $\boldsymbol{W,b}$ are the learnable parameters consisting of weight matrices $\boldsymbol{W} = \{W_1, ..., W_{l+1}\}$ and bias vectors $\boldsymbol{b} = \{b_1, ..., b_{l+1}\}$ for each of the l hidden layers h, and L is the loss function. The score function f outputs a score for each class. A score function with one hidden layer can be written as

$$f(x; W_1, b_1, W_2, b_2) = W_2^T max(0, W_1^T x + b_1) + b_2, \tag{2.15}$$

where W_1, b_1 are the parameters of the hidden layer h, and W_2, b_2 are the parameters of the second layer (in this case the output layer). W_1 is a weight matrix of size $D \times H$, where H is the number of hidden units (or neurons) of the hidden layer h and D is the dimension of training data x (if the input data is an image, x is a column vector containing the pixels), b_1 is a bias vector of size H, W_2 is the $C \times H$ weight matrix of the output layer for the C classes, and b_2 is the respective bias vector of size C.

The max operator including its argument makes up the hidden layer h which is, in the linear case, $h = g(W_1^T x + b_1)$.[5] The activation function g is a fundamental component of neural networks since it incorporates the computational non-linearity, that enables neural networks to learn complex non-linear functions. Omitting the activation function is mathematically equivalent to rearrange W_1, W_2 as a single matrix, *i.e.*, the equation is linear. In this example (eq. 2.15), the commonly chosen rectified linear unit, or ReLU [87], defined by $g(z) = max(0, z)$ is used as activation function.

A loss function evaluates the quality of the score value created by equation 2.15. The maximum likelihood estimation is used in many cases. It applies the cross-entropy (*i.e.*, the negative log-likelihood) as loss function, coupled with an appropriate output unit of

[5]The term "hidden layer" arises from the situation that the parameters of a hidden layer are not directly accessible from the input data. Solely the input data and the desired output data are given, and it is the neural network's task to arrange the weights within the layers adequately to solve the task at hand.

the neural network. A reasonable choice for an output unit for the classification example discussed here is the softmax unit and is

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \tag{2.16}$$

where z_i is the predicted score of the neural network for correct class i. Since the scores are commonly interpreted as logits, the softmax function's output can be treated as a probability vector of length three, which entries sum to one. Each value represents the certainty of the neural network's estimation for the particular class to be the correct one. Using the softmax output unit along with maximum likelihood leads to the following loss function L

$$L_i = -log\left(\frac{\exp(z_i)}{\sum_j \exp(z_j)}\right), \tag{2.17}$$

resulting, in turn, from minimizing the KL divergence between the observable data-generating process $\hat{p}_{data}$ and p_{model}.[6] When a neural network is optimized using a softmax output unit, the goal is hence to maximize the probability p_i of the correct class. The softmax output unit is exclusively used in the convolutional neural network architectures covered in the later chapters of this thesis.

Up to this point, a neural network aims to minimize the following objective

$$\hat{f} = \arg\min\left[-\frac{1}{n}\sum_{i=1}^{n} log\left(\frac{\exp(W_2^T max(0, W_1^T x_i + b_1) + b_2)_{y_i}}{\sum_j \exp(W_2^T max(0, W_1^T x_i + b_1) + b_2)_j}\right)\right], \tag{2.18}$$

where y_i corresponds to the prediction of the correct class. This objective, however, can be unstable in case the learnable weights (*i.e.*, W_1, b_1, W_2, b_2) are considerably large. Neural networks need to be adequately regularized to avoid large weights.

2.2.2 Regularization

An essential aspect of machine learning is to develop models that do not only perform well on the data they were trained on but also on unseen data. Regularization aims to minimize the difference between the error on the training and the test set. Goodfellow et al. [39] define regularization as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error".

Commonly used regularization strategies add a parameter norm penalty Ω and a hyper-parameter α ($\alpha \in [0, \infty[$), controlling the magnitude of the regularization to the objective which limits, in turn, the capacity of a neural network. The capacity corresponds to the number of learnable parameters, or in other words, the number of neurons. The higher the neural network's capacity, the larger the number of functions in the parameter function family F and thus the risk that the model overfits the training data.

[6] In fact, minimizing the KL divergence between these distributions, represents to align p_{model} with $\hat{p}_{data}$.

Weight decay. The L^2 norm penalty, also known as weight decay, is broadly used in deep learning, and it incorporates the term $\Omega(W) = \frac{1}{2} \|W\|_2^2$, where W is a weight matrix of a neural network. Adding the weight decay to the objective (eq. 2.14) results in

$$\hat{f} = \arg\min \mathbb{E}_{x,y\sim\hat{p}_{data}} L(y, f(x; \mathbf{W}, \mathbf{b}) - \alpha\Omega(\mathbf{W})). \tag{2.19}$$

And the objective of the example discussed in this section using the L^2 regularization is

$$\hat{f} = \arg\min \left[-\frac{1}{n} \sum_{i=1}^{n} log\left(\frac{\exp(W_2^T max(0, W_1^T x + b_1) + b_2)_{y_i}}{\sum_j \exp(W_2^T max(0, W_1^T x + b_1) + b_2)} \right) - \alpha(\|W_1{}_2^2\| + \|W_2{}_2^2\|) \right]. \tag{2.20}$$

Data augmentation. Training data augmentation is another regularization technique that is commonly applied in deep learning-based computer vision. Augmenting the training data follows the goal to artificially increase the number of training images and hence the generalization capability of a neural network. There is a multitude of possible data augmentation techniques. Simple and computationally efficient forms of data augmentation are established since they can usually be applied online and significantly enhance generalization. Examples of such augmentation techniques are geometric transformations of the input pixels such as random cropping, scaling, zooming, blurring, and flipping. Other augmentation schemes corrupt images by manipulating the image content, for example, by randomly occluding image regions or injecting image noise.

Data augmentation is also applied to increase the number of images in a particular scenario, where the availability of training images is limited. For example, for real-world applications, such as autonomous driving, data augmentation techniques have been proposed that insert synthetically generated fog in images [105] or adapting a model which is trained on daytime images to nighttime [104]. We show in chapter 3 that convolutional neural networks for semantic image segmentation can generalize quite well to a realistic type of image noise though the training data has been augmented with simple image noise models.

The previously presented augmentation schemes aim to make the training set more diverse or insert domain-specific examples (*e.g.*, foggy images). In chapter 4, we discuss data augmentation techniques that augment images in a way that affects the internal decision-making of convolutional neural networks, that increase the generalization through enhancing the network robustness against real-world image corruptions [36]. For example, applying such training strategies increases the robustness against, *e.g.*, image noise, even though the network has not "seen" image noise during the network training. Unlike humans, (convolutional) networks are generally biased towards objects' texture than towards objects' shape. Since common image corruptions like adverse weather or image noise do primarily destroy image texture, the models are quite vulnerable to these image corruptions. However, the biases of a network towards texture can be decreased when the images are augmented in a

way that the network cannot rely on the textural content of an image. The network needs then to develop additional cues (or increasing already existing cues) to predict accurately. The question of why convolutional neural networks are biased towards object texture arises. A hypothesis is Occam's Razor: If the texture cue is sufficient to solve the task at hand, then there is no need for a network to learn more complex cues [36].

2.2.3 Optimization

We presented previously the objective of a 2-layer neural network for image classification, using the softmax function as the output unit. This section presents a brief overview of how an objective (eq. 2.19, 2.20) can be optimized, or in other words, how a neural network can be trained to solve the classification task.

The objective in eq. 2.20, minimizes the average error on the full training data set. Minimizing this objective is known as empirical risk minimization. A drawback of empirical risk minimization is its sensitivity to overfitting, meaning that the difference between the error on the training set and error on the test is high, and further is the computational cost expensive. Training datasets in computer vision can consist of several million examples. Computing the expectation on the full training set for each parameter update is extremely expensive.

In general, gradient-based optimization methods are used for optimizing a (convolutional) neural network, through calculating the gradient $\hat{g}$ of the loss function with respect to the parameters θ using the backpropagation algorithm (presented later)

$$\hat{g} = \frac{1}{n}\nabla_\theta \sum_i L(f(x_i;\theta), y_i). \tag{2.21}$$

Instead of calculating the gradient on the full training set, only a subset m (usually called a mini-batch) of the training data n is considered. A commonly utilized optimization algorithm for deep learning is the stochastic gradient descent [68, 97] (SGD, see Algorithm 1 [39]).

Algorithm 1: Stochastic Gradient Descent

Set hyperparameter: learning rate α

Initialize: parameters θ

while *loss not converged* **do**

  Draw a mini-batch of m examples of the training set X with respective labels Y

  Estimate gradient: $\hat{g} = \frac{1}{m}\nabla_\theta \sum_i L(f(x_i;\theta), y_i)$

  Perform update of parameters θ using gradient estimate: $\theta \leftarrow \theta - \alpha\hat{g}$

end

One may combine the standard SGD with algorithms such as the momentum algorithm [95], which allows for faster convergence. The momentum algorithm incorporates

previous gradients estimates for the current parameter update.[7] The step size of the parameter update is larger when gradients point in the same direction. It adds a velocity variable v to SGD (see Algorithm 2 [39]), and the hyperparameter β controls the influence of previous gradients. The momentum algorithm is mostly used as an optimization algorithm for the experiments presented in later chapters of this thesis.

Algorithm 2: Stochastic Gradient Descent with Momentum

Set hyperparameter: learning rate α, momentum parameter β

Initialize: parameters θ, velocity v

while *loss not converged* **do**

Draw a mini-batch of n examples of the training set X with respective labels Y

Estimate gradient: $\hat{g} = \frac{1}{m}\nabla_\theta \sum_i L(f(x_i;\theta), y_i)$

Velocity update: $v \leftarrow \beta v - \alpha g$

Perform update of parameters θ using velocity: $\theta \leftarrow \theta + v$

end

Algorithms with adaptive learning rates, such as AdaGrad [29], RMSProp [51], and Adam [69] are widely used.[8]

Backpropagation. In the previous, we described the principle of neural network optimization, and we showed that it is based on estimating the gradient of the loss function with respect to the network parameters θ using a mini-batch of m examples. As (convolutional) neural networks often consist of hundreds of millions of parameters, the analytical calculation of gradient through differential calculus is very complex, error-prone, and every change conducted on a neural network architecture requires a re-derivation. Instead, the gradient of the loss function with respect to the learnable parameters is calculated using the backpropagation algorithm.

For discussing backpropagation, it is helpful to describe a neural network as a computational graph (directed acyclic graph), where information "flows" from starting points to an ending point. In Figure 2.9, we show the graph of the 2-layer neural network discussed so far in this chapter, which calculates the total cost $J = L(y, f(x; \mathbf{W},\mathbf{b}) - cR(\mathbf{W}))$, where $f(x; \mathbf{W},\mathbf{b}) = W_2^T max(0, W_1^T x + b_1) + b_2$ and the L^2 regularization R. Considering the computational graph, the input X is firstly multiplied by weight matrix W_1. Adding the bias vector b_1 computes U_2. The ReLU activation function is then applied on U_2, resulting in the hidden layer H. The output layer computes the scores, using W_2 and b_2, which are the input to the softmax output unit (eq. 2.16), using the maximum likelihood principle (eq. 2.17). Since we use the L^2 regularization, the weight matrices W_1, W_2 are further squared, element-wise accumulated, and summed with L to compute the total cost J.

[7] In physics, the momentum of a particle is mass times velocity.

[8] The Adam algorithm is generally recommended in the early stages of setting up a neural network model, as flawed and unfavorable choices of hyperparameters have a less corrupting impact.

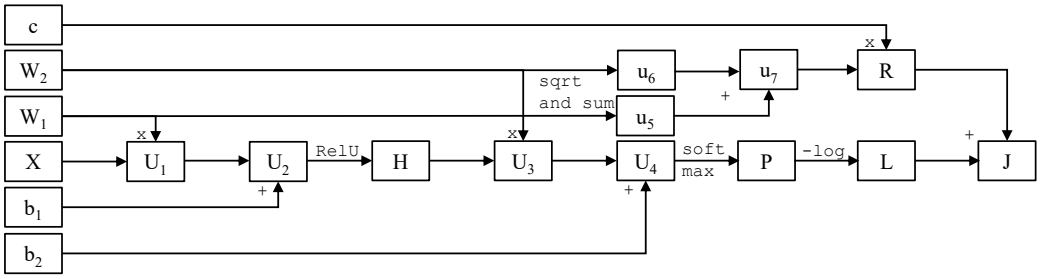


Figure 2.9: Computational graph of a 2-layer neural network with L^2 regularization.

The backpropagation algorithm [101] enables a safe way to calculate the required gradients through recursively applying the chain rule of calculus on intermediate variables throughout the network. In the scalar case, when, for example, the functions $y = g(x)$ and $z = f(g(x)) = f(y)$ are given, the chain rule allows to calculate $\frac{dz}{dx}$ as the multiplication of $\frac{dz}{dy}$ and $\frac{dy}{dx}$. In the context of a neural network, the gradient of the total (scalar) loss with respect to an intermediate variable (*e.g.*, U_3) is calculated by (applying the chain rule of calculus): Considering the backward path's direction, the gradient of the previous intermediate variable multiplied with the Jacobian of the operation, creating the current intermediate variable results in the desired gradient. We now exemplary apply the backpropagation algorithm through the 2-layer neural network, depicted as a computational graph in Figure 2.9.

We are ultimately interested in the gradients $\frac{\partial J}{\partial W_1}$, $\frac{\partial J}{\partial W_2}$, $\frac{\partial J}{\partial b_1}$, $\frac{\partial J}{\partial b_2}$. The first derivation which is calculated is the gradient for the scalar J, *i.e.*, $\frac{dJ}{dJ} = 1$. Then we continue the backward-path towards L. The derivative of the softmax output unit using the maximum likelihood principle (see Eq. 2.17) for an example i has the simple form $\frac{\partial L_i}{\partial z_{y_i}} = softmax(z_i)_{y_i} - 1$. The gradient $\frac{\partial J}{\partial U_4}$ can hence be computed by subtracting one from the correct class in each row of U_4 (multiplied by $\frac{dJ}{dJ} = 1$). In the next step, it is already possible to compute the gradient $\frac{\partial J}{\partial b_2}$ by multiplying $\frac{\partial J}{\partial U_4}$ and $\frac{\partial U_4}{\partial b_2}$, resulting in simply in a vector containing the sum of each column of $\frac{\partial J}{\partial U_4}$. The operation creating U_3 is a matrix multiplication HW_2. The gradient of the loss with respect to W_2 is computed by $\frac{\partial J}{\partial W_2} = \frac{\partial U_3}{\partial W_2}^T \frac{\partial J}{\partial U_4} = H^T \frac{\partial J}{\partial U_4}$. Regarding the gradients of the loss with respect to W_1 and b_1, we have to backpropagate through the hidden layer first. The chain rule allows to calculate $\frac{\partial J}{\partial H}$ by multiplying $\frac{\partial J}{\partial U_3}$ and $\frac{\partial U_3}{\partial H}^T$, resulting in $\frac{\partial J}{\partial H} = \frac{\partial J}{\partial U_3} W_2^T$. Traveling further, the next Jacobian to be calculated is $\frac{\partial J}{\partial U_2}$ by backpropagating through the ReLU operation. Its derivation sets every negative value of $\frac{\partial J}{\partial H}$ to zero. Analogously to b_2, $\frac{\partial J}{\partial b_1}$ is then simply a vector of the sum of each column of $\frac{\partial J}{\partial U_2}$. Finally, backpropagating into W_1 through the multiplication operation is

done by $\frac{\partial J}{\partial W_1} = \frac{\partial U_1}{\partial W_1}^T \frac{\partial J}{\partial U_1} = X^T \frac{\partial J}{\partial U_1}$, where $\frac{\partial J}{\partial U_1} = \frac{\partial J}{\partial U_2}\frac{\partial U_2}{\partial U_1} = \frac{\partial J}{\partial U_2} \cdot 1$.

L^2 regularization contributes also to the gradients $\frac{\partial J}{\partial W_1}$ and $\frac{\partial J}{\partial W_2}$, that is $\frac{\partial L^2}{\partial W} = 2cW$ for each weight matrix.

This example shall demonstrate that the backpropagation algorithm applies the chain rule of calculus recursively to calculate the Jacobians of nodes throughout a neural network in a generic way. It is important to note that the Jacobians of the neural network operations are very memory intensive. In general, it is not required to explicitly form the full Jacobians, as shown in the previous example for the multiplication nodes. In deep learning libraries, the backpropagation algorithm is efficiently implemented.

2.2.4 Convolutional Neural Networks

In computer vision, the input data being processed by artificial neural networks are images. Since images can consist of several million pixels, the weight matrices of hidden layers are very large. For example, a mini-batch consisting of 16 RGB images of size 512×512 pixels results in a design matrix X with more than 12 million entries. The weight matrix W of the first hidden layer processing X with 100 neurons would consist of roughly 78 million weights. The number of parameters rises quickly when neural networks with many hidden layers and higher capacity are used. However, images are grid-like ordered, which is a special kind of structure. Regular neural networks have been enhanced in a way that exploits the grid-like input data structure of images: Instead of applying regular matrix multiplication between the input data and a hidden layer, the convolution operation is applied, resulting in a convolutional (neural) network (CNNs) [74, 35]. The convolution operation allows reducing computational memory requirements drastically.

Convolution layer. The fundamental layer of a convolutional network is a convolutional layer. A filter kernel with small spatial dimensions, such as 3×3, is convolved with an image. The image region covered by a filter is referred to as the receptive field. The convolution operation moves the filter kernel effectively to each spatial location (pixel), computing the dot product between the filter kernel and the respective image region, as schematically illustrated in Figure 2.10 a. The mathematical definition of a convolution is not equivalent to the dot product between image region and filter kernel, but to the dot product between the image region and the flipped kernel. The convolution operation, as used in most deep learning libraries, is referred to as cross-correlation. However, since the filter kernels contain the weights that are eventually learned throughout the training, the convolutional network can learn either filter weights, which correspond to a flipped or a regular filter. The depth of a filter kernel must be equal to the depth of the input data (for example, the depth of the first convolution layer's filters is three for an RGB image). In this way, a filter kernel produces a two-dimensional output with a depth of one, which we refer to as an **activation map**. A

convolutional layer consists of many filters (we later present architectures with 2048 filters). The final convolution layer's output is the **activation volume**, which consists of the stacked activation maps created by the individual filters. Hence, the number of filters sets the depth dimension of the convolution layer's output and, in turn, the subsequent convolution layer's filter depth. The activation function (*e.g.*, ReLU) is applied to the activation volume. In deep learning libraries, properties such as stride (inserts spacing between the spatial locations being convolved with a filter) or padding can be set for each convolution layer. Stride 2 is often used to downsample the spatial dimension of the input by half.

Convolutional networks incorporate parameter sharing and sparse connectivity: The weights of a filter kernel are a) repeatedly used to compute activation maps and b) spatially restricted, whereas, for regular neural networks, every weight affects the output only once. Through this, the number of parameters to be stored, compared to regular neural networks, is significantly decreased, and convolutional networks are the de-facto standard for deep learning-based computer vision.

1×1 convolution. 1×1 convolutions [76] are broadly used in deep learning, as these operations do not affect the spatial dimension of the input, but can alter the depth. Whereas convolution layers with a stride of 2, for example, reduce the spatial dimension of the input volume, 1×1 convolutions (with stride one) computes simply the dot product for a single pixel. The filter dimension is $1 \times 1 \times d$, where d is both the filter's and the input volume's depth. If it is desired to halve the input volume's depth, the number of filters of the 1×1 convolution layer must be set as $\frac{d}{2}$.

Atrous or dilated convolution layer. In contrast to regular convolutions (see Figure 2.10 a), atrous or dilated convolution [12, 52, 91] is a convolution type that inserts spacing between the kernel parameters, increasing the kernel's receptive field (*i.e.*, field of view) (b) according to

$$Y(i) = \sum_k X(i + r \cdot k)F(k), \tag{2.22}$$

where Y is the output activation map and X is the input activation map, F is the convolution filter with k parameters, i is each location, and r is the rate. Figure 2.10 depicts rate 2, meaning that one spacing is added between kernel weights. An atrous convolution with rate 1 corresponds to a regular convolution. Since spaces between filter weights are filled with zeros, no additional weights have to be stored.

Depthwise and pointwise convolution layer. Depthwise convolution is a type of convolution where an individual filter kernel convolves a single 2D activation map. Whereas for regular convolutions, the filter depth equals the depth of the input activation volume, the filter depth of a depthwise convolution equals 1, and further, the number of filters equals the depth of the activation volume. Before applying ReLU, depthwise convolutions are often followed by 1×1 convolution that does not alter the depth of the activation volume. This 1×1

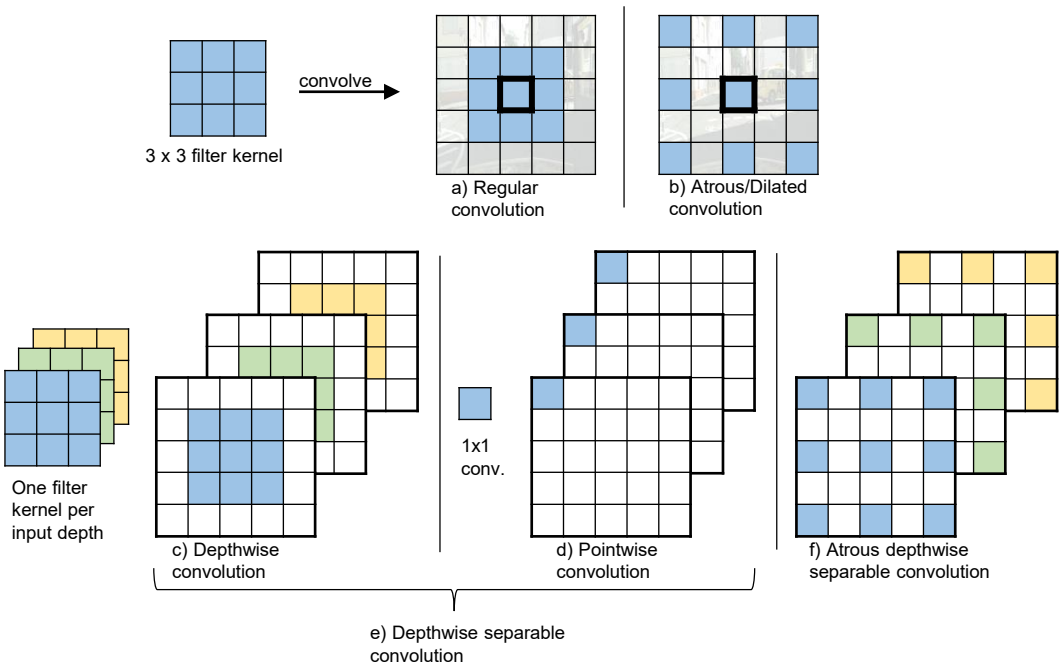


Figure 2.10: An overview of convolution operations. Atrous convolutions insert spacing between kernel parameters (b). Depthwise convolutions process every activation map (*i.e.*, depth 1) separately with a filter kernel (c). A 1×1 convolution, connecting the output of depthwise convolution, is referred to as a depthwise separable convolution (e). Applying atrous convolution in a depthwise convolution layer, instead of regular convolutions, is referred to as an atrous (depthwise) separable convolution (f).

convolution is referred to as pointwise convolution and connects the output of the previous depthwise convolution. The combination of these convolutions is a **depthwise separable convolution** [17] (e). When, in addition, the depthwise convolution consists of an atrous convolution, the final operation is referred to as an **atrous separable convolution** [16] (f). The advantage of applying (atrous) depthwise separable convolutions instead of regular convolutions is a significantly reduced computational budget. Such a network can perform equally well even though the number of learnable parameters is less (and hence, the capacity of the convolutional network).

Pooling layer. Pooling layers are used to spatially downsample the activation volume through pooling operations like max-pooling or average-pooling. One needs to set a stride and a pool width for the pooling operation. A max-pooling layer with pool width 2 and stride 2 downsamples the input by half as illustrated in Figure 2.13 d. **Global average pooling** [76] refers to the operation for which any two-dimensional activation map is aver-

aged to precisely a single number. The influence of global average pooling with respect to a convolutional network's model robustness is evaluated in the next chapter.

Fully-connected layer. A fully-connected layer is the standard layer of a regular neural network. Every weight of a weight matrix W interacts with every data point of the input. Fully-connected layers are usually used as last layers with respect to image classification, mapping the entire input to a class probability vector.

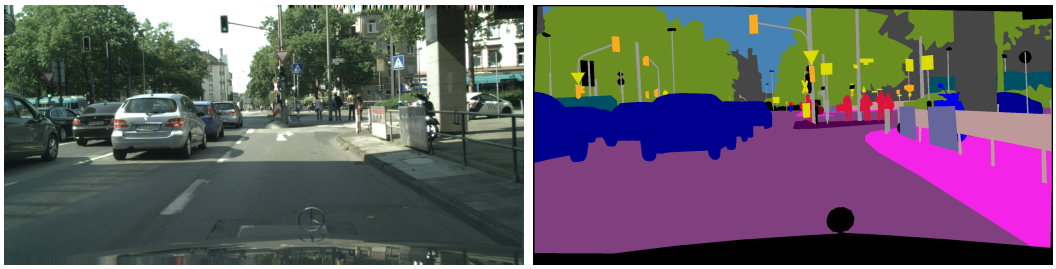
Spatial Pyramid Pooling layer. Unlike convolutional layers, which can process variable-sized input data, a fully-connected layer requires fixed-size input. An unfavorable approach is to rescale or crop every image to a fixed size; however, this results in loss of information. Instead, a spatial pyramid pooling [46, 41, 129] layer ensures the vector size consistency using the Bag-of-Words approach. Each activation map of an activation volume is max-pooled into a fixed number of local spatial bins. For example, an activation map of size 16×16 is max-pooled in bins of size 4×4, 2×2, and 1×1. The total number of bins $(16 + 4 + 1 = 21)$ is then a fixed-size vector, ready to be processed by a fully-connected layer. By binning the activation volume in a fixed number of bins, the network can process variable-sized input. Spatial pyramid pooling arises from the fact that the bins are structured from fine to coarse. This layer's concept is applied in the convolutional network architecture used in the remaining chapters of this thesis.

2.3 Image Segmentation

The computer vision task "semantic image segmentation" is a core aspect of this thesis. Firstly, we cover different types of image segmentation. We then address the background convolutional network architectures, and we present established networks that are used in the following chapters of this thesis.

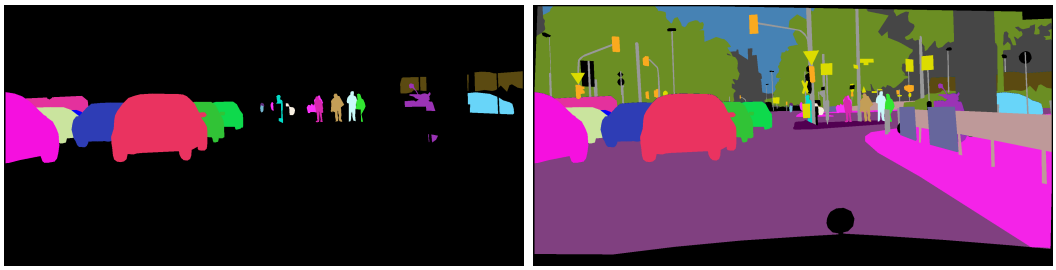
2.3.1 Terminology

Image segmentation is a wide and long studied task in computer vision. It follows the goal of assigning a semantic class to every pixel of an image. The applications of image segmentation are wide-ranging. To name a few: In medical diagnostics, image segmentation can assist specialists in analyzing magnetic resonance and computed tomography scans [107]. Image segmentation can detect plant leaf diseases [85], and classify and differentiate crop [78] in agronomy. Semantic segmentation is also applied to aerial images, segmenting them into roads, buildings, or vegetation, which is essential for navigation applications or for recognizing natural hazards such as floods [96]. Autonomous driving is probably one of the most prominent candidates for computer vision applications. Segmenting the vehicle's environment into cars and traffic lights and further understanding the semantic context is



(a) RGB image

(b) Semantic segmentation



(c) Instance segmentation

(d) Panoptic segmentation

Figure 2.11: An image of the urban driving dataset "Cityscapes" [20] (a) with the respective semantic segmentation ground-truth (b). Image segmentation aims to assign each pixel to the category "things", which are countable objects such as cars, and to the category "stuff", which are pixels of regions such as the sky. Semantic segmentation treats the image content solely as "stuff" (b), since individual instances of, *e.g.*, cars (dark-blue), are not differentiated. The task of instance segmentation focuses on detecting such instances and segmenting the objects of category "things" (c). The combination of both tasks, *i.e.*, semantic and instance segmentation, is referred to as panoptic segmentation (d).

of great importance, particularly with respect to "vulnerable" classes like pedestrians or cyclists, for that the autonomous vehicle's software must pay special attention.

The objects processed by a semantic segmentation algorithm and assigned to a class label are usually categorized into the class categories "stuff" and "things". The classes of category "stuff" are not countable and refers to regions or materials such as vegetation or sky. The classes of the category "things", on the other hand, are countable objects and include, for example, living beings or items. For the task of **semantic segmentation**, the image content is solely treated as "stuff".

On the other hand, we refer to **instance segmentation**, when the goal is to assign an individual label for, *e.g.*, each human in a group of persons. Instance segmentation is closely related to object detection. The application of instance segmentation is domain-specific because only objects of category "things" can be categorized in instances. For example, an image captured in an urban environment is likely to contain several persons

or cars. However, there is, semantically, only a single road or sky. The combination of instance segmentation and semantic segmentation has recently been referred to as **panoptic segmentation**. In summary, semantic segmentation is the generalized form of instance and panoptic segmentation [70], and the main focus of this work. The types of image segmentation are shown in Figure 2.11.

Intersection over Union. In the remaining thesis, we use the Intersection over Union (IoU [31]) as an evaluation metric for semantic segmentation predictions. The IoU for a certain class is calculated as

$$IoU = \frac{TP}{TP + FN + FP}, \tag{2.23}$$

where TP is the number of true-positive predicted pixels, FN and FP is the number of false-negative and false-positive predicted pixels, respectively.

2.3.2 Semantic Segmentation with Deep Learning

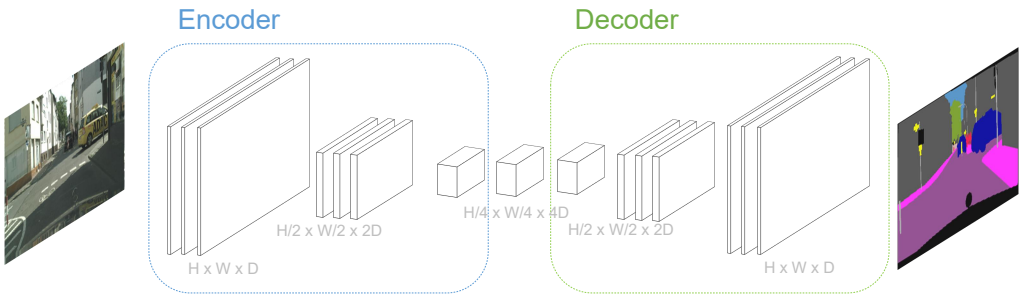


Figure 2.12: An exemplary encoder-decoder structure of a convolutional neural network for semantic image segmentation. The image is firstly processed by an encoder, which is generally a CNN architecture. It consists of several convolution layers that output an activation volume of size $H \times W$ and a depth D. The spatial dimension of the activation volume decreases steadily, in this schema, down to $\frac{H}{4} \times \frac{W}{4}$. The depth of the activation volume, however, increases. The decoder is responsible for transforming the activation volume back to the original spatial dimension, which outputs the semantic segmentation map.

For the task of semantic segmentation, the goal is to assign a class label for every pixel in the image. The question arises, how the architecture of a convolutional network, which predicts pixel-wise class labels, must be designed to output a 2D array containing class labels, *i.e.*, the desired segmentation map. In general, CNNs for semantic segmentation consists of an encoder that downsamples the spatial image dimension of the input image

and a decoder, which steadily increases the spatial dimension again until it matches the desired spatial dimension as illustrated in Figure 2.12. Generally, the encoder is an established convolutional neural network architecture, and convolution layers of diverse types (*e.g.*, atrous, depthwise separable) are its building blocks. Non-linear activation functions are required after the convolution operation. The terminology "encoder" and "decoder" are mostly used in the field of electrical communications engineering. In the context of deep learning-based computer vision, "encoding" describes the process where three-dimensional human-interpretable image data is encoded in an high-dimensional activation volume of size $\frac{W}{16} \times \frac{H}{16} \times 2048D$.[9] This high-dimensional, difficult interpretable activation volume is processed by the decoder, transforming the high-dimensional intermediate activation volume into the desired 2D segmentation map. Convolutional networks for semantic image segmentation are usually fully-convolutional networks [79] (FCN). A convolutional network for image classification can be transformed into a fully-convolutional network by replacing fully-connected layers with *e.g.*, a convolutional layer of filter size 1×1 (see Figure 2.10).

Upsampling. The decoder part of a convolutional network for semantic segmentation requires to up-sample the intermediate activation volumes. Figure 2.13 illustrates different methods. Please see the caption for the description. The architecture which is used in later chapters of this thesis applies mostly bilinear-interpolation (b).

2.3.3 Deep Convolutional Network Architectures for Image Segmentation

State-of-the-art semantic segmentation architectures are deep convolutional neural networks having diverse architectural properties, which we present the following. We refer to the trained variants of convolutional neural network architectures interchangeably as a model and network.

DeepLabv3+

DeepLabv3+ [16] is an architecture whose original version has been expanded three times [14, 12, 15]. In this thesis, this architecture (illustrated in Figure 2.14) is widely used as both baseline and reference models.

Network backbone with atrous convolutions. DeepLabv3+ processes an input image firstly by a deep CNN backbone, such as ResNets [47], Xception-based architectures[17], and MobileNets [54, 53]. In our setup, the spatial dimension of the activation volume processed by ASPP is downsampled by a factor of 16 in most cases. Atrous convolutions are further incorporated into the network backbone.

[9] These dimensions are selected as they occur in the DeepLabv3+ architecture, which will be presented later in detail.

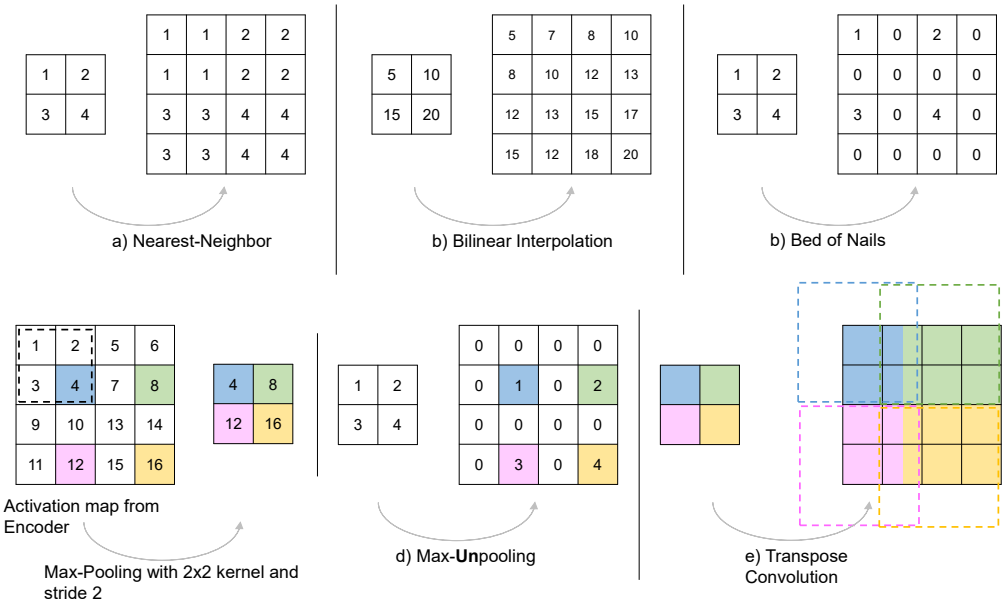


Figure 2.13: Several techniques for upsampling the activation map in the decoder of a convolutional network for semantic image segmentation. In each example, a 2×2 activation map is up-sampled by factor 2. **(a) Nearest-Neighbor:** The value of a cell is applied to its k nearest neighbors. **(b) Bilinear-interpolation:** Both spatial dimensions, *i.e.* width, and height, are linearly interpolated and weighted. **(c) Bed-of-Nails [34]:** Missing values are filled with zeros. **(d) Max-Unpooling [122]:** Similar to bed-of-nails except that the locations of a corresponding max-pooling operation in the encoder are stored. In this example, a 4×4 activation map of the encoder is downsampled to a 2×2 activation map using max-pooling with pool width and stride 2. The "remaining" content is colored. An activation map of 2×2 is "unpooled" in the decoder to 4×4 by using the spatial locations in the grid from the activation map in the encoder. **(e) Transpose convolution [89]:** A convolution kernel is used to weigh values by the 2×2 activation map. In this example, the transpose convolution has a kernel size of 3×3, and stride 2. The value of overlapping kernels is accumulated, which is implied by both colors within a cell of the activation map. As for regular convolutions in the encoder, the filter kernel weights of transposed convolutions are learned.

Atrous Spatial Pyramid Pooling. For extracting features at multiple resolution scales, many semantic segmentation architectures perform Spatial Pyramid Pooling. The Atrous Spatial Pyramid Pooling (ASPP, see Figure 2.14) module of the DeepLabv3+ processes the input by three dilated convolution layers with high symmetric rates r (6, 12, and 18), enabling the network to learn multi-scale representations. The number of weights of

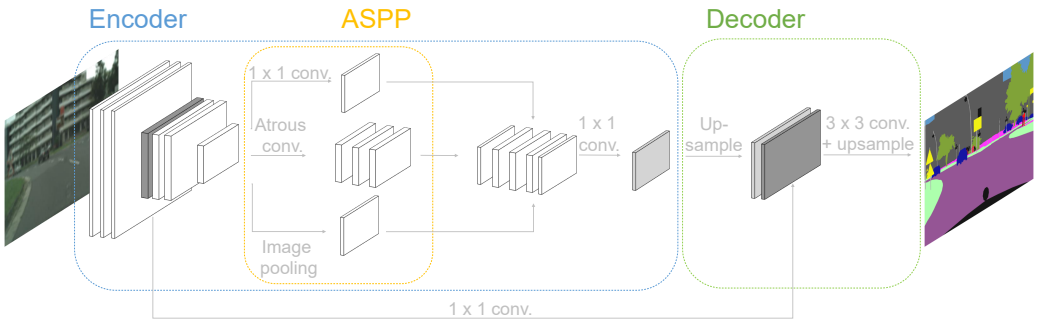


Figure 2.14: Overview of the DeepLabv3+ architecture. The input is firstly processed by the encoder, which consists of a CNN backbone with atrous convolutions. The resulting activation volume is then processed by the Atrous Spatial Pyramid Pooling (ASPP) module, which concatenates resulting activation volumes of a 1×1 convolution, three atrous convolution layer of symmetric rates 6, 12, and 18, and a global average pooling layer. The activation volume is then passed to the decoder, where it is bilinearly upsampled by factor 4 and is subsequently concatenated with low-level features of an early layer of the CNN backbone (processed by a 1×1 convolution first). A final bi-linear upsampling outputs the predicted semantic segmentation map.

an atrous convolution filter computing a dot product with a region of an activation map (that is not zero-padded) decreases for increasing atrous rate r. For example, the 9 weights of a 3×3 filter with an atrous rate of 18, operating on a 65×65 activation map, have roughly 20 % of the dot product computations in image regions that are not zero-padded. If the atrous rate is similar to the activation map's spatial size, the convolution becomes effectively a 1×1 convolution. To counter this, the ASPP consists further of global average pooling (see section 2.2.4) and a 1×1 convolution. Incorporating these properties enables us to learn representations on the image-level and avoids further atrous convolution layers with large rates to mimic 1×1 convolution.

Long-range link. A long-range link combines the encoder's early representations with representations learned by the ASPP or DPC (discussed in the following).

Dense Prediction Cell. Dense Prediction Cell [13] (DPC, see Figure 2.15) is an architectural module for extracting multi-scale representation for dense image segmentation such as the ASPP. However, unlike the ASPP, the DPC is not hand-crafted but designed through a neural-architecture-search, with the goal to maximize performance on semantic segmentation benchmarks.

In the following, we briefly present other convolutional networks for semantic segmenta-

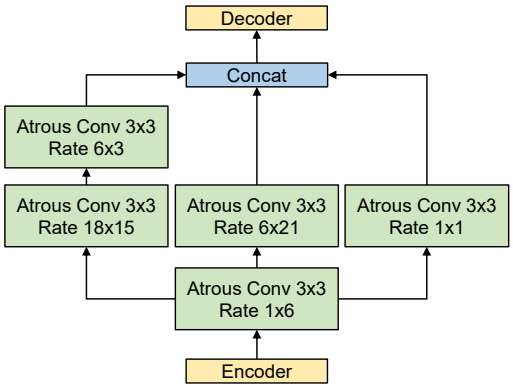


Figure 2.15: Dense Prediction Cell (DPC) of the DeepLabv3+ architecture. It constitutes an alternative multi-scale extraction module to the ASPP. In contrast to the ASPP, besides unsymmetric atrous rates, the ASPP processes its input simultaneously through three atrous convolutions with high rates (6, 12, and 18), whereas the DPC processes the input with a single dilated convolution layer with one element of the spatial dimension being 1.

tion that are mostly explored in chapter 3. As with DeepLabv3+, we will focus less on the established CNN architecture, which is often used as an encoder, but more on that particular architecture's special properties.

PSPNet

The Pyramid Scene Parsing Network [129] (PSPNet) proposed the *pyramid pooling module*, representing the basis predecessor of the ASPP of DeepLabv3+. The pyramid pooling module processes the activation volume created by a CNN backbone (ResNet) in different pyramid scales, which is then similar to DeepLabv3+, followed by a concatenation and an upsampling operation. A primary difference to the ASPP is that regular convolutions are used instead of dilated ones.

ICNet

The image-cascade network [128] (ICNet) is a convolutional network for real-time semantic image segmentation. The principle of ICNet is that large CNN structures (*i.e.*, compute-intensive parts of the architecture with many weights) process a downsampled variant of the input image data, whereas lightweight CNN structures process the images with a higher spatial dimension. In more detail, the input data having a downsampled spatial dimension of factor 4 is processed by the PSPNet. A few convolution layers process the image data both of the original size and the downsampled variant by a factor of 2. The spatial dimension

of the resulting activation maps is adjusted. The logits are then upsampled, creating the prediction of the 2D semantic segmentation map.

DilatedNet

The DilatedNet [120] is an early work that incorporates dilated (atrous) convolutions to learn multi-scale representations. The VGG-16 network is applied as the encoder, for which the last pooling and striding layers are replaced by several dilated convolution layer with increasing atrous rates. This series is referred to as a context module since it aggregates contextual information by systematically expanding the subsequent atrous/dilated convolution layer's receptive field.

Gated-Shape CNN

The Gated-Shape CNN (GSCNN) [112] architecture incorporates an architectural module that explicitly learns the semantic boundaries (or shapes) of the image. Hence, the full objective becomes a multi-task objective since the complete network contains the regular loss function for image segmentation and an additional loss function, evaluating the predicted quality of semantic boundaries.

2.4 Summary

In this chapter, we presented the technical background of image formation, deep learning, and image segmentation. We finalize this chapter with a summary.

Image Formation. The intensity of light arrives in the form of scene radiance at the camera optics, where it is diffracted, hitting the Bayer pattern and, finally, the image sensor. The pinhole camera model describes how a three-dimensional point in space can be mapped onto a 2D image plane by a vector-matrix multiplication between the coordinates of the point in space and the camera matrix C. In real cameras, the pinhole is replaced by the camera optics, which is an array of lenses. Camera optics enable modern photography and video recording by effectively controlling the camera's focus or capturing large field-of views. Camera lenses come along with several aberrations that corrupt the image quality and must hence be mitigated. Most optical aberrations can be modeled with a special convolution kernel referred to as Point-Spread-Function (PSF).

The spectral light irradiance arriving at the image sensor is controlled by the exposure time and the aperture size. The image sensor is then responsible for transforming the counted photons of light into digital R, G, and B values. In current cameras, the image sensor is either a CCD or CMOS sensor. The working principle for both is to count arriving photons and transfer their induced charge in electric voltage. An analog-to-digital converter quantizes

the electric signal, which is stored as the raw image data. The raw data is next processed by the image signal processing (ISP) unit, which usually demosaics the Bayer pattern linearly through interpolating the color values at every location. Besides many processing steps of the ISP, the image is as the last step usually compressed to a viewable/displayable color gamut, such as the standard RGB.

The world's dynamic range is considerably higher than it can be captured with standard sensing technology, resulting in adversely exposed images. A method to increase the dynamic range of images is to create a joint composite of several low dynamic range images referred to as *bracketing*. Through proper tone mapping, the drawback of adversely exposed images can be greatly reduced.

The image formation pipeline is accompanied by image noise sources, which must be taken into account for most image processing algorithms. The most dominant types of image noise are the i) photon shot noise, which is due to the random nature of light, causing that the number of arriving photons in a fixed time interval is not constant, ii) dark current or thermal noise, which is responsible for an undesirable electrical current due to thermal activities in the semiconductor material, and iii) read noise, which occurs in the readout process of integrated electrical devices of the image sensor. In HDR imaging, the SNR discontinuity at the transition of subsequent exposure times creates quite particular image artifacts. In the following chapter, we benchmark and investigate various convolutional network architectures with respect to a wealth of real-world image corruptions, such as imager noise, PSF blur, and geometric distortions.

Deep Learning. Deep learning-based algorithms are the de-facto state-of-the-art for most computer vision tasks. Under the assumption that a dataset's images are independently and identically distributed (i.i.d assumption), a supervised neural network approximates the data generating probability distribution by minimizing the error on the training set. A neural network for, *e.g.*, image classification, can be viewed as a stack of linear classifiers that are "connected" through non-linear activation functions such as the rectified-linear unit. A reason why neural networks are particularly good at learning the (observable) data-generating probability distribution is that the repetitive application of non-linearities fetches the data in a representation in that the linear classifiers can solve the task. The proper setup of a neural network algorithm for any task includes selecting an appropriate output unit (*e.g.*, for image classification, the softmax function is suitable) and regularization to mitigate overfitting. Common regularization techniques are weight decay, which adds a penalty factor to each weight avoiding the weight from becoming too large, and data augmentation (such as random cropping or image scaling) that artificially increases the number of training samples. When the input data being processed are images, the number of parameters would be too large to be processed by the hardware. (Fully) Convolutional networks are the standard types of neural networks for processing images. Instead of applying regular matrix multiplication

between a weight matrix and the input data, convolutions are used, decreasing the number of learnable weights through parameter sharing and sparse connectivity significantly. Deep learning algorithms are commonly trained using the stochastic gradient descent (SGD) algorithm, which iteratively estimates the gradient (through backpropagation) of a mini-batch of some examples. The backpropagation algorithm applies the chain rule of calculus recursively at operation nodes (such as matrix multiplications, activation functions, loss functions) to calculate these nodes' Jacobians throughout the neural network. Backpropagation allows calculating the gradient of the loss concerning the learnable network parameters efficiently. This section presented the fundamental technical background for deep learning, which is the main content of the remaining thesis.

Image Segmentation. Image segmentation is a comprehensive and long studied task in computer vision. It follows the goal of assigning a class to every pixel of an image. The classes of an image are in the context of image segmentation clustered into categories "things" (countable objects such as humans or cars) and "stuff" (non-countable parts of the image like sky or background). When the entire image is treated as stuff, the task is referred to as semantic segmentation, *i.e.*, multiple instances of the same class in an image are not separately classified. The task is referred to as instance segmentation if instances are separately classified. We refer to panoptic segmentation when both tasks are combined, such that every pixel and instance is labeled. The convolutional neural network architectures for conducting semantic segmentation consist in general of an encoding network (a part of the network that sequentially decreases the spatial dimensions) and a decoding network that increases the spatial dimension of the network again through decoding relevant (learned) representation of the encoder. The semantic segmentation architecture "DeepLabv3+" consists of many architectural properties such as dilated convolutions and atrous spatial pyramid pooling, making it a highly suited candidate for a baseline or reference architecture for further studies of the following chapters.

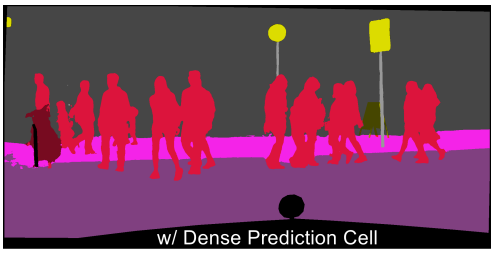
3

Benchmarking the Robustness of Semantic Segmentation Networks

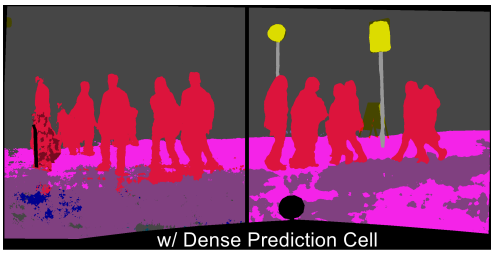
The content associated with this chapter has been published: Besides the following introduction, the entire substantial content of [60] and/or its supplementary material [63] is associated with section 3.1, 3.2, 3.3, 3.4.1, 3.4.2, 3.4.3, 3.5, 3.5.1, the thesis' abstract, introduction (chapter 1) and conclusion (chapter 5). Our publication [61] is an extended version of [60]. Besides the additional content of [61] towards [60, 63] within the previous enumerated sections and chapters, the remaining content of [61] is also associated with section 3.4.4, 3.4.5, and 3.5.2.

Convolutional network's performance is typically measured utilizing publicly available benchmarks that often consist of non-corrupted and post-processed imagery [20, 31]. However, several work has shown that network output is vulnerable to image corruptions [134, 114, 49, 37, 27, 38, 4], in particular image noise decreases model performance considerably. The image quality is affected by environmental aspects such as camera motion, lighting, climate conditions, and ambient temperature since these factors directly impact the camera's optical and electrical properties. Optical aberrations of the camera optics are also influencing the image quality and cause, *e.g.*, image blur. When evaluating semantic segmentation models, there are, generally, various choices: a) comparing different networks, or b) performing a precise ablation study of state-of-the-art network architectures. We conduct both variants since an ablation study (option b) is essential: Understanding architectural properties are likely valuable when constructing a functional system, where the nature of occurring image degradations are known beforehand. The authors in [37] showed that a

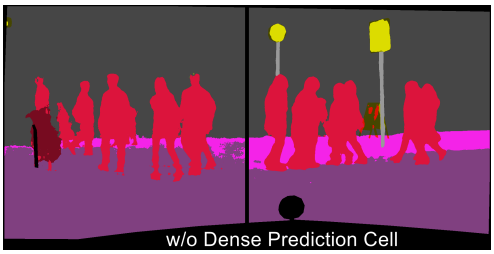
(a) Corrupted validation image (left: noise, right: blur)



(b) Prediction of best-performing architecture on clean image



(c) Prediction of best-performing architecture on corrupted image



(d) Prediction of ablated architecture on corrupted image

Figure 3.1: Results of our benchmark study. Here we train the state-of-the-art semantic segmentation model DeepLabv3+ on clean Cityscapes [20] data and test it on corrupted data. (a) A validation image from Cityscapes, where the left-hand side is corrupted by *shot noise* and the right-hand side by *defocus blur*. (b) Prediction of the best-performing model-variant on the corresponding clean image. (c) Prediction of the same architecture on the corrupted image (a). (d) Prediction of an ablated architecture on the corrupted image (a). We clearly see that prediction (d) is superior to (c), hence the corresponding model is more robust with respect to this image corruption [60] © 2020 IEEE.

GoogLeNet [108] is less robust to image noise than ResNet-152 [47]. Is the former architecture less robust to noise due to the shallower architecture, nonexistent residual connections, or other network-specific architectural design choices?

This chapter proposes an extensive benchmark of convolutional networks for semantic image segmentation with respect to common, real-world image corruptions. The term "robustness" refers in this thesis generally to measuring the performance of a convolutional neural network with respect to images corrupted by common real-world image degradations. Concretely, we organize the benchmark into three parts. Firstly, we benchmark a large-variety of established semantic segmentation architectures, giving a basic overview of different architectures' robustness properties. Secondly, we then benchmark the robustness of established architectural properties applied in semantic segmentation. We alter the net-

works' architectural design, re-train this new model variant, and conduct the benchmark allowing us to derive robust model design rules for semantic segmentation (see Figure 3.1). In both scenarios, the networks are trained on the original, clean training data.

Thirdly, we investigate semantic segmentation models' generalization behavior through training on various portions of corrupted data. We show that the generalization capability of semantic segmentation models depends strongly on the type of image corruption. Models generalize well for image noise and image blur, however, not with respect to other image corruptions of this benchmark.

In the next chapter, we propose a training strategy that generically increases the robustness of semantic segmentation models by increasing the neural network's shape bias.

3.1 Related Work

Several recent works dealt with the robustness of convolutional networks. The work in [4, 30] shows that translating or rotating the test data can drastically change the network's prediction. Azulay *et al.* antialiases intermediate representations and more extensive data augmentation. However, they are only partial solutions to these problems. Engstrom *et al.* find that robustness towards translations and rotations can be increased when the model is trained on a worst-of-10 sample (measured using the loss), but increased the overall training time by factor six. Michaelis *et al.* [84] benchmarks the robustness with respect to common image corruptions for object detection. The authors find significant performance drops for several types of image corruptions and, further, that more complex network backbones are less vulnerable to such image corruptions. Dodge and Karam [27] show, similar to Zhou [134] *et al.*, for full-image classification that CNNs are vulnerable to common image corruptions. The authors present in [28] that the classification performance of corrupted images is drastically lower than human performance.

Vasiljevic *et al.* [114] evaluate the effect of blur for semantic segmentation and full-image classification and find that output accuracy decreases with increasing severity of blur for both tasks.

Geirhos *et al.* [37] compares the generalization capability between convolutional networks and humans. The authors corrupted the ImageNet dataset by color variations, image noise, blur, and rotation. Models trained on image noise do not generalize well to other types of unseen image noise.

Hendrycks *et al.* [49] introduces the "ImageNet-C dataset". The authors corrupt the ImageNet dataset with common, real-world image corruptions, and find that even though the absolute performance from AlexNet [71] to ResNet [47] increases, relative model robustness is not affected. The authors further show that Multigrid and DenseNet architectures [67, 55] are less vulnerable to image noise than ResNets. Zendel *et al.* [124] create a CV model to

apply the hazard and operability analysis (HAZOP) to the computer vision domain and further provides an extensive checklist for image corruptions and visual hazards. This study demonstrates that most forms of image corruptions do also negatively influence stereo vision algorithms. Zendel *et al.* [123] propose a fruitful segmentation test-dataset ("WildDash") containing challenging visual hazards such as overexposure, lens distortion, or occlusions.

Yin *et al.* [118] find that training a model on one type of image corruption decreased performance on another type of image corruption. The authors conclude that augmenting data with a diverse set of low and high-frequency corruptions increases network robustness. In this thesis, we use many of the ImageNet-C corruptions and apply them to PASCAL VOC 2012, the Cityscapes dataset, ADE20K [31, 20, 132, 133]. Other work deals with robustness towards adverse weather conditions [104, 105, 115], night scenes [23], or geometric transformations [33, 100].

The term "robustness" refers in the deep learning context usually to adversarial robustness, which is an very active field of research (see, for example [57, 18, 42, 10, 83, 9]. Arnab *et al.* [3] is related to this chapter. The authors evaluated the robustness of semantic segmentation models for adversarial attacks with respect to many convolutional network architectures (*e.g.* [129, 5, 92, 128, 120]). However, we focus on the robustness with respect to common image corruptions and not with respect to adversarial attacks. Moreover, similar to the previously presented related work in this section, we modify a single architectural property per network, allowing a significant evaluation of that particular property's influence to model robustness. In our view, such an evaluation is not entirely given in [3] and the other related work.

Gilmer *et al.* [38] connect robustness w.r.t. Gaussian noise and robustness w.r.t. adversarial examples. The authors demonstrate that training methods increasing adversarial model robustness also come with an increase in the model robustness against several common image corruptions. A parallel work of [102] shows, however, that adversarial training reduces corruption robustness.

3.2 Common Image Corruptions

We rate the performance of semantic segmentation models with respect to a diverse range of common image corruptions. Most of the utilized corruptions are adopted from the ImageNet-C dataset by Hendrycks *et al.* [49], which we applied on several semantic segmentation datasets. We further extend the collection of corruptions provided by ImageNet-C by further ones, particularly PSF blur, intensity-dependent camera noise, and geometric distortions, that can be treated as proxy covering the considerable diversity of naturally occurring real-world image corruptions. We thus test every model on roughly $400,000$ corrupted images.

3.2.1 Common Image Corruptions of ImageNet-C

We utilize numerous image corruptions of ImageNet-C dataset [49], which comprises of different types of blur (motion, frosted glass, defocus, and Gaussian), image noise (Gaussian, shot, impulse and speckle), weather (snow, spatter, fog, and frost) and digital (contrast, brightness, and JPEG compression). Examples for a corrupted variant of an image of the Cityscapes dataset are shown in Figure 3.2). All image corruptions are parameterized with five severity levels, as shown for several candidates in Figure 3.3.

Figure 3.2: Illustration of Cityscapes-C [20] by utilizing the image corruptions provided by[49]. **First row:** Brightness, contrast, saturate, JPEG. **Second row:** Snow, spatter, fog, frost. For ease of reference, the following corruptions show the image region of the first images' red rectangle. **Third row:** Motion blur, defocus blur, frosted glass blur, Gaussian blur. **Fourth row:** Gaussian noise, impulse noise, Shot noise, speckle noise.

3.2.2 Camera-Based Common Image Corruptions

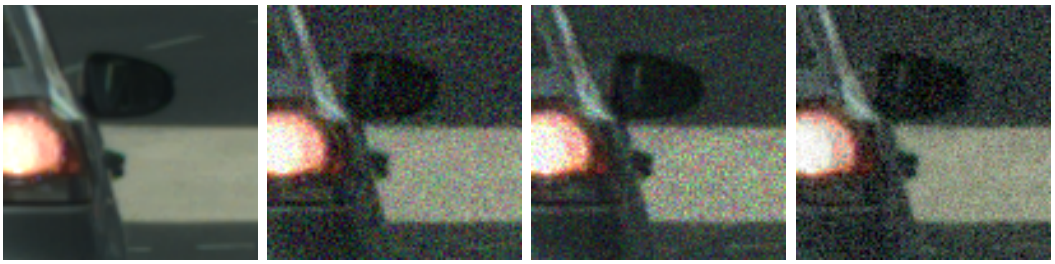
Intensity-Dependent Noise Model. Convolutional neural networks are vulnerable to image noise (see section 3.1). The previously presented noise models are generally relatively simple, *e.g.*, the imagery is uniformly corrupted with additive Gaussian noise. However, as described in section 2.1.5, real-world image noise is significantly more complex than the noise simulated by such a model since a variety of noise sources, such as photon shot noise, kTC noise, and dark current noise, affect the total. Real-world image noise is a mixture of numerous sorts of noise that contribute to the total noise level.

Figure 3.3: Illustration of the first, third and fifth severity level of Cityscapes-C [20] for a candidate of the categories *blur*, *noise*, *digital*, and *weather*. **First row:** Motion blur. **Second row:** Gaussian noise. **Third row:** Contrast. **Fourth row:** Snow

The proposed image noise model mimics commonly observable image noise caused by real-world cameras. The noise value added to a clean pixel in the linear color space is a sum of two features: Random *chrominance noise* and *luminance noise* values are drawn, which depends on the pixel's intensity of the respective color gamut. With the term *chrominance noise*, we refer to randomly chosen noise components for each pixel color, resulting in color noise. *Luminance noise*, on the other hand, refers to a random noise value added to each channel of a pixel equally, resulting in gray-scale noise. Like image noise caused by real cameras (*e.g.*, shot noise), pixels of darker image regions are noisier than pixels of brighter image regions. We model the noisy pixel intensity for a color channel c as a random variable $I_{noise,c}$:

$$\begin{aligned} I_{noise,c}(\Phi_c, N_{lum}, N_{chrom,c}; w_s) = \\ log_2(2^{\Phi_c} + w_s \cdot (N_{lum} + N_{chrom,c})), \end{aligned} \tag{3.1}$$

where Φ_c is the normalized pixel intensity of color channel c, N_{lum} and N_{chrom} are random variables following a Normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$, w_s is a weight factor, parameterized by one of the five available severity level s.



(a) Clean (b) Gaussian (c) Shot (d) Intensity

Figure 3.4: A crop of a validation image from Cityscapes [20] corrupted by various noise models. (a) Clean image. (b) Gaussian noise. (c) Shot noise. (d) Our proposed noise model. The amount of noise is high in regions with low pixel intensity [60] © 2020 IEEE.

Figure 3.4 illustrates noisy variants of an image-crop of the Cityscapes dataset. Our noise model's image noise depends clearly more on pixel intensity than the other noise models. We also propose a less generic noise model, imitating auto-bracketing HDR image noise (see Appendix A.1).

PSF blur. Every camera optics exhibits optical aberrations that mostly cause image blur due to refracting incoming light not in a common center (see section 2.1.2). We created real-world Point-Spread-Functions (PSF) in three severity levels using the optical design program *Zemax*, aggregating all optical aberrations causing spatially-varying image blur. The degree of blur increases, further, with a larger distance to the image center for our PSFs. The PSF models correspond to a customary front video automotive video camera with a horizontal field-of-view of 90 °. We denote this type of corruption in the following benchmark as PSF blur. We illustrate the intensity distribution of several PSF kernels for different angles of incidence in Figure 3.5.

Geometric distortion. Another property of a camera lens are geometric distortions, such as radial distortions, as discussed in section 2.1.2. Radial distortions are an important aspect of a robustness benchmark for two reasons. i) Distortion parameters of an optical system vary over time, are affected by environmental influences, differ from calibration stages, and thus, may never be fully compensated. ii) Additionally, image warping, which is part of the correcting process, can cause re-sampling artifacts that degrade the image's informational content. Therefore, one might prefer to use the geometrically distorted (*i.e.*, the original) image for tasks such as feature detection [44, p. 192f]. We use the command-line image processing tool *imagemagick* to geometrically distort both the test images of datasets and the respective ground truth as a polynomial of grade 4 [106] for our benchmark.

Figure 3.6 shows examples of our proposed common image corruptions.

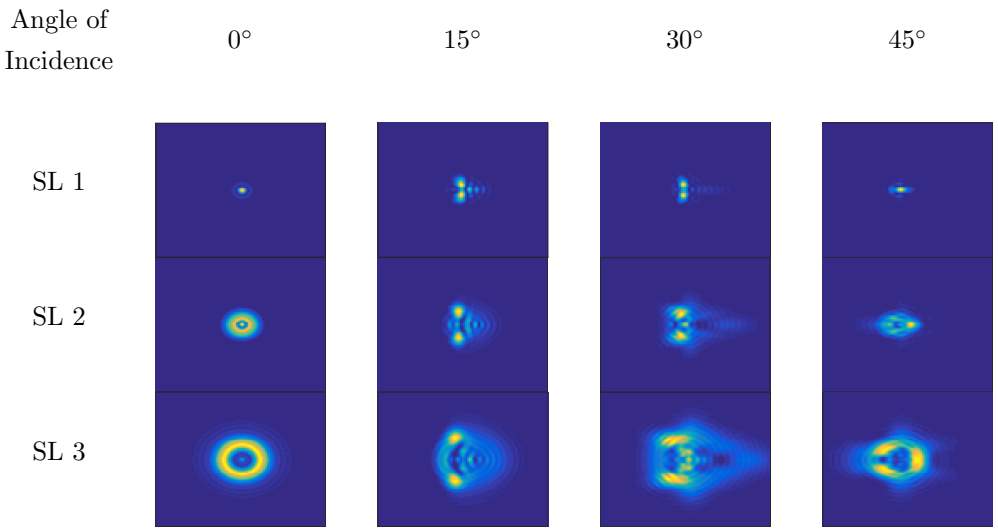


Figure 3.5: The intensity distribution of our modeled PSF kernels, which increases with the **severity level (SL)**. The shape of the PSF kernel depends on the image region, *i.e.*, the angle of incidence.

Figure 3.6: Illustration of geometric distortion and PSF blur applied on Cityscapes [20]. From left to right: clean image, radial barrel distortion, a crop of a clean image, and an image with PSF blur. PSF blur is quite pronounced near image edges.

3.3 Convolutional Network Architectures and Properties

We benchmark a large variety of semantic segmentation models and several architectural properties with respect to common image corruptions, mainly based on DeepLabv3+. Please see a detailed description of each architecture and property in section 2.3.3. To recall, it can be used with many network backbones, varying from state-of-art architectures (e.g., modified aligned Xception [94], denoted by Xception), several ResNets and MobileNets. DeepLabv3+ consists of established architectural properties, widely used for semantic image segmentation, resulting in a very appropriate candidate for an ablation study. DeepLabv3+ functions in this benchmark as the reference architecture and is illustrated, together with ablated variants, in Figure 3.7. The architecture we consider as the reference is indicated

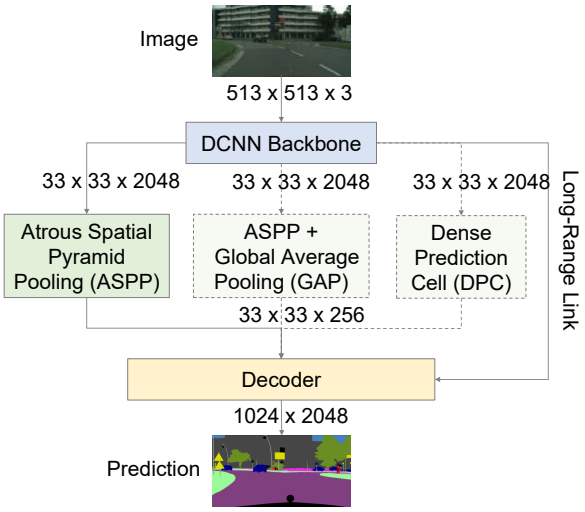


Figure 3.7: The reference architecture of DeepLabv3+ and ablated architectural variants. Input images are processed by the network backbone, containing atrous convolutions. The backbone output is further processed by a module (ASPP or DPC) extracting multi-scale features. A long-range link concatenates early features of the network backbone with backbone output. The decoder creates final estimates of semantic labels. Our reference model is shown by solid arrows (*i.e.*, without DPC and GAP) and ablated variants as dashed ones. The dimension of activation volumes is shown after each block [60] © 2020 IEEE.

by solid arrows, whereas dashed ones indicate each ablated variant. One model backbone (MobileNet, ResNet, or Xception) with atrous convolutions (**AC**) processes the input image. The backbone's output is then processed by one of the multi-scale modules that extract dense feature maps. The module is either Atrous Spatial Pyramid Pooling (**ASPP** or Dense Prediction Cell (**DPC**), whereas the former is without or with global average pooling (**GAP**)). The long-range link (**LRL**) combines early representations learned by the network backbone with representations extracted by ASPP or DPC. Finally, the decoder predicts the segmentation map.

To summarize, we consider the architecture with ASPP (instead of DPC), atrous convolution, long-range link, and without global average pooling as the reference. For the benchmark of ablated variants, we alter one property at a time, and we refer to the architecture, *e.g.*, without atrous convolutions as w\o AC (similar for ASPP/LRL). Replacing DPC by ASPP is indicated as w\o ASPP+w\DPC and adding GAP is referred to as w\GAP.

In addition to the network backbones offered by DeepLabv3+, we further benchmark a large variety of additional semantic segmentation networks, such as FCN8s (VGG-16), ICNet, DilatedNet, ResNet-38 [116] (ResNet-38 is a more shallow ResNet variant which

trades of width, *i.e.*, number of filters of a convolution layer, and depth, *i.e.*, number of stacked layers), PSPNet, and Gated-Shape-CNNs.

The utilized architectures and ablated variants are summarized in Table 3.1.

Architecture	**Architectural Modification**	**Backbone**
DeepLabv3+	for each backbone: removal of ASPP, AC, LRL, replace ASPP by DPC, utilize GAP	MobileNet-V2 ResNet-50/101 Xception-41/65/71
FCN8s	–	VGG-16
ICNet	–	ResNet
DilatedNet	–	VGG-16
ResNet-38	–	ResNet
PSPNet	–	ResNet
Gated Shape CNN	–	ResNet

Table 3.1: Overview of benchmarked convolutional network architectures. **(top)** For DeepLabv3+ we benchmarked in total six different network backbones, and trained for each backbone, in turn, the reference architecture and five ablated variants, respectively. We removed singly i) the Atrous Spatial Pyramid Pooling module (ASPP) ii) atrous convolutions (AC) iii) long-range link (LRL), iv) replaced ASPP by the dense prediction cell (DPC), and v) added global average pooling (GAP). **(bottom)** We also benchmarked several other, non-Deeplabv3+ based, semantic segmentation architectures.

3.4 Experiments

We present now the setup of our benchmark experiments and report results of architecture (section 3.4.2) and property (section 3.4.3) robustness with respect to common image corruptions. The former approach gives the overall robustness of numerous architectures, whereas the latter allows us to derive architectural properties' effect on model robustness. This section shows that architectural properties can significantly influence the robustness of semantic segmentation networks. Based on these results, we derive robust model design rules in section 3.5.1. We also discuss the generalization behavior of the presented semantic segmentation models through training them on corrupted image data in section 3.4.5. We show that DeepLabv3+ generalizes well to many types of image noise and blur.

3.4.1 Experimental Configuration

DeepLabv3+ network backbones. The DeepLabv3+ with many network backbones (MobileNet-V2 (MN-V2), ResNet-50 (RN-50), ResNet-101 (RN-101), Xception-41 (XC-41),

Xception-65 (XC-65) and Xception-71 (XC-71)) is trained on original (and in section 3.4.5 also on corrupted) image data with TensorFlow [1]. Each model is trained with crop-size 513×513, batch size 16, random scale data augmentation, fine-tuning batch normalization parameters [58], and initial learning rate 0.01 or 0.007.

Datasets. We conduct our benchmark on the three semantic segmentation datasets the Cityscapes dataset, PASCAL VOC 2012, and ADE20K for training and validation. The fine pixel-level annotations Cityscapes consists of 2975 train and 500 validation images, PASCAL VOC 2012 of $1,464$ train and $1,449$ validation images, and ADE20K of $20,210$ train, 2000 validation images, and 150 semantic classes with strongly varying image dimensions.

Evaluation metrics. The Intersection-over-Union (mIoU [31]) is used to rate semantic segmentation perfomance. We average the scores for an individual image corruption across its severity levels. We additionally revise the notion of Corruption Error and relative Corruption Error from [49]: The term *Degradation D*, where $D = 1 - mIoU$ is used in place of *Error*. We aggregate degradations D across severity levels, which are modeled by [49]. For mutually comparable models, the degradation D of a trained model f is divided through the degradation of a reference model ref. Doing so defines the *Corruption Degradation* (CD) of a model as

$$CD_c^f = \left(\sum_{s=1}^{5} D_{s,c}^f\right) \Big/ \left(\sum_{s=1}^{5} D_{s,c}^{ref}\right), \tag{3.2}$$

where c denotes the corruption type (*e.g.*, Gaussian blur) and s its severity level. For image noise-based corruptions, we considered only the first three severity levels. The CD is mainly used for evaluating and robustness across several models. Nevertheless, the relative Corruption Degradation (rCD) metric is also used, incorporating model degradation with respect to clean data.

$$rCD_c^f = \left(\sum_{s=1}^{5} D_{s,c}^f - D_{clean}^f\right) \Big/ \left(\sum_{s=1}^{5} D_{s,c}^{ref} - D_{clean}^{ref}\right). \tag{3.3}$$

The rCD, on the other hand, incorporates the respective model performance on clean data. The degradation of clean data is for both models (*i.e.*, the model for which the robustness is to be rated, and the reference model) is subtracted, resulting in a measure that gives a ratio of the absolute performance decrease in the presence of image corruption.

3.4.2 Benchmarking Neural Network Architectures

We train various DeepLabv3+ networks and non-DeepLab based semantic segmentation architectures on the clean, original Cityscapes training set. Table 3.2 shows the corresponding mean-IoU for which is averaged over the severity levels of each image corruption. The top (bottom) part of the Table contains the results of the benchmark for DeepLabv3+ (non-DeepLab) based models.

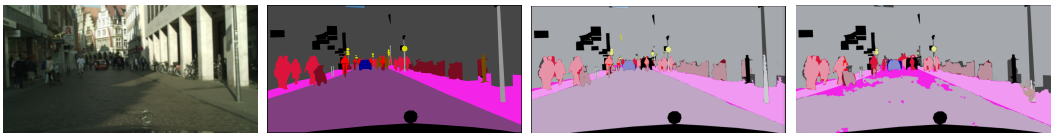
		Blur					**Noise**					**Digital**				**Weather**				
Architecture	Clean	Motion	Defocus	Glass	Gaussian	PSF	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	72.0	53.5	49.0	45.3	49.1	70.5	6.4	7.0	6.6	16.6	26.9	51.7	46.7	32.4	27.2	13.7	38.9	47.4	17.3	65.5
RN-50	76.6	58.5	56.6	47.2	57.7	74.8	6.5	7.2	10.0	31.1	30.9	58.2	54.7	41.3	27.4	12.0	42.0	55.9	22.8	69.5
RN-101	77.1	59.1	56.3	47.7	57.3	75.2	13.2	13.9	16.3	36.9	39.9	59.2	54.5	41.5	37.4	11.9	47.8	55.1	22.7	69.7
XC-41	77.8	61.6	54.9	51.0	54.7	76.1	**17.0**	**17.3**	**21.6**	**43.7**	48.6	63.6	56.9	**51.7**	38.5	18.2	46.6	57.6	20.6	**73.0**
XC-65	78.4	63.9	59.1	**52.8**	59.2	**76.8**	15.0	10.6	19.8	42.4	46.5	65.9	**59.1**	46.1	31.4	**19.3**	**50.7**	63.6	**23.8**	72.7
XC-71	**78.6**	**64.1**	**60.9**	52.0	**60.4**	76.4	14.9	10.8	19.4	41.2	**50.2**	**68.0**	58.7	47.1	**40.2**	18.8	50.4	**64.1**	20.2	71.0
ICNet	65.9	45.8	44.6	**47.4**	44.7	65.2	8.4	8.4	10.6	27.9	29.7	41.0	33.1	27.5	**34.0**	6.3	30.5	27.3	11.0	56.5
VGG-16	66.7	42.7	31.1	37.0	34.1	61.4	6.7	5.7	7.8	24.9	18.8	53.3	39.0	36.0	21.2	11.3	31.6	37.6	19.7	62.5
Dilated	68.6	44.4	36.3	32.5	38.4	61.1	**15.6**	14.0	**18.4**	32.7	35.4	52.7	32.6	38.1	29.1	12.5	32.3	34.7	19.2	63.3
RN-38	77.5	54.6	45.1	43.3	47.2	74.9	13.7	**16.0**	18.2	**38.3**	**35.9**	60.0	50.6	46.9	14.7	**13.5**	45.9	52.9	22.2	73.2
PSPNet	78.8	**59.8**	53.2	44.4	53.9	76.9	11.0	15.4	15.4	34.2	32.4	60.4	51.8	30.6	21.4	8.4	42.7	34.4	16.2	73.6
GSCNN	**80.9**	58.9	**58.4**	41.9	**60.1**	**80.3**	5.5	2.6	6.8	24.7	29.7	**75.9**	**61.9**	**70.7**	12.0	12.4	**47.3**	**67.9**	**32.6**	**76.4**

Table 3.2: Mean-IoU averaged over severity levels for clean and corrupted variants of the Cityscapes validation set for several network backbones of the DeepLabv3+ architectures MobileNet-V2 (MN-V2), ResNets (RN), and Xceptions (XC) **(top)** and non-DeepLab based models **(bottom)**. Every mIoU is averaged over all available severity levels, except for corruptions of category noise where only the first three (of five) severity levels are considered. Xception based network backbones are most robust against each corruption in most cases. Most models are robust against our realistic PSF blur. Highest mIoU per corruption is bold [60] © 2020 IEEE.

Performance w.r.t. clean. With respect to DeepLabv3+, Xception-71 has the highest mIoU of 78.6 %. Many Xceptions are further considerably better than ResNets and MobileNet-V2. The Gated-Shape CNN has the highest mIoU on clean data of every tested model, exceeding 80 %. The mIoU is in this benchmark the lowest for ICNet (65.9 %).

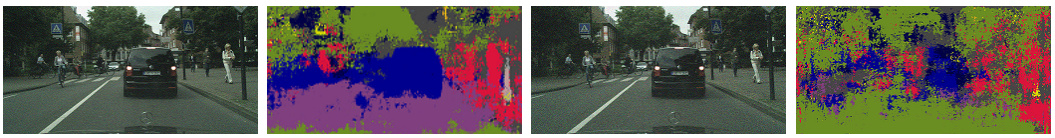
Performance w.r.t. blur. Every model (except DilatedNet and VGG16) performs decently for PSF blur since the mIoU drops by approximately 2 %, respectively. Interestingly, a light network backbone like the MobileNet-V2 is barely prone to this real-world image blur. Image blur leads on Cityscapes often to confusions between similar classes, such as rider and person, especially for far-distant objects. False-negative and false-positive predictions occur, resulting in overlooked persons (see Figure 3.8).

Performance w.r.t. noise. The mIoU scores presented for image corruption category noise are calculated through averaging over the first three severity levels (instead of five) due to the drastically corrupting impact on model performance (see Figure 3.9). Xception-based network backbones of DeepLabv3+ are generally best-performing with respect to image noise and often better than non-DeepLabv3+ models. Lightweight architectures, such as MobileNet-V2, ICNet, VGG-16, and GSCNN, are way more vulnerable to image noise than remaining architectures. Surprisingly, the best model on clean data (GSCNN) is by far the most vulnerable for image noise.



(a) Blurred validation image (b) ground truth (gt) (c) Overlay clean estimate + gt (d) Overlay blur estimate + gt

Figure 3.8: Prediction of the reference architecture of DeepLabv3+ on blurred input data, using Xception-71 as network backbone. (a) A blurred validation image of the Cityscapes dataset [20] and corresponding ground truth (b). (c) Prediction of the clean image overlaid with the ground truth. In this visualization, true-positives are alpha-blended, false-positives, and false-negatives remain unchanged. Hence, wrongly classified pixels can be easier spotted. (d) Prediction on the blurred image overlaid with the ground truth (b). Whereas the riders are mostly correctly classified in (c), they are in (d) miss-classified as persons. Extensive areas of road are wrongly classified as sidewalk.



(a) Corrupted validation image (b) Prediction on (a) (c) Corrupted validation image (d) Prediction on (c)

Figure 3.9: Drastic influence of image noise on model performance. (a) A validation image of Cityscapes [20] is corrupted by the second severity level of Gaussian noise and respective prediction (b). (c) A validation image of Cityscapes is corrupted by the third severity level of Gaussian Noise and respective prediction (d). Predictions are produced by the reference model, using Xception-71.

Performance w.r.t. digital. Most architectures handle the first severity levels of corruption types contrast, brightness, and saturation well. JPEG compression, on the other hand, decreases mIoU considerably. It is worth noting that PSPNet and GSCNN have for JPEG compression halved or even a lower mIoU score than Xception-41 and -71, even though all models have similar performance on the original data.

Performance w.r.t. weather. The model performance drops particularly towards image corruptions, which significantly degrade the textural information, such as frost and snow.

Performance w.r.t. geometric distortion. The GSCNN is the most robust model against this image corruption. Whereas most models withstand the first severity level well, the mIoU of GSCNN drops only by less than 1 %.

This benchmark shows a similar outcome as [36]: Image corruptions degrading an image's texture (*e.g.*, image noise, snow, frost, JPEG), often decrease model performance significantly more than corruptions that preserve image texture to some degree (*e.g.*, brightness, blur, geometric distortion, contrast).

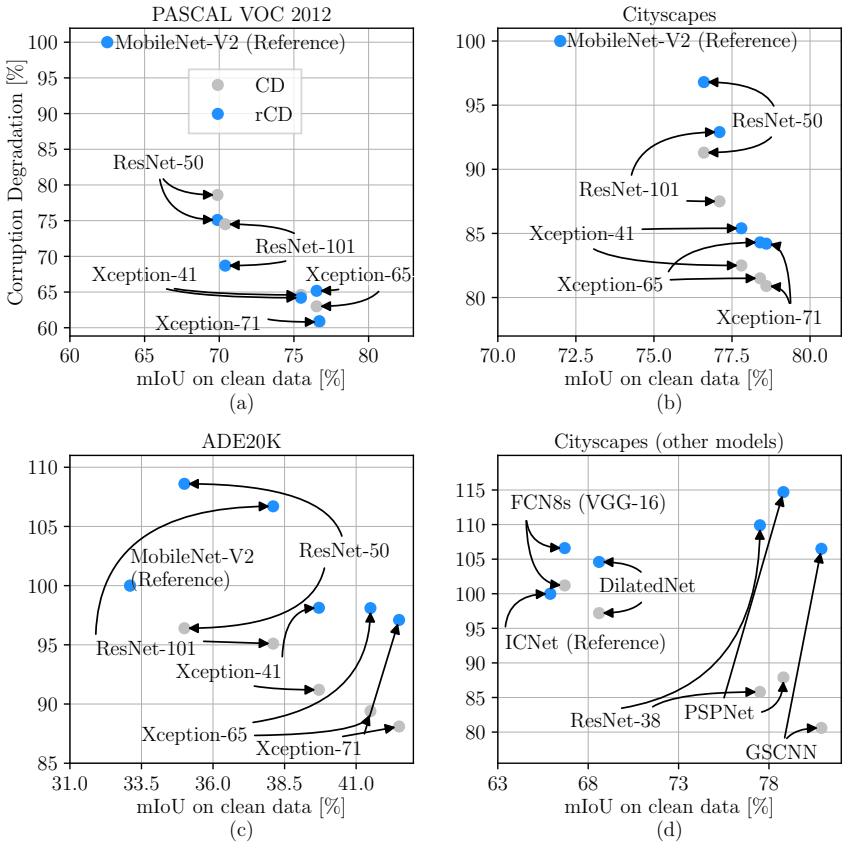


Figure 3.10: (a–c) CD and rCD for several network backbones of the DeepLabv3+ architecture evaluated on PASCAL VOC 2012, the Cityscapes dataset, and ADE20K. MobileNet-V2 is the reference model in each case. rCD and CD values below 100 % represent higher robustness than the reference model. In almost every case, model robustness increases with model performance (*i.e.*, mIoU on clean data). Xception-71 is the most robust network backbone on each dataset. (d) CD and rCD for non-DeepLabv3+ based models evaluated on Cityscapes. While CD decreases with increasing performance on clean data, rCD is larger than 100 % [60] © 2020 IEEE.

We now consider the two metrics Corruption Degradation (CD) and relative Corruption Degradation (rCD) to rate the robustness with respect to image corruptions of the presented architectures and network backbones. Figure 3.10 presents the CD and rCD scores averaged

overall image corruptions with respect to the mIoU evaluated on clean (*i.e.*, original) image data (lower scores indicate higher robustness). We do not consider PSF blur in this Figure. Due to the way smaller PSF blur's influence blur on the mIoU, minor changes in mIoU of less than 0.5 % can considerably influence the rCD.

Discussion of CD. Subplot a−c shows individual results for PASCAL VOC 2012, Cityscapes, and ADE20K, and subplot d shows the results for the non-DeepLab-based networks evaluated on Cityscapes. The CD for Xception-71 is the lowest (meaning most robust) for DeepLabv3+ architecture for every dataset, and it generally decreases with increasing mIoU on clean data. Except for PSPNet and FCN8s (VGG16), a similar result can be seen for the non-DeepLab models. Among them, the most robust architecture is clearly the GSCNN. The poor performance of MobileNet-V2 on clean data causes low CD scores with respect to PASCAL VOC 2012. The CD for networks evaluated on the Cityscapes dataset (subplot b and d) are comparable, even though the respective reference models differ, having comparable mIoU on clean data.

Discussion of rCD. On the other hand, the relative Corruption Degradation (rCD) between subplots a−c and subplot d are contrary. With respect to subplots a−c, the rCD decreases similar to the CD except for a few exceptions. The authors of [49] show a similar result for full-image classification: The rCD for several benchmarked models is more or less constant, though the accuracy on clean data differs strongly, as Figure 3.10 d) shows as well. However, evaluating within an entire semantic segmentation architecture such as the DeepLabv3+ results in a somewhat contrary result (*i.e.*, decreasing rCD). This finding is similar to [90, 84] for other computer vision tasks.

The following hypothesis might explain the previously presented result. Geirhos *et al.* [36] demonstrated that (i) networks for full-image classification use object texture, rather than object shape, for solving the classification task, and (ii) network performance w.r.t. image corruption increases for models relying stronger on object shapes than textures.

If we transfer these findings to semantic segmentation, a possible explanation for the superior performance of Xception-71 and the GSCNN might be that these networks have a higher shape bias than, *e.g.*, ResNets.

We finalize this section with Figure 3.11, presenting the CD and rCD averaged for the proposed image corruption categories for non-DeepLabv3+ based models. The CD of image corruption "jpeg compression" of category digital is not included in this barplot since, contrary to the remaining image corruptions of that category, the respective CDs range between 107 % and 133 %. FCN8s-VGG16 and DilatedNet are prone to image corruptions of category blur. DilatedNet is more robust against many corruptions of category noise, digital, and weather than the reference architecture. ResNet-38 is extraordinarily robust against corruptions of category weather. The rCD of PSPNet is oftentimes higher than 100 % for each

category. GSCNN is vulnerable to image noise. High rCDs indicate a significant decrease of mIoU in the presence of this corruption.

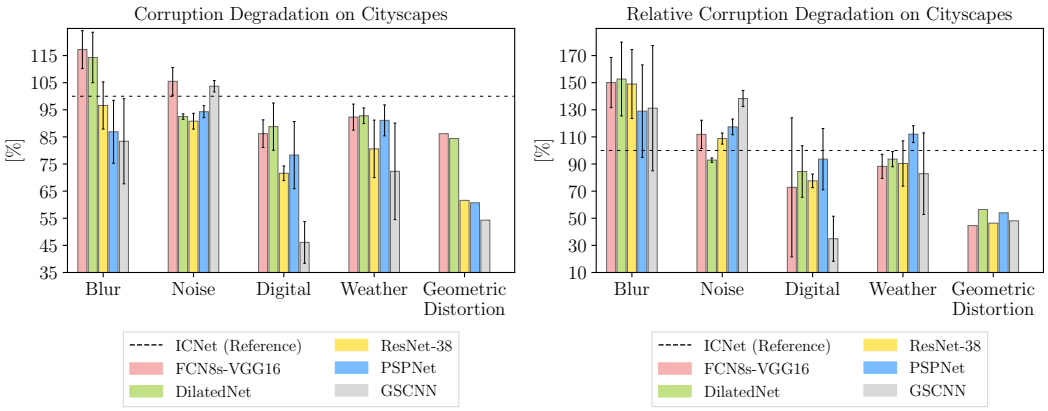


Figure 3.11: CD **(left)** and rCD **(right)** evaluated on Cityscapes for ICNet (used as reference architecture), FCN8s-VGG16, DilatedNet, ResNet-38, PSPNet, GSCNN with respect to common image corruptions of our benchmark. Bars above 100 % represent a decrease in performance compared to the reference architecture. Each bar except for geometric distortion is averaged within a corruption category (error bars indicate the standard deviation).

3.4.3 Benchmarking Architectural Properties

This section presents and discusses the results of an extensive ablation for the DeepLab3+ architecture. We modify a single architectural property per model, allowing a significant evaluation of that particular property's influence to model robustness. To recall, we ablate the following network backbones: Xception-71 (XC-71), Xception-65 (XC-65), Xception-41 (XC-41), ResNet-101, ResNet-50 and, MobileNet-V2 (MN-V2) as described in the experimental setup previously. XC-71 performs best for clean data but is computationally the most expensive backbone. On the other hand, MobileNet-V2 requires approximately a tenth of the XC-71's storage space. For each backbone, we ablated the architectural properties as listed in Table 3.1. We then train each candidate on the clean training set of Cityscapes, PASCAL VOC 2012, and ADE20K, aggregating to more than 100 trainings. Only the most distinct, statistically significant results are discussed in the following sections.

Cityscapes

Table 3.3 lists the averaged mIoU for XC-71. The removal of ASPP from Xception-71 reduces mIoU form 78.6 % to 73.9 %. The ablated architecture without atrous convolutions

		Blur					Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	PSF	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
XC-71	78.6	64.1	60.9	52.0	60.4	76.4	14.9	10.8	19.4	41.2	**50.2**	**68.0**	58.7	47.1	40.2	**18.8**	50.4	64.1	20.2	71.0
w/o ASPP	73.9	60.7	59.5	51.5	58.4	72.8	**18.5**	**14.7**	**22.3**	39.8	44.7	63.4	56.2	42.7	39.9	17.6	49.0	58.3	21.8	69.3
w/o AC	77.9	62.2	57.9	51.8	58.2	76.1	7.7	5.7	11.2	32.8	43.2	67.6	55.6	46.0	**40.7**	18.2	50.1	61.1	**21.6**	71.1
w/o ASPP + w/ DPC	**78.8**	62.8	59.4	52.6	58.2	76.9	7.3	2.8	10.7	33.0	42.4	64.8	59.4	45.3	32.0	14.4	48.6	64.0	20.8	72.1
w/o LRL	77.9	64.2	**63.2**	50.7	**62.2**	76.7	13.9	9.3	18.2	**41.3**	49.9	64.5	59.2	44.3	36.1	16.9	48.7	**64.3**	21.3	71.3
w/ GAP	78.6	**64.2**	61.7	**55.9**	60.7	**77.8**	9.7	8.4	13.9	36.9	45.6	68.0	**60.2**	**48.4**	40.6	16.8	**51.0**	62.1	20.9	**73.6**

Table 3.3: Mean-IoU averaged over severity levels for clean and corrupted variants of the Cityscapes validation dataset for Xception-71 and five corresponding architectural ablations. Based on DeepLabv3+, we evaluate the removal of atrous spatial pyramid pooling (**ASPP**), atrous convolutions (**AC**), and long-range link (**LRL**) on model robustness with respect to common image corruptions. We further replaced ASPP by Dense Prediction Cell (**DPC**) and utilized global average pooling (**GAP**). The standard deviation for non-deterministic corruptions is 0.3 or less. Highest mIoU per corruption is bold [60] © 2020 IEEE.

(AC) has an mIoU of 77.9 %. Using Dense Prediction Cell (DPC) instead of ASPP leads to the highest mIoU on clean data, *i.e.* 78.8 %. Removing the long-range link (LRL) between encoder and decoder leads to an mIoU of 77.9 %. Using global average pooling yields 78.6 %.

We show the average CD within every corruption category, such as blur, in Figure 3.12 (bars above 100 % are less robust than the reference).

Effect of ASPP. Removing the Atrous Spatial Pyramid Pooling module reduces mIoU significantly (Table 3.3 first column).

Effect of AC. AC are generally increasing the robustness against image blur for many network backbones (especially for XC-71 and ResNets, as qualitatively illustrated in Figure 3.13). When no atrous convolutions are used, the mIoU of defocus blur decreases by 3.8 % for ResNet-101 (CD = 109 %). This effect can be explained as follows: Image blur decreases an image's high-frequent content, resulting in consecutive pixels storing a similar signal. When atrous convolutions are applied, the information contained within a convolution filter is higher since direct neighbors with similar signals are skipped.

Also, for XC-71 and ResNets, AC enhances the robustness w.r.t. image noise as qualitatively illustrated in Figure 3.14. The mIoU scores for Gaussian noise (severity level 1) are 12.2 % (XC-71), 10.8 % (ResNet-101), 8.0 % (ResNet-50) less than respective reference.

For some backbones, AC also increases the robustness with respect to geometric distortion. For MN-V2 and ResNets, the averaged mIoU can decrease by 4.2 % ($\text{CD}^{\text{ResNet-50}}$= 109 %, $\text{CD}^{\text{ResNet-101}}$= 114 %, $\text{CD}^{\text{MN-V2}}$= 106 %).

To sum up, AC usually increase and rarely decrease the robustness for many network backbones for the Cityscapes dataset.

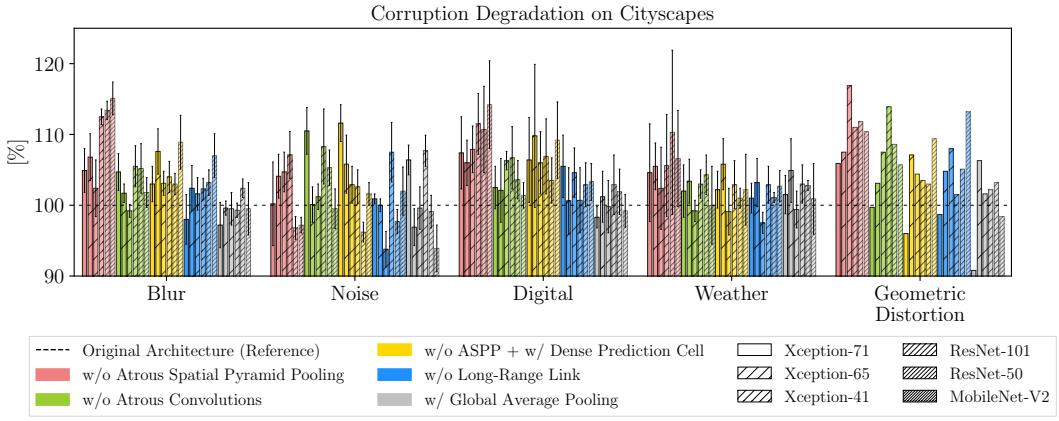
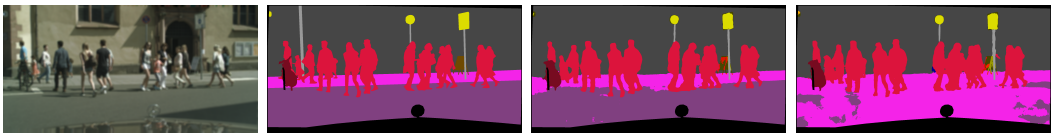


Figure 3.12: CD evaluated on the Cityscapes dataset for the proposed ablated variants of the DeepLabv3+ architecture with respect to image corruptions, employing six different network backbones. Bars above 100 % represent decreased model robustness than the respective reference architecture. Each ablated/modified architecture is re-trained on the original training dataset. Removing ASPP reduces the model performance significantly. Atrous convolutions increase robustness against blur. The model becomes vulnerable against most effects when Dense Prediction Cell is used. Each bar is the average CD of a corruption category, except for geometric distortion (error bars indicate the standard deviation) [60] © 2020 IEEE.

Effect of DPC. The network is clearly vulnerable against many image corruptions, even though this ablated architecture (*i.e.*, w\DPC) is the best-performing model on clean data for XC-71. The CD for defocus blur on XC-65 and MN-V2 are 110 % and 113 %, respectively. The mIoU decreases by 4.1 % and 6.8 %. With respect to XC-71, the CD for noise lie between 109 % and 115 % and the corresponding rCD between 110 % and 128 %. This ablated candidate's mIoU is the lowest for almost every noise type (Table 3.3). A similar outcome results for other corruptions and network backbones.

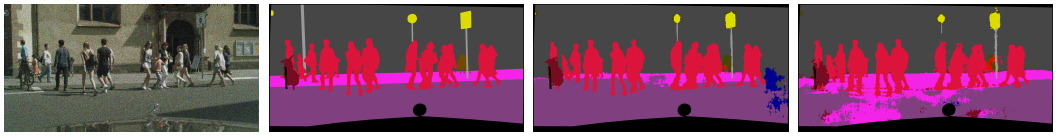
The DPC is created by a neural-architecture-search (NAS, *e.g.* [136, 135, 93]) with the objective to maximize the mIoU on original data. Our result implies that such architectures overfit on this objective, *i.e.*, on clean, original image data. Therefore, conducting a NAS on corrupted images could reveal robust architectures–similar to [22] with respect to the robustness against adversarial examples.

We further find that DPC could learn less multi-scale features than the ASPP, which increase the robustness against common image corruptions. Geirhos *et al.* [36] show that classification models are more robust to common corruption if the shape bias of a model is increased. Whereas ASPP processes its input in parallel by three atrous convolution (AC) layers with large symmetric rates (6, 12, 18), DPC firstly processes the input by a single AC layer with a small rate (1×6) (see Figure 2.15). When a model with DPC is tested on



(a) corrupted image (b) ground truth (c) prediction of ref. model (d) prediction w/o AC

Figure 3.13: Predictions of reference architecture and the ablated variant without atrous convolutions (AC) of Cityscapes [20], which is especially vulnerable to blur.



(a) corrupted image (b) ground truth (c) prediction of ref. model (d) prediction w/o AC

Figure 3.14: Predictions of both the reference architecture and the ablated architecture without atrous convolutions (AC) of Cityscapes [20], which is then especially vulnerable to image noise.

corrupted data, it cannot rely on the same beneficial multi-scale cues (due to the comparable small atrous convolution with the rate 1×6) as ASPP and may, therefore, perform worse.

Effect of LRL. Removing the long-range link (LRL) in a ResNet-101 causes the model to be vulnerable to image noise; it is especially prone to intensity noise (CD = 116 %). With respect to XC-71, the CDs of digital corruptions such as *brightness* are high (*e.g.*, $\text{CD}^{\text{XC-71}} = 111\,\%$).

An MN-V2 without LRL decreases robustness against defocus blur and geometric distortion since the mIoU reduces by 5.1 % (CD = 110 %) and 4.6 % (CD = 113 %), respectively.

Effect of GAP. For most Xceptions, global average pooling enhances the robustness. It is worth noting that the XC-71 (ResNet-101) are particularly prone to image noise. The respective CD scores lie between 103 % and 109 % (106 % and 112 %). The ResNet-101 behaves similarly. The mIoU increases when applied in MobileNet-V2.

PASCAL VOC 2012

The results of the ablation study for PASCAL VOC 2012, presented in this section, differs from the previous results for the Cityscapes dataset, as shown in Figure 3.15, Table 3.4, 3.5. PASCAL VOC 2012 is less challenging than Cityscapes, and hence the performance of the ablated variant is similar. MobileNet-V2 is not evaluated in this section since this model performs poorly.

Effect of ASPP. Removing the multi-scale extraction module reduces, similar to Cityscapes,

		Blur				Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
XC-71	76.7	56.5	59.1	40.2	59.5	56.6	57.8	57.6	63.2	69.9	72.1	57.1	72.6	68.1	43.9	60.9	66.1	50.9	**73.6**
w/o ASPP	70.5	48.3	49.2	33.3	50.1	47.5	47.1	48.2	54.6	62.7	65.1	48.8	65.6	60.2	37.0	53.4	57.3	44.1	69.3
w/o AC	75.7	55.9	58.8	**41.8**	59.0	57.1	58.2	57.3	62.6	69.5	71.0	56.9	71.4	67.6	41.9	60.9	64.1	48.2	73.0
w/o ASPP + w/ DPC	76.8	53.5	54.6	35.8	55.4	55.5	55.6	54.7	60.5	69.0	71.4	54.0	71.0	66.3	42.5	58.3	63.3	49.7	73.3
w/o LRL	76.3	56.4	56.4	40.5	55.9	**59.9**	**59.3**	**60.2**	**64.6**	71.1	72.1	53.0	72.3	67.7	43.8	59.3	64.1	50.8	73.2
w/ GAP	**77.7**	**57.8**	**58.7**	38.3	**59.1**	58.8	55.2	58.4	63.7	**71.9**	**73.8**	**60.4**	**73.9**	**69.2**	**46.9**	**61.6**	**67.9**	**53.5**	73.5

Table 3.4: Mean-IoU averaged over severity levels for clean and corrupted variants of PASCAL VOC 2012 validation dataset for Xception-71 and five corresponding architectural ablations. The standard deviation for non-deterministic corruptions is 0.3 or less. Highest mIoU per corruption is bold.

		Blur				Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
ResNet-50	69.6	38.7	43.5	31.1	45.5	43.2	40.7	44.2	50.9	59.8	63.5	50.3	63.8	58.2	31.3	47.0	56.9	39.8	67.2
ResNet-101	70.3	45.8	45.6	33.2	46.6	49.4	48.3	50.1	55.4	61.3	64.5	50.6	65.3	59.7	31.4	50.4	57.6	41.2	67.6
Xception-41	75.5	52.9	54.7	35.5	53.9	55.8	53.3	56.7	62.8	67.6	70.8	51.9	70.9	64.6	42.5	59.0	63.1	48.4	73.0
Xception-65	76.5	53.5	58.3	37.7	57.2	56.6	54.7	57.4	62.5	69.3	71.8	55.9	72.1	66.7	40.2	58.5	64.0	47.5	73.6
Xception-71	**76.7**	**56.5**	**59.1**	**40.2**	**59.5**	**56.6**	**57.8**	**57.6**	**63.2**	**69.9**	**72.1**	**57.1**	**72.6**	**68.1**	**43.9**	**60.9**	**66.1**	**50.9**	**73.6**

Table 3.5: Average mIoU for clean and corrupted variants of the PASCAL VOC 2012 validation set for several network backbones of the DeepLabv3+ architecture. Every mIoU is averaged over all available severity levels, except for corruptions of category noise where only the first three (of five) severity levels are considered. Highest mIoU per corruption is bold.

the model performance considerably.

Effect of AC. Contrary to Cityscapes, AC have no positive effect against blur, which can be explained due to the differences between PASCAL VOC 2012 and Cityscapes. With respect to Cityscapes, a network without dilated convolutions oversees spatially small classes, or classes, which are far distant from the camera (see Figure 3.8). Such scenarios are not or hardly present in PASCAL VOC 2012.

However, AC also increases the mIoU for many networks with respect to geometric distortion, and regarding XC-41 and ResNet-101, the robustness increases against image noise.

Effect of DPC. A network with DPC is also on this dataset vulnerable to many image corruptions. The CD scores rise from XC-41 to XC-71. The DPC's corrupting influence is particularly pronounced for XC-71 since the total average CD score is 106 % (rCD = 117 %). The neural-architecture-search, which created the DPC, might have been conducted using

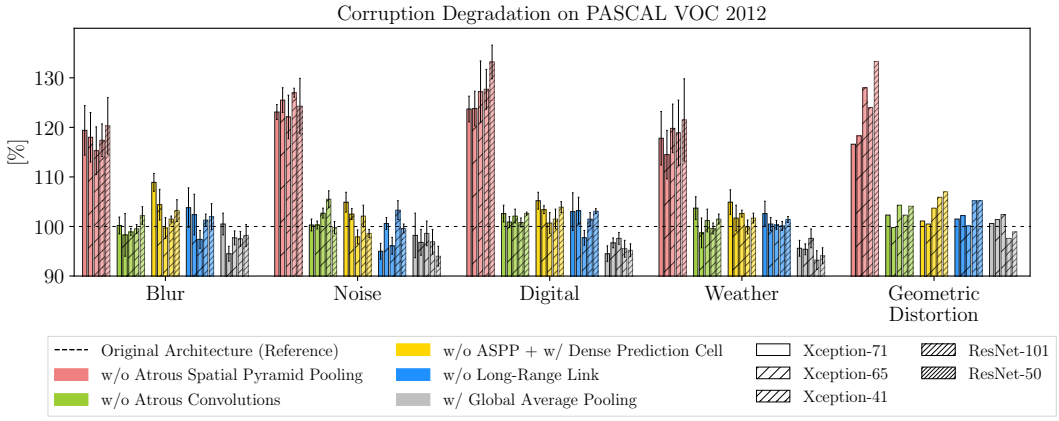


Figure 3.15: CD evaluated on PASCAL VOC 2012 for the several ablated variants of the DeepLabv3+ architecture with respect to image corruptions, for five different network backbones. Each bar except for geometric distortion is averaged within a corruption category (error bars indicate the standard deviation). Bars above 100 % represent a decrease in model robustness compared to the respective reference architecture. Each ablated architecture is re-trained on the original training dataset. Removing ASPP reduces the model performance significantly. AC and LRL decrease robustness against corruptions of category digital slightly. Xception-71 is prone to many corruptions when DPC is used. GAP increases performance against most corruptions. Each network backbone performs further best on clean data when GAP is used.

the XC-71, resulting in an enhanced overfitting effect.

Effect of LRL. An XC-71 and XC-41 without LRL increases the model's robustness towards image noise, likely due to early representation's discarding since the degree of image noise is higher in early CNN layers. The shallow XC-41 is robust to noise for PASCAL VOC 2012 and Cityscapes since less corrupted encoder and decoder representations are combined. Hence, for deeper networks, such as the ResNet-101, this behavior does not hold. However, XC-65 behaves differently: As mentioned in section 3.4.2, XC-65 is on this dataset as well the most robust network backbone against image noise. The LRL shows w.r.t. ResNets the most effect for geometrically distorted images.

Effect of GAP. The overall network robustness increases particularly significant for each backbone when global average pooling is used. The performance on clean data increases by maximum 2.2 %. GAP's effectiveness towards this dataset can probably be explained due to differences to the Cityscapes dataset and ADE20K: GAP averages 2048 activation maps with the spatial dimension 33×33 in our training setup. Both datasets have a significantly higher number of classes and instances per image than PASCAL VOC 2012, and moreover, these are spatially way more distributed. Applying GAP on ADE20K or Cityscapes might

discard too many representations due to the pooling operation.

ADE20K

		Blur				Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
XC-71	42.4	**24.4**	26.4	19.5	23.9	**24.0**	**20.3**	**23.3**	**27.5**	**36.8**	**37.2**	**25.3**	**35.7**	**34.7**	16.1	**29.4**	31.3	19.8	**37.1**
w/o ASPP	40.6	21.9	24.2	17.5	21.9	20.8	16.6	20.0	24.2	34.0	34.8	22.5	33.1	32.4	12.9	26.3	28.9	16.5	35.2
w/o AC	41.8	24.3	25.4	**19.6**	23.6	24.0	19.9	22.9	26.2	35.7	36.1	23.2	34.8	33.7	15.7	28.3	29.9	19.7	35.8
w/o ASPP + w/ DPC	**42.5**	23.3	25.9	18.4	23.1	23.4	18.9	22.4	26.5	36.3	36.5	24.1	34.9	34.2	15.8	28.3	30.6	18.6	36.3
w/o LRL	42.2	22.9	25.9	18.7	23.5	21.7	18.7	21.1	25.4	35.5	36.3	24.3	34.5	34.0	15.1	28.6	30.6	19.7	36.4
w/ GAP	42.0	24.0	**26.6**	19.1	**24.0**	23.6	19.8	22.8	26.7	35.9	37.0	25.0	35.2	34.6	**16.7**	29.3	**31.6**	**20.9**	36.3

Table 3.6: Mean-IoU averaged over severity levels for clean and corrupted variants of ADE20K validation dataset for Xception-71 and five corresponding architectural ablations. The standard deviation for non-deterministic corruptions is 0.3 or less. Highest mIoU per corruption is bold.

		Blur				Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MobileNet-V2	33.1	16.1	16.6	14.9	16.5	12.1	11.5	12.4	17.0	24.7	27.2	14.8	26.5	25.1	7.8	18.5	20.1	10.7	28.3
ResNet-50	37.4	18.0	19.7	16.9	19.2	14.1	12.8	14.4	19.4	28.5	31.1	18.0	30.1	29.5	8.8	21.5	23.9	13.6	32.9
ResNet-101	38.1	19.1	20.6	17.3	19.8	15.4	14.6	15.7	20.7	28.8	31.6	19.7	31.2	31.4	10.2	22.9	25.6	14.0	32.8
Xception-41	39.7	22.1	22.7	17.4	20.8	20.8	18.1	20.5	24.8	33.7	34.2	20.9	32.5	32.6	13.0	25.0	28.4	17.0	34.4
Xception-65	41.4	23.4	25.2	18.9	22.7	23.2	19.8	22.9	27.1	35.4	36.1	23.5	34.8	34.2	14.8	27.7	30.0	18.4	35.6
Xception-71	**42.4**	**24.4**	**26.4**	**19.5**	**23.9**	**24.0**	**20.3**	**23.3**	**27.5**	**36.8**	**37.2**	**25.3**	**35.7**	**34.7**	**16.1**	**29.4**	**31.3**	**19.8**	**37.1**

Table 3.7: Average mIoU for clean and corrupted variants of the ADE20K validation set for several network backbones of the DeepLabv3+ architecture. Highest mIoU per corruption is bold.

The mIoU for clean data is between 33.1 % (MN-V2) and 42.5 % (XC-71 with DPC) as listed in Table 3.6 and 3.7. Many Xceptions (ResNets) are the best-performing networks with respect to clean, original data when DPC (GAP) is used. Figure 3.16 illustrates that the mean CD of every ablated candidate is often close to 100.0 %, which indicate in general, that architectural design choices affect the individual model performance less when the overall mIoU is poor.

Effect of ASPP. The ASPP can affect model performance also on this dataset considerably.

Effect of AC. Several network backbones without AC are slightly more vulnerable to digital and weather corruptions than the reference.

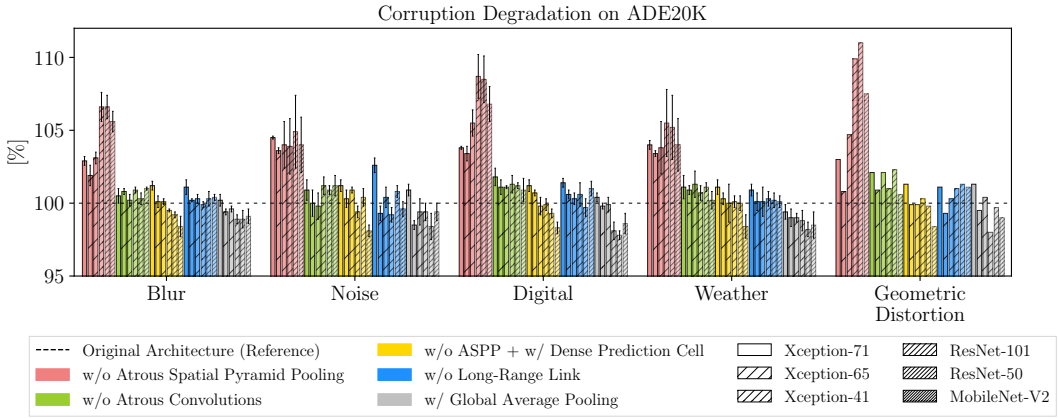


Figure 3.16: CD evaluated on ADE20K for several ablated variants of the DeepLabv3+ architecture with respect to image corruptions for six different network backbones. Each bar except for geometric distortion is averaged within a corruption category (error bars indicate the standard deviation). Bars above 100 % represent a relative decrease in model robustness compared to the respective reference architecture. Compared to the other datasets, the effect of architectural ablations on ADE20K is significantly less pronounced. Each ablated architecture is re-trained on the original training dataset. Removing ASPP decreases performance oftentimes. AC increase performance slightly against most corruptions. DPC and LRL hamper the performance for Xception-71 with respect to several image corruptions. GAP increases the robustness for most backbones against many image corruptions.

Effect of DPC. Like for the Cityscapes dataset and PASCAL VOC 2012, a model using DPC is mostly vulnerable to image corruptions, especially the XC-71.

Effect of LRL. The backbones (especially XC-71) without LRL influences the performance against image noise.

Effect of GAP. The CD of a network with GAP is often below 100 %, indicating that they are slightly more robust than the reference.

Contrary to Cityscapes and PASCAL VOC 2012, the convolutional networks applied for ADE20K perform significantly worse. The results for ADE20K indicate that the potential influence of an architectural property takes effect at higher performances.

3.4.4 Quantitative Results with Respect to Common Corruptions

Figure 3.17 illustrates the mIoU evaluated on the datasets for individual severity levels. The Figure shows the degrading performance with increasing severity level for some candidates of category blur, noise, digital, and weather of a reference model and all corresponding archi-

tectural ablations. Please note that severity level 0 corresponds to clean data. **First row:** Xception-71 evaluated on the Cityscapes dataset for defocus blur, speckle noise, contrast, and spatter. **Second row:** ResNet-101 evaluated on PASCAL VOC 2012 for motion blur, shot noise, JPEG, and snow. **Third row:** Xception-41 evaluated on ADE20K for Gaussian blur, intensity noise, brightness, and fog. The ablated variant without ASPP has the lowest mIoU, in most cases. However, it performs best for speckle noise for severity level 3 and above. The mIoU of the ablated variant without AC is relatively low for defocus blur and contrast. The mIoU of the ablated variant without ASPP and with DPC is relatively low for speckle noise, shot noise (for severity level 4 and 5), spatter. The mIoU of the ablated variant without LRL is relatively high for speckle noise and shot noise. The mIoU of the ablated variant with GAP is high for PASCAL VOC 2012 on clean data and low for speckle noise.

3.4.5 Generalization Behavior of Semantic Segmentation Models

Generalization for Single Image Corruptions

In the previous sections, we evaluated semantic segmentation models' robustness when networks are trained solely on clean data and tested on corrupted data. For the following experiments, we add corrupted data to the clean training sets and evaluate model performance.

We trained DeepLabv3+ with the ResNet-50 backbone and added a corrupted variant of each image corruptions category (*i.e.*, blur, noise, digital, and weather). This results in four trainings where, compared to a "regular" training, the number of training data is doubled. Please see Figure 3.18 for results. Each plot shows the performance degradation for increasing severity level, for either a model trained on clean data only (dashed lines) or both clean and corrupted data (solid lines). The last image corruption in each legend serves as training data, marked by an asterisk, and the scalar value indicates its severity level.

Study on blur. The performance on clean data decreases by 2.6 % when image data corrupted by **Gaussian blur** is added to the clean training data. Mean-IoU further increases for the remaining types of blur. The performance is especially high for defocus blur, probably due to similarity to Gaussian blur.

Study on image noise. The performance on clean data decreases by 1.9 % when image noise is added to the training data. Interestingly, the model can generalize quite well to a wide range of noise models (similar to [102] for full-image classification). The model performs well for severity level 4 and 5 of speckle noise, though it is trained on severity level 3. The signal-to-noise ratio of severity level 5 is more than 3 dB less than of severity level 3, which corresponds to doubling the degree of noise for that severity level. Whereas the mIoU for Gaussian, impulse, and shot noise is already below 10 % for severity level 2, when the model

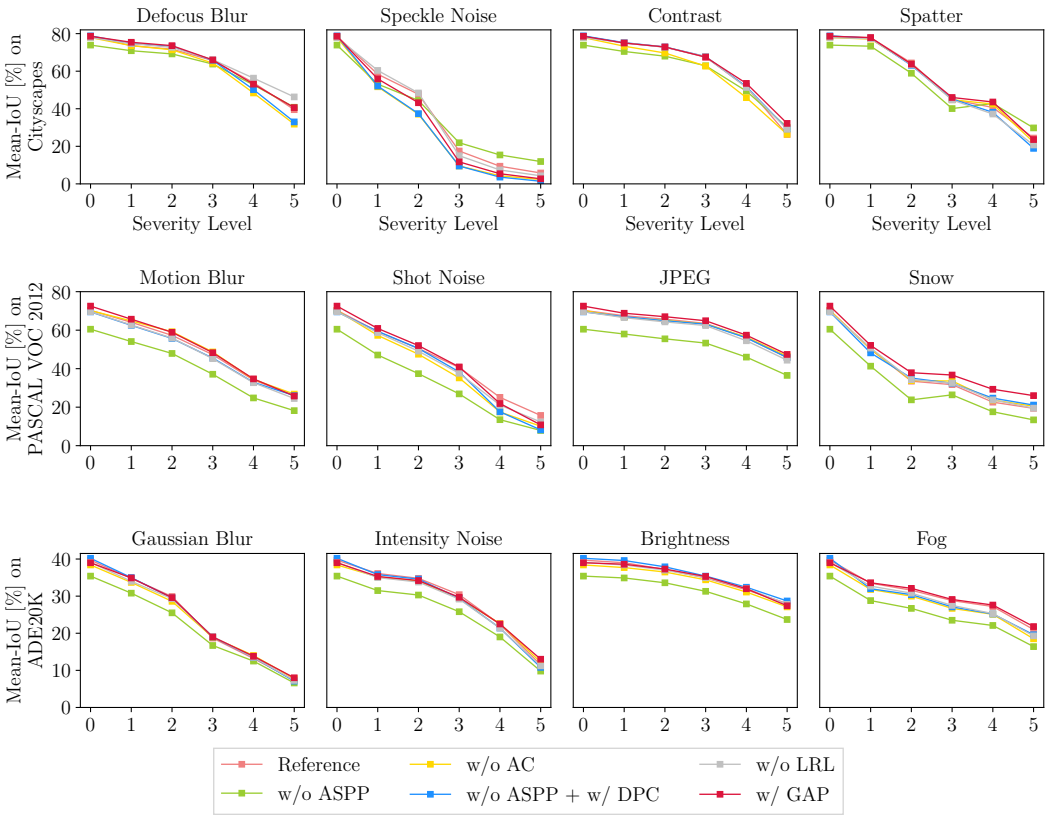


Figure 3.17: Mean-IoU for many candidates with respect to the image corruption categories: blur (**first column**), noise (**second column**), digital (**third column**), and weather (**fourth column**) for a reference model and all corresponding architectural ablated variants, evaluated for every severity levels on Cityscapes, PASCAL VOC 2012, and ADE20K. The standard deviation for non-deterministic corruptions is 0.3 or less.

is trained on clean data only, it is significantly increased for the model that is trained on image noise. The model performance decreases significantly for higher severity levels for image noise types that are not part of the training data.

Study on digital corruptions. The performance on clean data increases slightly by 0.4 % when image corruption **saturate** is added to the training data. The mIoU increases– besides "saturate"– only for "brightness" compared to the model trained only on clean data. The image corruptions of this category are quite diverse. "Brightness" and "saturate" have a contrary effect as "contrast". The "JPEG compression", on the other hand, posterizes large areas of the image. The non-similarity of these effects is also indicated by the large error bars in Figure 3.12.

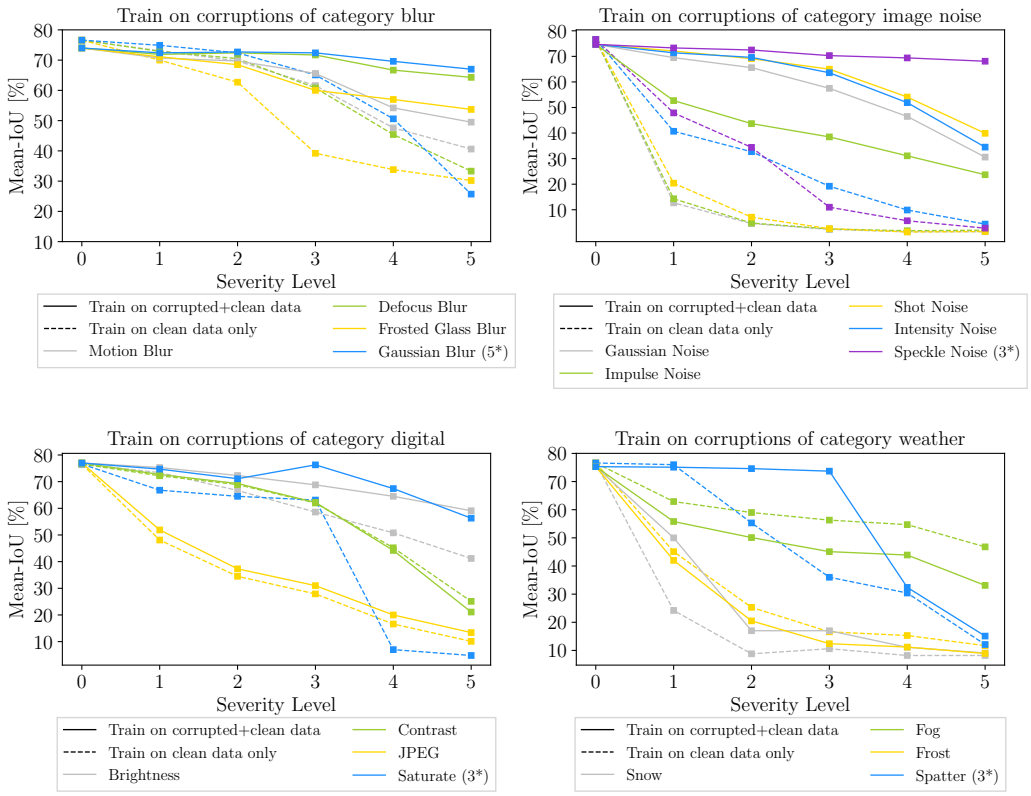


Figure 3.18: Mean-IoU when corrupted data is added to the clean training set. We train four models of DeepLabv3+ using the ResNet-50 backbone and added a corrupted variant of each image corruption category (*i.e.*, blur, noise, digital, and weather). Each plot shows the performance degradation for increasing severity level, for either a model trained on clean data only **(dashed lines)** or clean and corrupted data **(solid lines)**. Please note that severity level 0 corresponds to mean-IoU for the clean data. The standard deviation for non-deterministic corruptions is 0.3 or less. The last image corruption in each legend serves as training data, marked by an asterisk, and the scalar value indicates its severity level. When the model is trained on corruptions of category blur, noise, and digital, it can generalize to unseen types of respective image corruptions. The model generalizes significantly well to a wide range of noise models. The model is not able to perform well on every unseen image corruption of category digital.

Study on weather corruptions. The performance on clean data decreases by 1.9 % when image corruption **spatter** is added to the training data. Contrary to image noise, the model cannot generalize to a more severe degree of the corrupted data the model is trained on

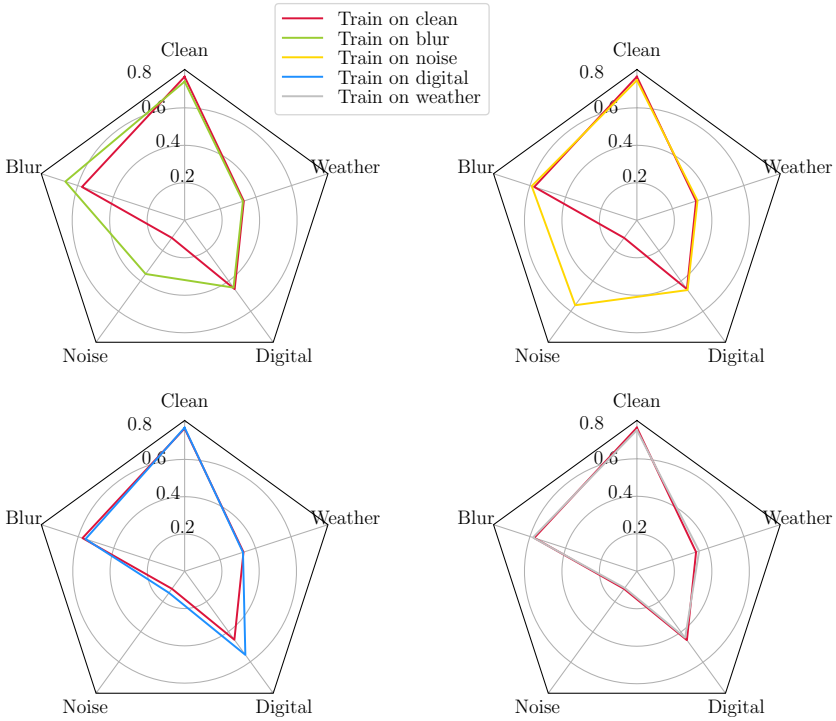


Figure 3.19: Mean-IoU for clean data and the four image corruption categories blur, noise, digital, and weather averaged over severity levels. Each radar plot illustrates the averaged mIoU of a trained model on clean data only and one that is also trained on one image corruption category. Models trained on noise, digital, or weather corruptions increase the performance in general solely for the respective image corruption category. A model that is trained on blur, however, increases the performance also on image noise significantly.

(*i.e.*, its performance on the fourth and fifth severity level of "spatter" is hardly increased). The mIoU for image corruption "snow" increases significantly for the first severity level. Interestingly, this model does not generalize with respect to "fog" and "frost" and performs even worse than the reference model, which is trained only on clean data.

We previously discussed the model performance restricted within an image corruption category. In our final evaluation, we present the mIoU (averaged over severity levels) of the remaining image corruption categories (see Figure 3.19). Please note that the results in this Figure are based on the same experiments as conducted for Figure 3.18. Interestingly, when blurred data is added to the clean training data, the model increases mIoU for noisy data also. When image noise is added to the clean training data, the performance of the remaining image corruption categories is barely affected. Similar results can be observed

when images of category digital are added to the clean training data. For image corruptions of category weather, the average mIoU is only slightly increased when we train the model on this particular category.

Generalization for Multiple Image Corruptions

This subsection constitutes content which is not present in [61]. In the previous section, we discussed the generalization capabilities of semantic segmentation networks when the training data is doubled, *i.e.*, one corrupted training set is added to the clean data. In this section, we want to answer the generalization capability question if we add all the corrupted data to the training data.

The experimental setup for the following experiments is that the training set consists, besides of clean data, of all image corruption candidates of the previous section. **Thus, the training set consists of the clean data and Gaussian blur, speckle noise, saturate and spatter of varying severity levels.**

Results w.r.t. clean data. For the results in Figure 3.20, we trained on the first, third, and fifth severity levels of the previously mentioned training set and, also, each setup is trained for varying durations in terms of training iterations.

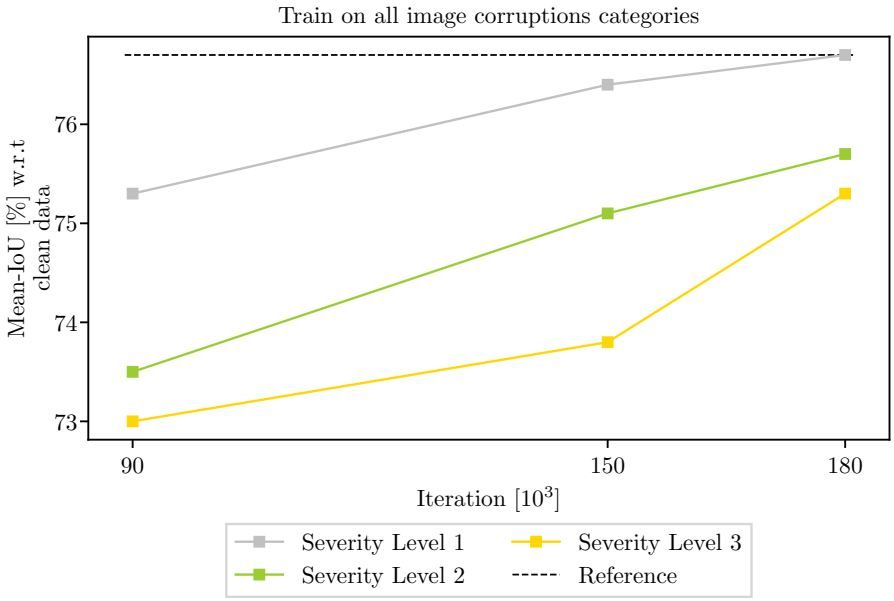


Figure 3.20: Mean-IoU for clean (*i.e.*, the original, non-corrupted) data of a ResNet-50 trained with clean and corrupted data consisting of Gaussian blur, speckle noise, saturate, and spatter, which is trained with varying severity levels and training iterations.

The following main messages can be derived from Figure 3.20.

- The reference model, *i.e.*, the model which is regularly trained on clean data, reaches its maximum performance (mIoU equals almost 77 %) after 90,000 iterations
- Adding corrupted data to the training set decreases the performance on clean data
- This decrease is stronger for increasing severity of the corrupted data
- For severity level 1 (silver) a comparable performance on clean data is reached when the training duration is doubled
- For severity level 3 and 5 (green, gold) a comparable performance on clean data requires more than the doubled training duration to reach a comparable performance on clean data

Results w.r.t. corrupted data. We now discuss the generalization performance for corrupted data in Figure 3.21. The following main messages can be derived from these results:

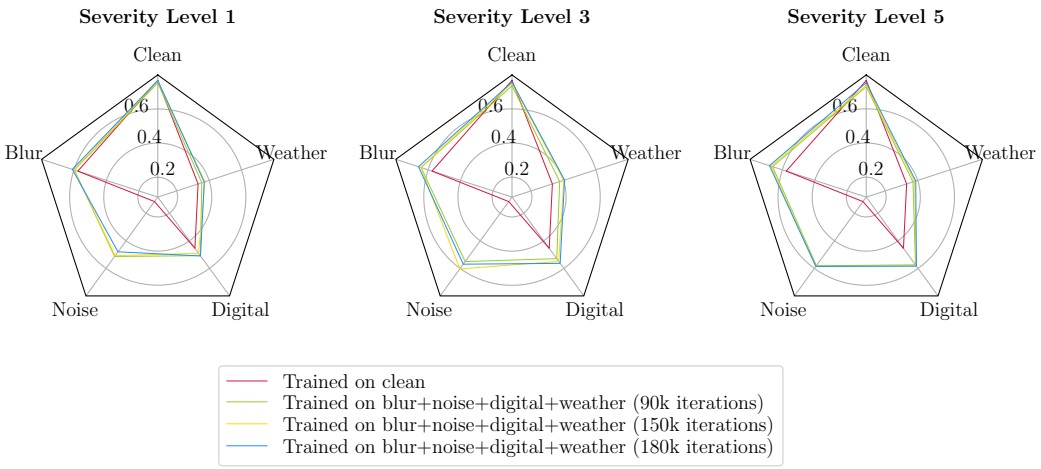


Figure 3.21: Mean-IoU for corrupted data of a ResNet-50 trained with clean and corrupted data consisting of Gaussian blur, speckle noise, saturate, and spatter, which is trained with varying severity levels and training iterations.

- Training for more iterations does, unlike the performance on clean data, not necessarily increase the performance on corrupted data. For severity level 1 (left radar chart), the average mIoU for image blur is equal for each training durations. With respect to image noise, the performance does even decrease. A similar trend can be observed for severity level 5 (right radar chart), wherewith increasing training duration, basically the performance on clean data increases
- Training using the most severe corruption does not necessarily yield the highest performance against corrupted data. Whereas this assumption holds for image blur (mIoU

increases by approx. 8 % when severity level 5 instead of severity level 1 is used for training), the performance for noise and weather is highest when trained using severity level 3

- Training using a medium level of corruption severity (middle radar plot) yields the best result in this experiment
- Surprisingly, the performance w.r.t. image noise, when trained for 180k iterations, is often less than trained for 150k iterations

Influence of Realistic Image Corruptions

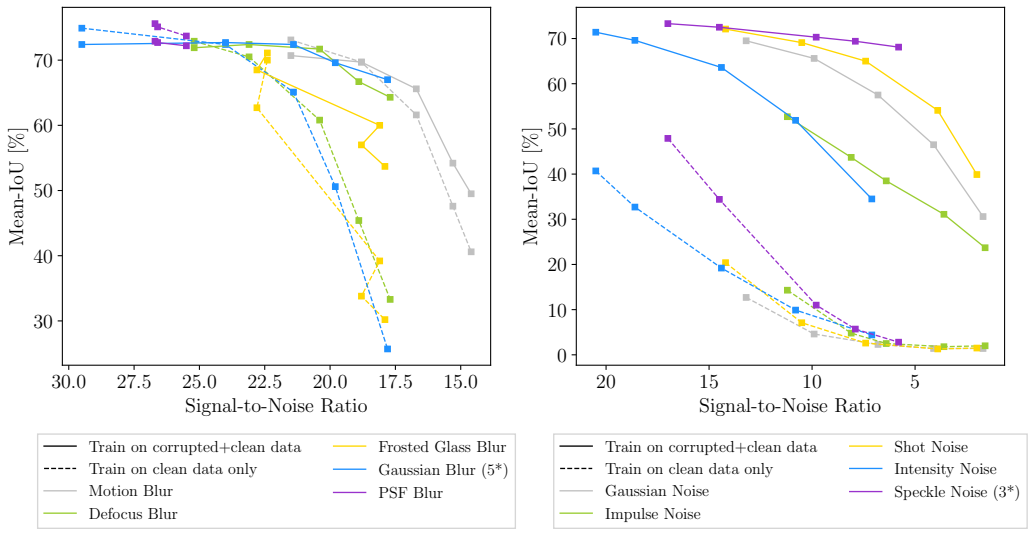


Figure 3.22: Model performance when corrupted data is added to the training set. We train four models of DeepLabv3+ using the ResNet-50 backbone and add a corrupted variant of blur and image noise. To make the image corruptions mutually more comparable, each abscissa corresponds to the averaged Signal-to-Noise ratio of the respective image corruption. The models are trained on Gaussian blur (severity level 5, left) or speckle noise (severity level 3, right). The standard deviation for non-deterministic corruptions is 0.3 or less.

This section focuses on evaluating model robustness with respect to our proposed, more realistic image corruptions. Figure 3.22 shows the model performance of the ResNet-50 again when corrupted data is added to the training set. To make severity levels mutually comparable, we average their Signal-to-Noise ratio (SNR) in this Figure over the validation set, *i.e.* each abscissa represents the averaged SNR of a severity level.[1]

PSF blur. We observe that our modeled PSF blur (purple, Figure 3.22 left) is in terms

[1]Contrary to Figure 3.18, where each abscissa corresponds to a severity level in terms of integer values.

of SNR by considerably less severe than the remaining image corruptions of category blur. The mIoU with respect to PSF blur of the first two severity levels is considerably higher than for the remaining types of blur with a similar SNR (*i.e.*, severity level 1 of defocus blur and motion blur), which is observed not only for the ResNet-50 (as illustrated in this Figure) backbone but also for all remaining backbones.

These results indicate that a CNN might learn, to a certain extent, real PSF blur, which is inherently present in the training data. The fact that the mIoU with respect to PSF blur and Gaussian blur (*i.e.*, the weakest blurs regarding their SNR) decreases when Gaussian blur is added to the training data might also support this hypothesis. However, the performance quickly degrades similarly to an mIoU score comparable to the remaining blur types.

Intensity noise. The model performs significantly worse for our proposed noise model than for speckle noise when the model is trained only with clean data (purple, Figure 3.22 right, dashed lines). The model's mIoU converges to a common value for each image corruption of category noise. When noisy data is added to the training data, the model performs clearly superior to this particular image corruption. The mIoU of the fifth severity level of speckle noise and third severity level of impulse noise has a similar SNR, but the mIoU differs by approximately 30 %.

This result shows that semantic segmentation models generalize on image noise since a clear mIoU increase is observable; however, it depends strongly on the particular type of image noise. Based on this result, the poor performance with respect to our proposed intensity noise (blue line) indicates that training a model with unrealistic image noise models is not a reasonable choice for increasing model robustness towards real-world image noise.

Geometric distortion. As stated in section 3.4.2, the model performance with respect to geometric distortion is comparable among the benchmarked architectures (see the last column of Table 3.2). In general, the GSCNN is the most robust network against geometric distortion. The mIoU of GSCNN decreases for the first severity level by less than 1 %. The Xception-based backbones are for the DeepLabv3+ architecture, the best-performing networks.

3.5 Conclusions

This chapter covered an exhaustive, comprehensive evaluation of both established and state-of-the-art semantic segmentation networks towards common real-world image corruptions for three semantic segmentation datasets. This study reports various findings of the robustness of specific architectural choices and semantic segmentation models' generalization behavior. These findings are useful for practitioners to design the right model for their task at hand, where the types of image corruptions are often known. Based on the study, we are able to conclusively present robust model design rules, which are likely helpful in aiding the

development of robust semantic segmentation networks.

3.5.1 Robust Model Design Rules

Network backbones and architectures. Regarding DeepLabv3+, Xception-41 has, in most cases, the best price-performance ratio. It performs primarily with respect to Cityscapes and ADE20K close to Xception-71 (the most robust network backbone overall), for a similar performance on clean data but approximately 50 % less storage space and less complex architecture. Xception-based backbones are generally more robust than ResNets; however, this difference is low for a less severe degree of image corruption. MobileNet-V2 is vulnerable to most image corruptions, also for a low severity, however, it is capable of handling blurred data well. For non-DeepLab-based models, the GSCNN, a model that incorporates shape information, is overall robust against most weather and digital corruptions and geometrically distorted input but is also extraordinarily vulnerable against image noise.

Atrous Spatial Pyramid Pooling. The ASPP is essential for radially distorted images. Removing ASPP decreases the mIoU, especially for PASCAL VOC 2012, considerably. The relative decrease, when no ASPP is used, is less for the remaining datasets.

Atrous convolutions. On Cityscapes, atrous convolutions should generally be used since they increase robustness towards many real-world corruptions. Atrous convolutions increase the robustness against image blur and noise for many network backbones for such a dataset. With respect to ADE20K, similar tendencies can be observed.

Dense Prediction Cell. Models using the DPC instead of ASPP are vulnerable to many types of image corruptions throughout the datasets, mostly for image noise. Hence, this vulnerability should be considered in applications, such as low-light scenarios, where image noise may be considerably high.

Long-Range Link. The previously discussed results indicate that more shallow networks as Xception-41 and ResNet-50 are more robust to corruptions of category image noise, and we recommend hence to omit an LRL for these networks if the respective application comes along with image noise.

Global average pooling. We recommend always apply global average pooling on PASCAL VOC 2012, as the network performance and robustness is often increased. For Cityscapes, utilizing GAP in Xception-71 is clearly vulnerable to image noise.

3.5.2 Generalization

We have presented a wealth of experiments regarding the generalization capability of semantic segmentation networks. Whereas the models can generally perform well on many types of blur and image noise when the respective corruption is part of the training data, we cannot observe any explicit generalization of the categories digital and weather. The pronounced

diversity of these image corruption categories is probably responsible for this. When trained on a single corruption category, the dominant increase in model performance is generally within that particular category. However, when trained on image blur, the performance on image noise increases also significantly. Contrary to image classification results, the performance for categories blur, digital, and weather barely increases when trained using image noise.

Training a model on multiple image corruptions simultaneously leads to a more general increase in performance against corrupted images. However, this increase is less than training on single, particular, image corruptions. It is also important to note that the performance on clean data can decrease significantly, and the training time needs to be doubled to maintain a comparable performance on clean data. Whereas the previous results clearly show that more extended training with multiple corrupted data increases performance on clean data, the same does not hold for corrupted data.

When a model is trained with a more realistic type of blur and image noise, the results indicate that convolutional networks learn the inherently present corruptions of an image. Additionally, we observe that a trained model on a simple noise model is not the best choice for increasing the robustness towards real-world image noise.

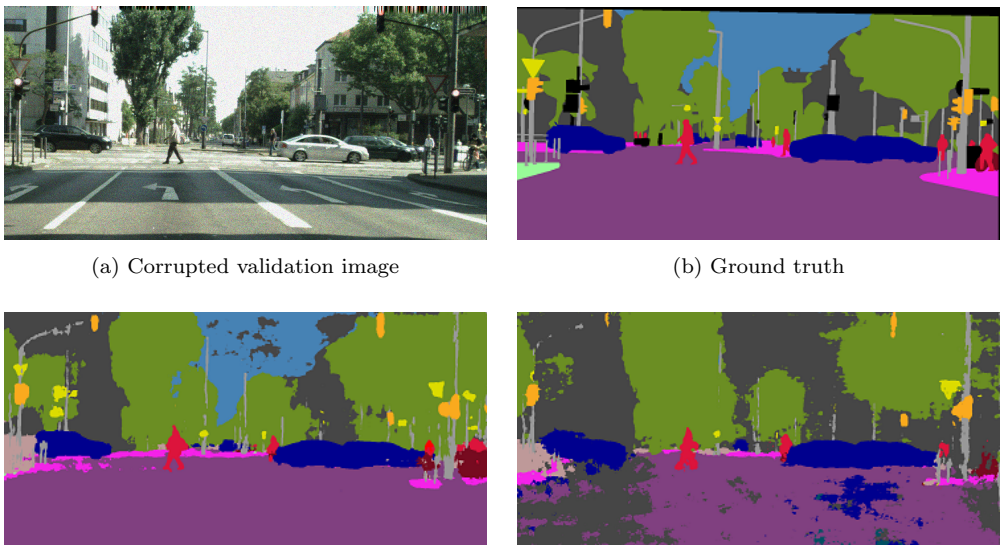
4

Painting-by-Numbers: A Robustness-Increasing Data Augmentation

The content associated with this chapter has been published in [62]. In more detail: Besides the following introduction, the entire substantial content of [62] and/or its (electronic) supplementary material [64] is associated with this chapter. Besides this chapter, content of [62, 64] are in the thesis' abstract, introduction (chapter 1) and conclusion (chapter 5).

The previous chapter's extensive benchmark shows that convolutional networks for semantic segmentation are vulnerable to various common image corruptions, especially against image noise. Real-world image corruptions cannot be avoided in safety-critical applications: Environmental influences, such as adverse weather conditions, may corrupt the image quality significantly. Foggy weather decreases the image contrast, and low-light scenarios may exhibit image noise. Fast-moving objects or camera motion causes image blur. Such influences cannot be entirely suppressed by sensing technology, and CNNs must hence be robust against common image corruptions. Obviously a CNN should also be robust towards adversarial perturbations (e.g., [109, 57, 18, 42, 10, 83, 9]). In this chapter, we focus on the robustness increase of such convolutional networks. We build upon an insight from image classification that network robustness can be improved by increasing the convolutional network's bias towards object shapes. In more detail, we propose a simple (see a code implementation in A.3), yet effective, data augmentation strategy that increases this shape bias through decreasing the amount of texture in the training data. The basic idea is to alpha-blend some training images with a texture-free representation of the scene. The texture-based appearance of a training image is less reliable, forcing the network to develop additional

shape-based cues for the segmentation. In this way, our schema does not require additional training data, as we directly use the available semantic segmentation ground-truth. We refer to our training schema as "Painting-by-Numbers", and we present it in detail in the following sections. Painting-by-Numbers increases the mIoU on for images with noise up to 40 %, even though the model is not trained on image noise. Figure 4.1 shows the effectiveness of Painting-by-Numbers qualitatively with respect to image noise.



(a) Corrupted validation image

(b) Ground truth

(c) Prediction with our training schema

(d) Prediction with standard training schema

Figure 4.1: Results of the ResNet-50 trained with Painting-by-Numbers. (a) An image crop of the Cityscapes [20] validation set is corrupted by shot noise. (c) Prediction of a model that is trained with Painting-by-Numbers. (d) Prediction of the same model with reference, regular training schema. The prediction with our training schema (c) is clearly superior to the prediction of the reference training schema (d), though our model is not trained with image noise. In particular, pixels of the classes *road*, *traffic signs*, *cars*, *persons* and *poles*, are more accurately predicted.

4.1 Related Work

The research interest in increasing the robustness of CNN models with respect to common image corruptions grows. Most methods have been proposed for the task of full-image classification. Mahajan et al. [81] and Xie et al. [117] show that using more training data increases the robustness. The same result is found when more complex network backbones are used, also for object detection [84]. Hendrycks et al. [49] show that using adversarial

logit pairing [65] increases the robustness not only for adversarial perturbations but also for common perturbations. Zhang [127] increases the robustness against shifted input. The authors of [130, 72] increase model robustness through stability training methods. Zhang *et al.* [126] propose a training framework referred to as "auxiliary training" where the weights of the image classifier which is regularly trained on clean data, are aligned with the weights of a classifier that is trained on corrupted data.

Cohen *et al.* [19] increase model robustness through test-time adjustments. Several other works apply data augmentation techniques to increase generalization performance. Whereas some work occludes parts of images [131, 26], crops, replaces and mixes several images [121, 125, 111], or apply various (learned) sets of distortions [50, 21], other methods augment with artificial noise to increase robustness [38, 80, 102].

Geirhos et al. [36] show that humans and convolutional networks seem to classify images with different strategies. Unlike humans, convolutional neural networks trained on ImageNet [25] seem to rely more on local object texture instead of global object shape. The authors transfer the style of human-made paintings on the ImageNet dataset, which degrades the image's textural information content. The resulting dataset is termed "Stylized-ImageNet". The authors increased the robustness of a convolutional network by training the model on Stylized-ImageNet. We applied the style-transfer technique of Geirhos *et al.* to Cityscapes but found the resulting images to be quite noisy (see Figure 4.2). Therefore, training on such data might increase robustness not solely due to increased shape bias but rather due to increased image corruption. We aim to find a training schema that does not have any image corruption added.

(a) Original data (b) Stylized data (c) Zoom of (a) (d) Zoom of (b)

Figure 4.2: Illustration of the style transfer technique of [36]. An original training image (a) of the Cityscapes dataset [20] is stylized by a painting (b). Images (c) and (d) show the image content of the red rectangle of (a), where (d) is clearly noisier compared to the original data (c).

4.2 Training Schema: Painting-by-Numbers

Our goal is to generically increase the robustness of semantic segmentation models for common image corruptions. Here, *robustness* refers to training a model on clean data and

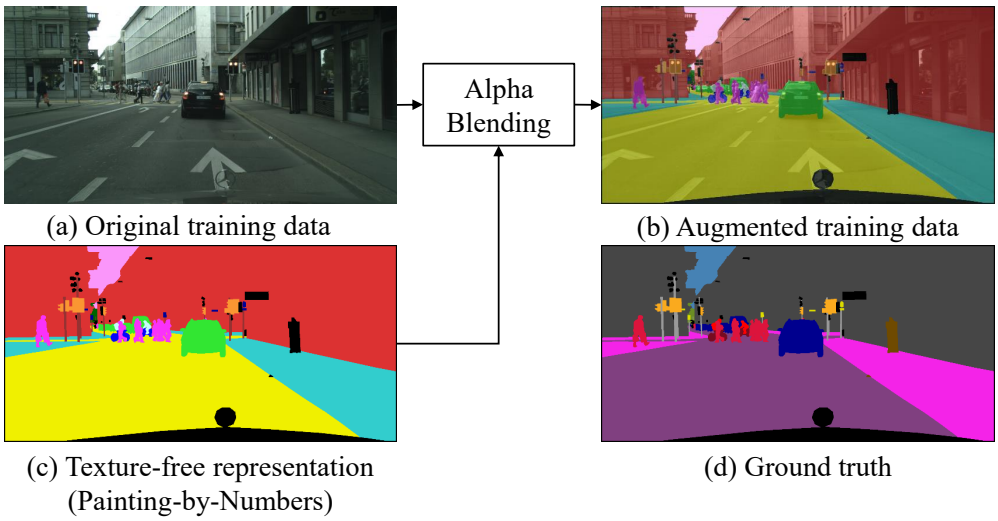


Figure 4.3: Overview of our training strategy, which we refer to as Painting-by-Numbers. (a) An RGB image of the Cityscapes training set [20] and the respective ground-truth in (d). We paint the numbers, i.e., ground-truth labels of (d) randomly, leading to a texture-free representation as exemplarily shown in (c). Painting the numbers **randomly** is essential since these colors are not likely to appear in real imagery. The final training image **(b)** is then generated by alpha-blending (a) with (c). A fraction of the training data is augmented as in b). Using this data hence as training data increases the robustness against common corruptions.

subsequently validating it on corrupted data. Simply adding corrupted data to the training set does certainly increase the robustness against common corruptions. In the previous chapter, we have seen that this approach comes along with drawbacks: Firstly, a significantly increased training time. Secondly, the possibility to overfit to specific image corruptions [37, 114] and reduced performance on clean data [114]. Thirdly, it further may be hard to actually identify all sources of corruption for new test scenarios. For our training schema, we build on the finding of [36]. We propose an augmentation schema (Painting-by-Numbers) that modifies the training process so that the model develops shape-based cues for the decision of how to segment a pixel, resulting in a generic increase of model robustness.

The basis of our schema is that we treat the segmentation ground-truth as a texture-free representation of the original training data (see Figure 4.3). We then colorize (or *paint*) the ground-truth labels (or *numbers*) randomly (Painting-by-Numbers) to generate a representation as shown in Figure 4.3 (c). We uniformly sample the color from the sRGB color gamut with range $[0, 255]$, similar to the images of the Cityscapes dataset. Painting the numbers randomly is essential since these colors are not likely to appear in real imagery.

Finally, we alpha-blend this this representation with the original training image, according to eq. 4.1 (we motivate the blending step shortly),

$$I_{blended} = \alpha \times I_{painting-by-numbers} + (1 - \alpha) \times I_{original} \tag{4.1}$$

where $I_{blended}$ is the resulting alpha-blended training image (Figure 4.3 b), α is the blend parameter (where $\alpha = 1$ corresponds to a representation where original training input is entirely blended), $I_{painting-by-numbers}$ is the texture-free representation (Figure 4.3 c) and $I_{original}$ is the original training image (Figure 4.3 a). In this image, we use $\alpha = 0.7$. When a convolutional network is trained on such data, it is forced to develop (or increase) its shape-bias since we actively corrupt the textural content of the image. Therefore, an image's texture features are less reliable, and a network needs to develop additional cues to segment pixels correctly. Painting-by-Numbers is computationally efficient. We implemented Painting-by-Numbers on CPU using TensorFlow and augmented half of the images of each mini-batch on-the-fly. For our setup (a machine with 4x 1080 Ti (11 GB), and Intel Xeon CPU E5-2699 with 44 CPU cores with 2.2 GHz), the training time with Painting-by-Numbers increases by approx. 2.5 %. When implemented on GPU, a network trained with Painting-by-Numbers is approx. 2.0 % slower than training without Painting-by-Numbers.

Motivation blending with $0 < \alpha < 1$. When a convolutional network is trained

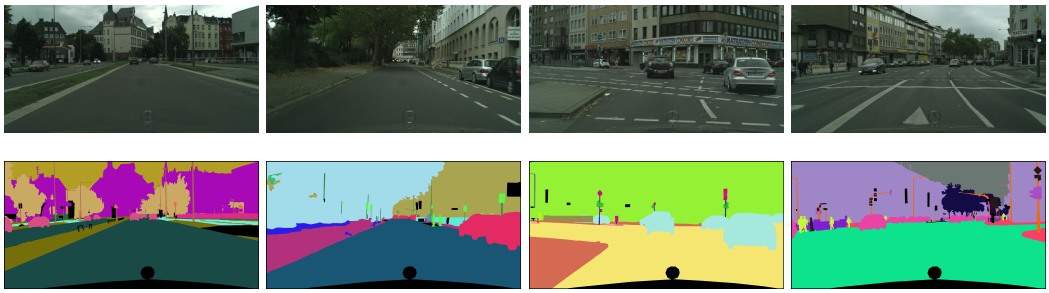


Figure 4.4: Illustration of randomized, texture-free training data of the Cityscapes dataset [20]. **(top)** Original examples of Cityscapes' training data. **(bottom)** Respective texture-free variants.

solely on texture-free images (*i.e.*, on images as shown in the second row of Figure 4.4) meaning that the blend parameter α is fixed to 1, the model performs decently on the texture-free validation set of Cityscapes, as qualitatively shown in Figure 4.5. The IoU for classes *road, sidewalk, car, persons, and traffic signs* is high and hence segmented well. This is a positive signal because it means that the model is able to learn from entirely texture-free training data.

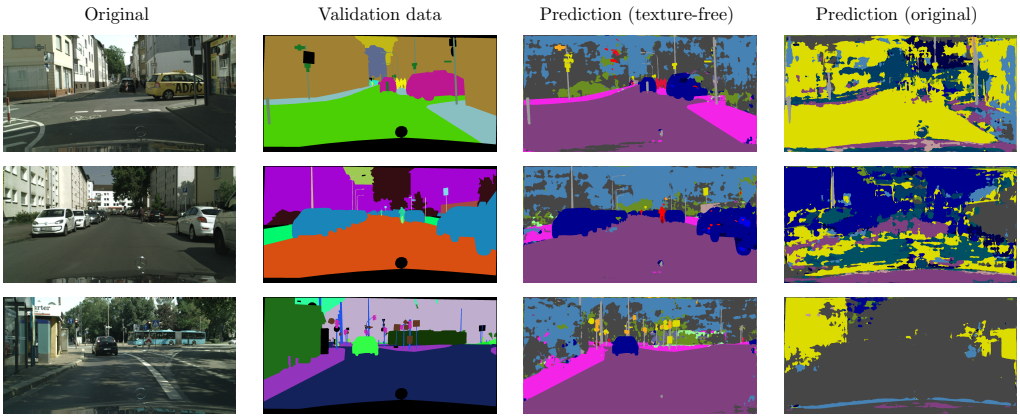


Figure 4.5: Qualitative results of a ResNet-50 trained solely on texture-free data of the Cityscapes dataset [20]. Each row shows the original validation data **(first column, for the ease of reference)**, the texture-free validation data **(second column)**, and a model that is trained on such texture-free data only **(third column)**. Even though the model (trained on texture-free data, third column) is struggling for the class building and sky, it is able to segment road, sidewalk, cars, persons, poles, and traffic signs well. The model cannot utilize any textural cues since it is solely trained on texture-free images. This result indicates that a convolutional network can generally learn from entirely texture-free data through utilizing other cues such as shape (*e.g.*, cars and persons have a distinct shape), context (*e.g.*, cars are on top of a road), or layout (*e.g.*, sidewalk are oblong stripes near image edges). For comparison, the **fourth column** shows a model's prediction regularly trained on original data, showing clearly, that it cannot handle texture-free image data.

When we augmented only half of a training batch, instead of every image, (α is still fixed to 1), the performance on both the original validation set and texture-free validation set was, again, considerably high; However, the robustness of the new model with respect to common image corruptions was not increased. We hypothesize that such a model learns to predict well for two different domains: the original data and the texture-free data. This motivates us to choose $\alpha < 1$ for some training images. As we will see, with a varying degree of alpha-blending, the robustness of the model towards common image corruptions increases significantly and, at the same time, keeps a consistently good performance on clean data.

Training protocol. We implement Painting-by-Numbers in the DeepLabv3+ architecture, and we demonstrate the effectiveness of Painting-by-Numbers for several network backbones: MobileNet-V2, ResNet-50, ResNet-101, Xception-41, Xception-71. We augment exactly the half of a batch by our Painting-by-Numbers approach and leave the remaining half unchanged. This ensures to keep a similar performance on clean data, comparable to a network that is trained regularly on clean data only. We train the DeepLabv3+ us-

ing the authors suggested training protocol: crop size 513×513, initial learning rate 0.01, "poly" [77] learning rate schedule, using the Atrous Spatial Pyramid Pooling (ASPP) module, fine-tuning batch normalization [58] parameters, output stride 16.[1] The only data augmentation techniques applied are random scale data augmentation and random flipping during the network training. We use no global average pooling and train every model with TensorFlow [1].

Evaluation protocol. We evaluate trained networks on the image corruptions of ImageNet-C (see section 3.2), *i.e.*, several types of blur (Gaussian, motion, defocus, frosted glass), image noise (Gaussian, impulse, shot, speckle), weather (snow, spatter, fog, frost), and digital transformations (JPEG, brightness, contrast). We evaluate the mean-IoU to rate semantic segmentation performance on full-size images.

4.3 Experiments and Results

In this section, we demonstrate the effectiveness of Painting-by-Numbers. We discuss implementation details in section 4.3.1. We then show the results on the Cityscapes dataset in section 4.3.2. As mentioned previously, we argue that the increase against common corruptions is based on an increased shape-bias of the respective network. We conduct a series of experiments to validate this assumption in the next section.

4.3.1 Implementation Details

We experiment with varying implementations and augmentation schemes, such as varying parameters for alpha-blending or image augmentation strategies.

Parameters for alpha-blending. Our experiments show that a fixed value for α does not yield the best results. Instead, we use two parameters for alpha-blending, α_{min} and α_{max}. These values define an interval from which α is drawn. They are the essential hyperparameters needed to achieve the best results towards common image corruptions. If α_{min} is too low, i.e., the amount of texture in the image is high, the robustness increase for common corruptions is minor. If α_{min} is too high, i.e., the amount of texture in the image is further diminished, the robustness decreases with respect to common corruptions (as discussed previously). We observe that the models only connect learned features from the two domains (original data domain and alpha-blended data domain) if the latter's texture is present, i.e., $0 < \alpha < 1$. Values of around $\alpha_{min} = 0.5$ and $0.9 < \alpha_{max} < 1.0$ have proven to be a reasonable initial choice.

Mini-batch augmentation scheme. We experiment with two types of augmenting the images of a mini-batch. In one scheme, we use an augmentation probability, aug_{prob}, de-

[1] Due to hardware limitations we were not able to train on the suggested crop size of 769.

(a) Instances are painted randomly

(b) Instances are painted by RGB-mean

(c) Instances are painted by RGB-median

Figure 4.6: Examples of several coloring schemes used for Painting-by-Numbers applied to an image of Cityscapes [20].

ciding for each image individually if Painting-by-Numbers augment it. In the other scheme, we always augment precisely the half of a batch by Painting-By-Numbers, where the only parameters to be optimized are α_{min} and α_{max}. Both types of batch augmentation increase the robustness against common corruptions.

Incorporating instance labels. The Cityscapes datasets contain instance labels for several classes (besides the "regular" semantic segmentation ground truth). These instance labels can be used to paint each class-instance separately with Painting-by-Numbers, as illustrated in Figure 4.6 (a), which increases robustness with respect to common image corruptions as well. We target with Painting-by-Numbers the semantic segmentation task and the more general semantic labels which are available for many reference datasets.

Paint with mean and median RGB. We further paint the images with a more consistent color, such as the mean and median RGB value of the class or instance (instead of painting the semantic classes randomly), as illustrated in Figure 4.6 (b) and (c). This approach does, as expected, not increase the model robustness. Instead of forcing a model to not rely on texture and color appearance, by corrupting these very properties, the network learns to assign a mean and median value to classes and instances, contrary to the effect of random painting. Hence, there is no need to increase the shape-bias for predicting the segmentation map when the colors are likely to appear in real imagery.

Best Setup. We train MobileNet-V2, ResNet-50, ResNet-101, Xception-41, and Xception-71 with Painting-by-Numbers. We evaluate the models on Gaussian noise to select the hyperparameters. For ResNet-50, and Xception-41, we observe best results when we draw α uniformly from the interval $\alpha_{min} = 0.70$ and $\alpha_{max} = 0.99$. For the remaining networks, we observe the best results when we draw α from the interval between $\alpha_{min} = 0.50$ and $\alpha_{max} = 0.99$. In every training, we augment the half mini-batch of each training iteration.

4.3.2 Results on the Cityscapes Dataset

We refer to a trained network with standard training schema as the reference model (i.e., trained on clean data only) and a model trained with Painting-by-Numbers as our model

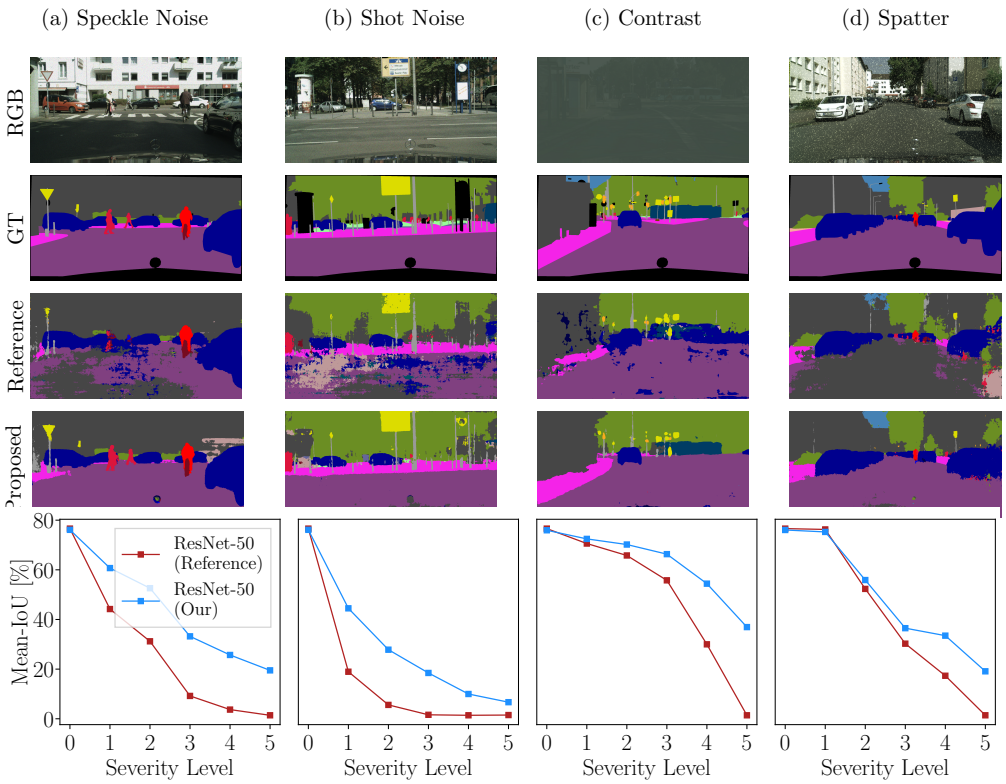


Figure 4.7: **(top)** Qualitative results by the ResNet-50 backbone on four corrupted images of the Cityscapes validation dataset [20] for both the reference model and our model (i.e., trained with Painting-by-Numbers). **(bottom)** Quantitative results with respect to the corrupted variants of the full Cityscapes dataset. All image corruptions are parameterized with five severity levels, where severity level 0 corresponds to clean (i.e., original) data. Whereas with respect to clean data, the mean-IoU of both models is roughly the same, we see that our model is clearly superior for all types of noise added. The standard deviation for non-deterministic corruptions is 0.3 or less. For consistent performance on clean data, the mean-IoU on corrupted data increases when the model is trained with Painting-by-Numbers. For the first severity level of shot noise, the mIoU of our model is even higher by 25 %.

in the following sections. Figure 4.7 shows both qualitative and quantitative results on corrupted variants of the Cityscapes dataset when the DeepLabv3+ using ResNet-50 as network backbone is trained with both training schemes. Every image corruption is parameterized into five severity levels, where severity level 0 corresponds to the clean data.

The reference model is struggling to predict well in the presence of image corruptions (Figure 4.7 top). It segments extensive parts of *road* wrongly as *building* for both image corruptions *spatter* and *image noise*. When the same model is trained with Painting-by-

Numbers, the predictions are clearly superior. With respect to quantitative results (Figure 4.7 bottom), our model performs significantly better for image corruptions of category *speckle noise*, *shot noise*, and *contrast*. The image corruption *contrast* decreases the contrast of the full image, corrupting hence the textural image content strongly. A network that is able to rely also on shape-based cues for image segmentation is hence a well-performing model for such an image corruption. The mean-IoU on *spatter* is for both models comparable for the first severity level, but it is for our model higher by almost 15 % for the fourth severity level.

The results with respect to the remaining image corruptions for the Cityscapes dataset are listed in Table 4.1. We show the effectiveness of Painting-by-Numbers besides ResNet-50 also for MobileNet-V2, ResNet-101, Xception-41, and Xception-71. In the first column, we report the performance on clean data, i.e., the original Cityscapes validation set. The mIoU evaluated on several types of image corruptions is listed accordingly. Each value is the average for up to five severity levels. We reported for both clean and corrupted data, the results of the reference model and our model.

		Blur				**Noise**				**Digital**				**Weather**			
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost
Reference																	
MobileNet-V2	**73.0**	**52.4**	**47.0**	**44.7**	**48.1**	9.6	14.2	9.8	25.6	50.4	43.8	32.5	**20.3**	10.8	43.3	47.7	16.1
ResNet-50	**76.6**	57.1	**55.2**	45.3	**56.5**	10.7	13.4	12.1	37.7	59.8	52.7	41.7	**23.4**	**12.9**	39.8	56.2	19.0
ResNet-101	76.0	**58.9**	**55.3**	47.8	**56.3**	22.9	22.9	23.1	45.5	57.7	56.8	41.6	**32.5**	**11.9**	45.5	55.8	23.2
Xception-41	77.8	61.6	**54.9**	51.0	**54.7**	27.9	28.4	27.2	53.5	63.6	56.9	51.7	**38.5**	18.2	46.6	57.6	20.6
Xception-71	77.9	62.5	**58.5**	**52.6**	**57.7**	22.0	11.5	21.6	48.7	67.0	57.2	45.7	**36.1**	16.0	48.0	63.9	20.5
Painting by Numbers																	
MobileNet-V2	72.2	49.5	41.4	40.7	43.0	**17.4**	**18.4**	**16.8**	**35.7**	**62.5**	**50.8**	**51.0**	17.6	**12.1**	**46.9**	**56.5**	**22.4**
ResNet-50	76.1	**58.1**	53.5	**50.3**	55.1	**35.7**	**34.3**	**36.1**	**56.7**	**68.8**	**64.2**	**60.5**	21.3	10.6	**46.1**	**61.0**	**22.9**
ResNet-101	**76.3**	58.1	54.2	**48.7**	54.7	**41.6**	**44.3**	**40.6**	**57.4**	**70.5**	**64.4**	**65.0**	25.6	10.8	**50.1**	**56.9**	**28.0**
Xception-41	**78.5**	**65.5**	54.2	**51.1**	51.8	**46.9**	**44.9**	**46.9**	**64.3**	**73.4**	**60.2**	**68.8**	15.7	**19.3**	**55.8**	**65.7**	**28.2**
Xception-71	**78.6**	**63.0**	53.6	48.6	52.2	**35.5**	**38.4**	**34.2**	**57.6**	**74.9**	**63.9**	**69.1**	22.2	**18.2**	**57.4**	**65.4**	**25.5**

Table 4.1: Performance overview evaluated on the Cityscapes dataset. Each entry shows the mean-IoU of several corrupted variants of the Cityscapes dataset. Every image corruption is parameterized into five severity levels, and the resulting mean-IoU is averaged. The standard deviation for non-deterministic corruptions is 0.3 or less. For image noise-based corruptions, we excluded every severity level whose signal-to-noise ratio less than 10. The higher mIoU of either the reference model or the respective model trained with Painting-by-Numbers is bold. Overall, we see most (74 %) bold numbers for the Painting-by-Numbers models.

Performance with respect to clean data. Even though we paint the exact half of the training data and train both models for the same amount of iterations, the performance on clean data is often barely affected, and for Xception-based backbones even increased.

Performance with respect to image blur. The performance of our model with respect to image blur does not notably increase. Painting-by-Numbers does not increase the performance for this type of image corruption because image blur corrupts the object shapes by smearing the object boundaries. Hence, our learned shape-bias does not work well for this type of corruption.

Performance on image noise. Painting-by-Numbers increases the robustness with respect to image noise most effectively (see Figures above). This is especially intriguing as Painting-by-Numbers is contrary to the regular approach of including noisy data to increase robustness towards image noise. The results show clearly that the performance for every type of image noise is considerably increased. For example, the absolute mIoU of Xception-41 for Gaussian noise, impulse noise, shot noise, and speckle noise increases by 19.0 %, 16.5 %, 19.7 %, and 11.0 %, respectively. Classes with distinct shapes as *cars*, *persons*, or *poles* are more accurately segmented than by the reference.

Performance with respect to digital corruptions. A network trained with Painting-by-Numbers increases significantly the mIoU against the corruptions *brightness*, *contrast*, and *saturation*–but not JPEG artifacts. The reason is that *JPEG compression* corrupts the boundary of objects and incorporates new boundaries through posterization artifacts (see Figure 4.8). Our network cannot hence profit from its increased shape-bias. The Figure illustrates how the JPEG compression algorithm posterizes larger areas of the *car* and *road*. This is somewhat contrary to Painting-by-Numbers: Whereas our training schema (i.e., Painting-by-Numbers) alpha-blends the image with a homogeneous, texture-free representation of a class, the JPEG compression causes new, non-distinct shapes within a class instance, see Figure 4.8 b.

(a) JPEG compressed validation image (b) Zoom of (a)

Figure 4.8: A validation image of Cityscapes [20] corrupted by *JPEG compression* in (a) and a respective zoom in (b). The crop in (b) visualizes the posterization effect of *JPEG compression*. Whereas Painting-by-Numbers alpha-blends the image with a homogeneous, texture-free representation of a class, the JPEG compression causes new, non-distinct shapes within a class.

Performance on weather corruptions. Xception-71 and Xception-41 increases the performance with respect to *spatter* by 9.4 % and 9.2 %, respectively. Xception-41 further increases the mIoU against *frost* by 7.6 %. Every model increases the performance against *fog*. We cannot observe a significant performance increase for *snow*. Even though our model's higher mIoU with respect to common image corruptions of category weather is less than, for example, for image noise, the predictions of a network trained with Painting-by-Numbers are improved for key-classes of category "things" such as *cars*, and *persons* significantly. Table 4.2 lists the individual IoU score for both the reference model and our model using the ResNet-50 backbone.

With respect to *spatter*, the IoU score for classes *car*, and *person* is significantly higher by 46.6 %, and 31.3 %. Whereas our model struggles with respect to several classes for corruption *snow*, the IoU for "things" as *person*, and *rider* is higher by more than 7.0 %. For image corruption *fog*, our model performs better for almost all classes. Regarding image corruption frost, the IoU score for *cars* of our model is higher by 8.3 %.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Reference (RN-50)																				
Snow	**81.2**	**16.6**	**60.0**	1.4	**3.2**	**21.2**	**15.9**	**34.3**	**40.4**	**20.6**	**70.0**	25.6	5.7	24.4	8.4	12.0	3.6	**3.5**	**35.6**	**25.5**
Spatter	48.5	6.5	55.2	**7.5**	12.5	30.7	**35.1**	30.3	**66.1**	16.3	3.9	19.4	15.1	26.7	1.8	17.9	1.2	14.6	34.6	23.4
Fog	93.2	60.5	79.2	18.6	35.3	**46.0**	40.4	**63.9**	73.8	7.8	77.9	69.2	46.7	85.6	52.3	68.8	47.1	45.3	66.4	56.7
Frost	**54.7**	**12.7**	38.7	1.0	**13.1**	13.6	11.0	38.5	41.8	**12.2**	40.8	21.1	8.1	32.3	7.0	8.6	3.9	0.1	23.6	20.1
Proposed (RN-50)																				
Snow	59.8	4.3	47.5	**1.5**	2.8	10.5	11.5	29.7	26.3	9.8	67.3	**32.7**	**8.4**	**32.1**	**12.0**	**15.7**	**7.3**	1.6	30.1	21.6
Spatter	**79.9**	**24.9**	**69.3**	5.9	**27.1**	**36.8**	33.7	**39.4**	61.3	**24.7**	**42.6**	**50.7**	**23.1**	**73.3**	**20.2**	**30.1**	**7.2**	**19.6**	**50.0**	**37.9**
Fog	**95.7**	**70.5**	**84.8**	**32.3**	**46.6**	44.0	**47.4**	62.9	**84.8**	**33.0**	**87.8**	**70.5**	**50.7**	**90.6**	**62.1**	**79.9**	**68.6**	**51.5**	**67.6**	**64.8**
Frost	51.6	10.9	**49.8**	**2.0**	11.8	**15.4**	**15.4**	**42.5**	**50.9**	8.2	**58.3**	**24.2**	**13.7**	**40.6**	**8.1**	**15.7**	**9.3**	**1.8**	**35.4**	**24.5**

Table 4.2: IoU for each class of several candidates of category *weather* evaluated on the Cityscapes dataset using ResNet-50 backbone. The IoU score of *spatter* of our model for classes *car* and *person* is significantly higher (46.6 % and 31.3 %, respectively) than the IoU of the reference model. Overall, we see most bold numbers for "things" of our Painting-by-Numbers model.

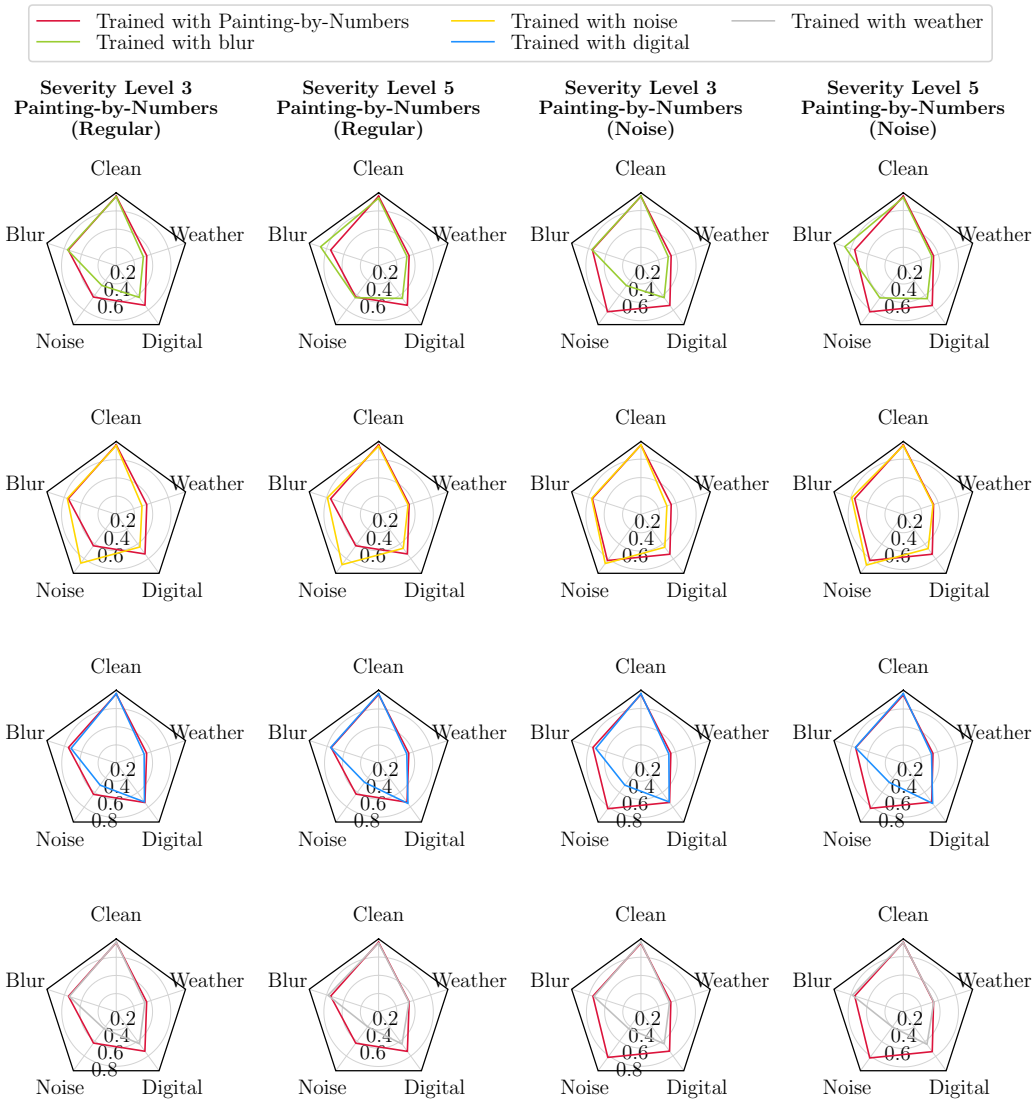


Figure 4.9: Mean-IoU with respect to corrupted data of a ResNet-50 trained with Painting-by-Numbers **(red)** compared with a regular data augmentation, where either Gaussian blur, speckle noise, saturation, or spatter is added to the clean training data. Each network is either trained with severity level 3 of the respective image corruption **(first and third column)** or severity level 5 **(second and fourth column)**. We generally observe the best overall performance for a model trained with Painting-by-Numbers. However, networks that are trained with image noise are clearly superior with respect to the generalization of image noise-based corruptions. The third and fourth columns show the results when Painting-by-Numbers is combined with noise augmentation, combining the best of both worlds.

4.3.3 Comparison to Regular Data Augmentation

In chapter 3, we discussed the generalization capability of semantic segmentation networks when corrupted data is added to the training set. We now compare Painting-by-Numbers to such a regular data augmentation in Figure 4.9.

Each column of Figure 4.9 shows the generalization performance w.r.t. corrupted data, for a ResNet-50 that is either trained with Painting-by-Numbers, or on a particular image corruption of the categories blur, noise, digital, and weather. In the first column and second column, we train each ResNet on the third or fifth severity level and show the results for Painting-by-Numbers' variant, as it is discussed so far in this chapter. The third and fourth columns show the same results except that we added additive Gaussian noise to Painting-by-Numbers, which will be discussed shortly. The main messages with respect to regular Painting-by-Numbers, *i.e.* the first and second column, are:

- With respect to data augmentation using the third severity level (first column), Painting-by-Numbers has clearly the best overall performance
- With respect to data augmentation using the fifth severity level (second column), Painting-by-Numbers still has the best overall performance. However, the data augmentation with blur boosts performance for both image noise and blur (as discussed in the previous chapter)
- The generalization capability of the networks with respect to image noise is clearly superior to Painting-by-Numbers. For high performance on image noise, it appears to be essential to train image noise, as discussed in [102]
- For this reason, we combined Painting-by-Numbers with a simple additive noise augmentation (third and fourth column). Whereas the generic performance with respect to these diverse image corruptions is maintained, the performance on image noise is greatly enhanced and almost comparable to the regular augmentation with noise
- To summarize, combining Painting-by-Numbers with a regular noise augmentation is a simple method to increase the robustness significantly

4.3.4 Combining Dense Prediction Cell and Painting-by-Numbers

In the previous chapter, we demonstrated that a specific neural network module, dense prediction cell (DPC), that performs best on clean data, is extraordinarily vulnerable to corrupted data. With respect to the Cityscapes dataset, especially Xception-based network architectures equipped with DPC are vulnerable against common image corruptions.

This section shows that Painting-by-Numbers increases the robustness of a model with DPC significantly, as illustrated in Figure 4.10. Painting-by-Numbers does, again, not hamper the performance on clean data, and both models are trained for the same amount of

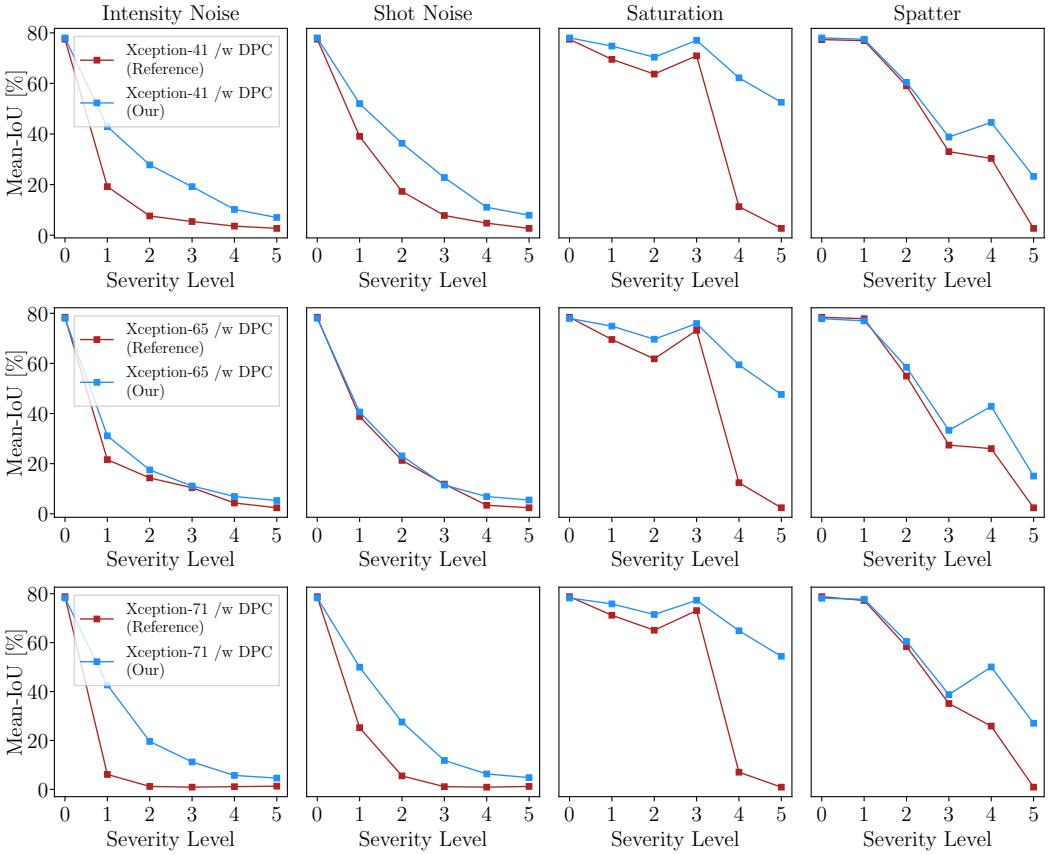


Figure 4.10: Quantitative results of Xception-41 **(first row)**, Xception-65 **(second row)**, and Xception-71 **(third row)** with respect to the corrupted variants of the full Cityscapes validation set. Each model is equipped with the dense prediction cell, *i.e.*, an architectural property that achieves the best performance on clean data but is vulnerable to corrupted data. All image corruptions are parameterized with five severity levels, where severity level 0 corresponds to the clean data. The standard deviation for non-deterministic corruptions is 0.3 or less. Painting-by-Numbers increases the model robustness significantly while keeping similar performance with respect to clean data. The mIoU for Xception-71 increases on the intensity noise by almost 40 %.

iterations. The mIoU increases significantly: for the first severity level of intensity noise by almost 40 %, even though the model has not seen image noise during training.

Table 4.3 shows the detailed qualitative results for the reference models with DPC (top) and when they are trained with Painting-by-Numbers (bottom). Again, we see many more bold numbers in the bottom part of the Table, showing that Painting-by-Numbers significantly increases model performance.

		Blur				Noise				Digital				Weather			
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost
Reference																	
XC-41 w/ DPC	77.5	60.6	**53.0**	**50.8**	**52.5**	27.5	19.2	27.2	53.6	63.6	53.4	46.0	**36.0**	17.6	50.0	56.7	20.6
XC-65 w/ DPC	**78.4**	**63.9**	**59.4**	**53.5**	**58.8**	29.6	21.6	29.8	53.3	65.2	56.5	48.0	**31.5**	18.8	49.4	59.1	20.7
XC-71 w/ DPC	**78.8**	62.8	**59.4**	**52.6**	**58.2**	16.4	6.1	15.5	44.7	64.8	59.4	45.3	**32.0**	14.4	48.6	64.0	20.8
Painting by Numbers																	
XC-41 w/ DPC	**78.0**	**62.7**	49.3	49.2	47.9	**44.3**	**42.9**	**43.5**	**62.2**	**73.7**	**61.6**	**68.1**	20.3	**18.1**	**56.3**	**64.2**	**25.3**
XC-65 w/ DPC	78.0	63.1	51.7	**49.9**	49.4	**32.7**	**31.1**	**31.6**	**55.2**	**73.0**	**57.4**	**67.0**	23.5	14.5	**54.9**	**66.0**	**20.8**
XC-71 w/ DPC	78.3	**63.3**	54.3	51.2	53.5	**41.0**	**42.7**	**38.8**	**60.0**	**74.9**	**60.4**	**69.3**	25.8	**16.3**	**57.3**	**64.6**	**26.9**

Table 4.3: Performance overview of Xception-based network backbones equipped with Dense Prediction Cell evaluated on the Cityscapes dataset. Each entry shows the mean-IoU of several corrupted variants of the Cityscapes dataset. Every image corruption is parameterized into five severity levels, and the resulting mean-IoU is averaged. For image noise-based corruptions, we excluded every severity level whose signal-to-noise ratio less than 10. The standard deviation for non-deterministic corruptions is 0.3 or less. The higher mIoU of either the reference model or the respective model trained with Painting-by-Numbers is bold. Overall, we see the most bold numbers for the Painting-by-Numbers models.

4.4 Understanding Painting-by-Numbers

In this section, we validate that a trained model with Painting-by-Numbers has indeed an increased shape-bias. We conduct a series of experiments to validate this assumption. These are based on the following consideration: Semantic classes with a) no texture at all or b) texture that is strongly corrupted should be more reliably segmented by a model trained with Painting-by-Numbers. We create many variants of the Cityscapes validation set that are corrupted on class-level, as illustrated in Figure 4.11. In (a), we remove the texture of class *cars* and replace it with the dataset-wide RGB-mean of the training set of this particular class. The respective class does, in this way, not contain any texture but homogeneous color information only. In (b) and (c) the classes *building* and *car* are corrupted by a severe additive Gaussian noise and Gaussian blur, respectively. In (d), we replace as in (a) the class texture of an image and, additionally, remove the class background resulting in only the silhouette of a class being present. This experiment allows us to mitigate a possibly present context or background bias. Please note that Figure 4.11 shows only a small set of examples. We applied these corruptions for each class, accumulating to 76 corrupted datasets for these validation experiments.

Networks tested on such images need to segment all semantic classes individually even though they cannot rely on class texture. A network needs to use other cues, as shape-based ones, to classify–especially images corrupted in the way as shown in Figure 4.11 a) and

d)–these images correctly.

(a) Replaced *car* by RGB-mean (b) Corrupted *building* by severe noise (c) Corrupted *car* by severe blur (d) Replaced *car* by RGB-mean and removed background texture

Figure 4.11: Examples of the image data to validate an increased network bias towards object shape when models are trained with Painting-by-Numbers. We evaluate the semantic predictions when a network cannot rely on the class texture by removing or strongly corrupting the Cityscapes validation set's [20] individual class texture. (a) The texture is fully replaced by the dataset-wide RGB-mean value of the respective class. (b) An individual class is corrupted by severe noise. (c) An individual class is corrupted by severe blur. (d) The texture is fully replaced by the dataset-wide RGB-mean value of the respective class as in (a), but, in addition, also the texture of the background is removed, resulting in only the silhouette being present.

Instead of the (mean) Intersection-over-Union, we use the sensitivity s

$$s = \frac{TP}{TP + FN}, \tag{4.2}$$

where TP are true-positives, and FN are false-negatives as evaluation metric for the experiments in this section. The sensitivity is more appropriate for these applications than IoU (see eq. 2.23) since we are solely interested in the class-level segmentation performance. Because all classes but one is clean or not corrupted, false-positively (FP) segmented pixels are of no interest. Utilizing IoU could, especially for classes covering fewer image regions, result in misleading scores, whereas the sensitivity s false-positively segmented pixels are not considered. Table 4.4 shows the experiments' results for validating an increased shape-bias by Paining-by-Numbers, which are generated again with DeepLabv3+ using the ResNet-50 network backbone.

Quantitative results. The results in Tab. 4.4 Like the convention of the previous section, we refer to a network that is trained with the standard training schema as a reference model (i.e., only clean data used) and a model trained with Painting-by-Numbers as our model. The top (bottom) part of the Table contains the sensitivity score for each class of the reference model (our model). Each line of the Table shows the sensitivity of s for the corrupted data described previously and the sensitivity for clean data. The higher sensitivity of a network backbone of either the reference model (top) or our model (bottom) is bold. In the following, we will separately discuss the quantitative results for class categories "stuff"

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Reference																				
Clean	98.8	**93.1**	**96.6**	53.3	**69.6**	**74.5**	**81.7**	**85.7**	**96.7**	**74.1**	**97.8**	**91.5**	**76.9**	**97.6**	85.1	**92.8**	**70.7**	**79.2**	**88.7**	**84.4**
RGB-mean	92.7	21.1	**88.9**	**40.7**	5.4	68.9	12.8	31.5	1.4	**3.3**	**97.7**	73.4	62.5	24.5	13.6	**16.3**	**9.0**	2.9	0.9	35.1
Noise ($scale = 0.5$)	5.8	0.8	**95.0**	0.2	1.7	4.7	6.4	**39.2**	0.1	1.4	2.8	8.1	2.9	4.5	0.0	0.0	**7.1**	0.2	3.4	9.7
Noise ($scale = 1.0$)	2.9	0.0	**94.0**	0.1	1.2	2.0	6.4	**40.2**	0.0	0.5	0.2	4.4	1.2	3.6	0.0	0.0	**3.0**	0.2	2.0	8.5
Blur ($\sigma = 20$)	**94.6**	42.8	**89.3**	**38.0**	1.8	63.6	18.4	19.1	0.6	**7.3**	94.0	55.4	55.5	56.7	32.5	**24.0**	**7.7**	9.0	0.9	37.4
Blur ($\sigma = 30$)	**94.1**	42.1	**89.4**	**33.2**	1.2	62.3	14.0	16.2	0.6	**2.3**	93.8	54.6	51.8	44.2	29.8	**18.6**	**8.1**	4.2	1.1	34.8
Silhouette	2.1	0.0	**93.3**	0.4	0.0	1.0	0.0	0.1	0.0	0.0	**67.4**	0.8	0.0	0.0	0.1	**5.6**	**2.2**	0.0	0.0	9.1
Painting-by-Numbers																				
Clean	**99.0**	90.3	96.3	**56.0**	67.1	68.9	76.5	81.1	96.2	66.3	97.1	89.5	74.0	96.8	**89.7**	86.0	59.6	72.2	87.8	81.6
RGB-mean	**97.9**	**53.8**	51.2	34.2	**14.9**	**79.7**	**38.4**	**40.5**	**1.8**	2.3	97.4	**78.4**	**66.3**	**78.6**	**37.6**	3.5	0.4	**9.1**	**4.6**	**41.6**
Noise ($scale = 0.5$)	**97.4**	**50.9**	92.1	**8.4**	**37.4**	**34.1**	**8.2**	11.1	**23.3**	**30.6**	**32.3**	**50.1**	**19.7**	**49.8**	**31.5**	**1.9**	0.0	**0.3**	**26.7**	**31.9**
Noise ($scale = 1.0$)	**95.9**	**51.7**	91.3	**9.6**	**29.4**	**32.3**	**7.1**	9.9	**12.2**	**27.2**	**33.6**	**52.7**	**21.3**	**40.6**	**25.8**	**1.1**	0.0	**0.4**	**23.3**	**29.8**
Blur ($\sigma = 20$)	49.3	**43.5**	86.5	18.7	**4.7**	**73.6**	**55.1**	**29.8**	**1.0**	0.8	**94.3**	**75.5**	**73.2**	**71.9**	**56.6**	7.9	0.5	**20.2**	**3.5**	**40.3**
Blur ($\sigma = 30$)	46.3	**48.0**	83.2	14.1	**4.5**	**73.6**	**49.7**	**25.4**	**1.0**	0.5	**94.7**	**74.6**	**71.1**	**73.7**	**47.9**	3.8	0.2	**18.2**	**3.8**	**38.6**
Silhouette	**99.1**	**58.4**	18.1	**5.9**	**2.1**	**86.2**	**31.8**	**26.6**	**4.1**	**8.2**	61.9	**79.5**	**1.9**	**77.4**	**0.1**	0.0	0.0	**3.8**	**12.1**	**30.4**

Table 4.4: Sensitivity score s per class for several corrupted variants on the class-level of the Cityscapes datasets. **Clean:** The sensitivity on clean (i.e. original, non-corrupted) data. **RGB-mean:** The texture of a class is replaced with the dataset-wide RGB mean of that class. **Noise:** The texture of a class is corrupted by severe additive Gaussian noise. **Blur:** The texture of a class is corrupted by severe Gaussian blur. **Silhouette:** Similar to RGB-mean, but with an additionally black background. The higher sensitivity score of a network backbone of either the reference (top) or our model (bottom) is bold. Overall, we see the most bold numbers for our Painting-by-Numbers model.

and "things" (please see section 2.3 for an explanation of these terms).

Both networks perform well for classes "stuff" since the amount of textural information is often low, such as for *road*, *wall*, *sidewalk*, and *sky*. The sensitivity of both models differs for *road* by 5.2 %, for *wall* by 6.5 %, and for sky by 0.3 %. Whereas the absolute sensitivity for both models is above 90.0 % for *road* and *sky*, it is less than 41 % for *wall*. Our model performs for *sidewalk* significantly better by 32.7 % ($s_{reference} = 21.1\,\%$, $s_{ours} = 53.8\,\%$).

Painting-by-Numbers performs worse than the reference for classes "stuff" with a large amount of textual information, such as *building*, *vegetation*, and *terrain*. For example, the sensitivity score of our model for *building* is 37.7 % less. We explain this with the fact that classes belonging to "stuff" have no distinct shape. Hence, Painting-by-Numbers cannot aid performance by relying on the (increased) shape-bias. When, additionally, the amount of texture of a class is extensive, our model's sensitivity is less than the reference model.

Interestingly, the reference model can perform well when the texture of the category "things" is replaced by RGB-mean. Its sensitivity for *person* is 73.4 % which is only 5.0 % less than the sensitivity of our model. We observe a similar result for the class *rider* ($s_{reference} = 62.5\,\%$, $s_{ours} = 66.3\,\%$).

However, our model performs often significantly better than the reference model for most

of the remaining "things" such as *car*. The sensitivity score of our model for this class is $s_{ours} = 78.6\,\%$, which is 54.1 % higher than the sensitivity score of the reference. This high sensitivity is probably due to a very pronounced shape-bias, which is, in turn, due to both the distinct shape of *cars* and the comparatively high number of *cars* in the training set [20]. The shape-bias for class *cars* is, hence, likely to be the highest. Our model performs with respect to other classes of "things" also better than the reference model. For example, the sensitivity score for classes *traffic light*, *traffic sign* and *pole* is higher by 25.6 %, 9.0 %, and 9.8 %, respectively. Both models perform poorly on "vehicles" that are, compared to *cars*, less frequent present in the training set (e.g., *truck, motorcycle, train*).

In the presence of severe Gaussian noise, the reference model is struggling to segment classes. The sensitivity is poor for every class, except for *traffic signs* and *building*, because, in the presence of image noise, the reference model tends to segment pixels often as these very classes as shown in Figures 4.7 and 4.12. Our model's sensitivity scores are often significantly higher with respect to these classes. Similar to the previously discussed results, the sensitivity with respect to "stuff" with less texture is often high (e.g., $s_{ours} = 95.9\,\%$ for *road*). The sensitivity scores are also high for "things" such as *persons* and *cars* ($s_{ours} = 52.7\,\%$, and $s_{ours} = 40.6\,\%$, respectively). Our models segment many classes corrupted by severe image noise, even though the model is not trained on image noise.

The reference model generally performs well when classes are low-pass filtered by severe Gaussian blur. This result is in accordance with the previous chapter's results, where we show that convolutional networks for semantic segmentation are quite robust to image blur, especially DeepLab-based networks. For class category "things", our model outperforms the reference model in most cases as well. The sensitivity score of our model for, *e.g.*, *person*, *rider*, and *car* is by approximately 20.0 % higher.

Silhouette. As discussed previously, the reference model is not able to segment any classes except *building* and *sky*, since it may rely mostly on the mean RGB of these classes when the class texture is removed (see Figure 4.11 a). When we additionally black the background (see Figure 4.11 d), our model, on the other hand, achieves considerably higher sensitivity scores for "things" such as *road*, *traffic light*, *traffic sign*, *pole*, *person*, and *car*.

Qualitative results. In Figure 4.12, we discuss the qualitative results of the experiments of this section. We illustrate, that our model segments the image decently when the classes *car* and *persons* are corrupted. The first row shows the original validation image and the corrupted variants for the classes *car* and respective ground truth in the second row. The third and fourth row shows the predicted segmentation maps of the reference model and our model. As discussed previously, the reference model segments classes, in the presence of severe image noise, mostly as *traffic sign*, whereas our model can segment the correct classes predominantly. Row $5-8$ show a similar result for class *persons*. Please see the caption for a more detailed discussion.

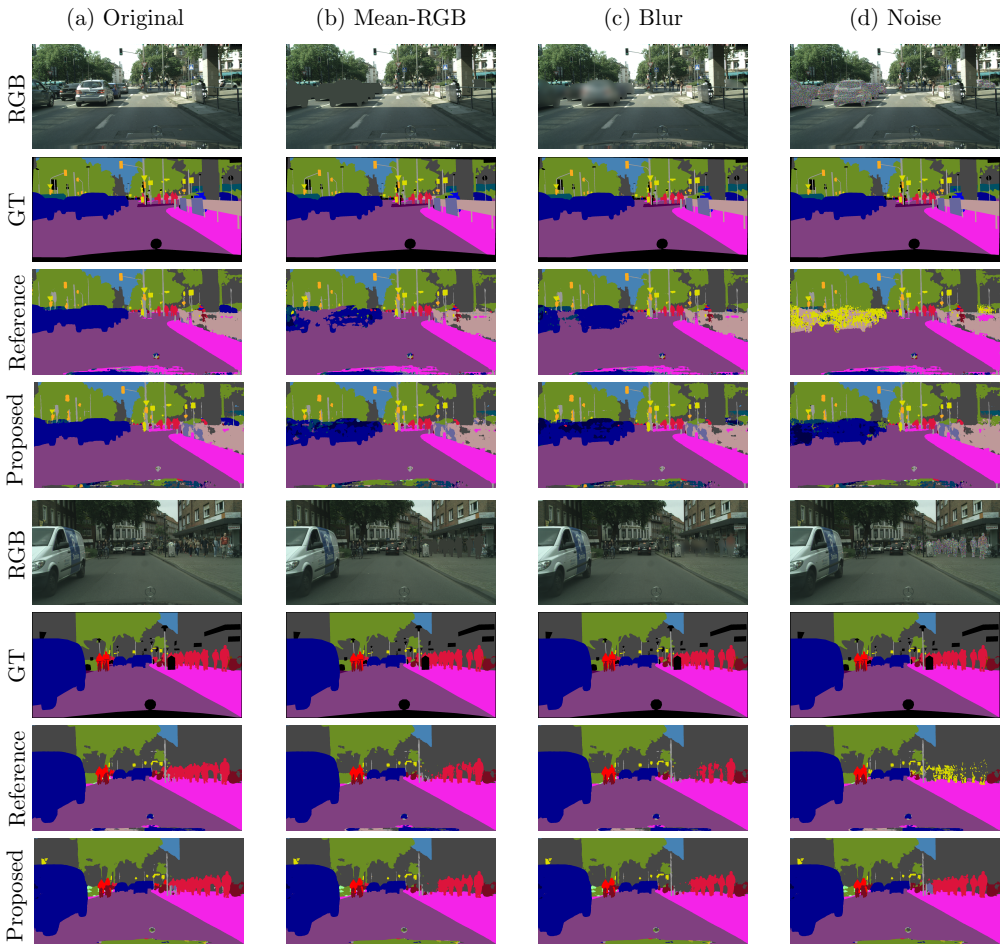


Figure 4.12: Qualitative results of our experiments to understand the effect of Painting-by-Numbers. We train the ResNet-50 network backbone on Cityscapes [20] with a standard training schema (i.e., with clean data only, reference model) and with Painting-by-Numbers (our model). **(top)** The first row shows the original validation image and the corrupted variants for class *car* and the respective ground truth in the second row. We replace either the class texture by the dataset-wide RGB-mean, strongly low-pass filtered the class, or added severe Gaussian image noise. The third row shows the predictions of the reference model. The fourth row shows the predictions of our model. The predictions in the fourth row (our model) are superior to the third row (reference model). Our model is able to withstand the image noise based corruption (last column) for which the reference model confuses *cars* with *traffic signs* mostly. **(bottom)** For *persons*, the reference model predicts well, when the RGB-mean replaces the texture of the class. Both models are relatively robust when the classes are low-pass filtered by severe Gaussian blur. Similar to the results with respect to class *car*, the reference model struggles to predict well for severe image noise and confuses *persons* also with *traffic signs* mostly.

4.5 Conclusions

This chapter proposed a simple yet effective data augmentation schema (Painting-by-Numbers) for semantic image segmentation. Painting-by-Numbers increases the robustness with respect to a wealth of common image corruptions in a generic way. Our training schema corrupts training data so that the textural information content of semantic classes becomes less reliable, forcing the network to develop and increase its shape-bias to segment the image correctly. Blending a texture-free image with the original training image is responsible for inducing this shape-bias.

But why does Painting-by-Numbers even work? Our experiments show that a convolutional network for semantic image segmentation is generally able to learn from entirely texture-free data, as shown in Figure 4.5. Whereas the segmentation performance with respect to classes without distinct shapes, such as buildings or sky, is low, classes like cars and persons are segmented well. The accurate predictions for the class road do further indicate that convolutional networks have a strong bias towards a class' location in the image: The road's shape is very diverse; however, in more or less every image of Cityscapes, the class road occupies a large area in the bottom half of the image. Whereas its texture can be very diverse, ranging from a homogeneous, consistent appearance to illuminated and shady with many road markings, it is almost entirely segmented correctly in a texture-free representation.

Our results for validating an increased shape-bias show a clear shape-oriented trend for the pixel segmentation. The experiments, where the texture of a class is entirely removed, and its extension, where the background is additionally removed (this avoids the network to use any foreground-background context information), are probably the most vital indicators for an increased bias object shape. Painting-by-Numbers is most effective for class category "things", such as cars and persons. As usual in machine learning-based optimization, the number of class instances in a training set greatly influences the individual shape bias towards a particular class: A shape-bias for class "train" is factually not present.

Painting-by-Numbers benefits' are that it does not require any additional data (as we directly access the anyway existing semantic segmentation ground truth), is easy to implement in any supervised segmentation model, and is computationally efficient. The CPU training time barely increases, and when Painting-by-Numbers is implemented on GPU, the training time increase is negligible.

Comparing Painting-by-Numbers with regular data augmentation, as presented in the previous chapter, shows that a model trained with Painting-by-Numbers has the best overall performance against the diverse set of image corruptions. This is an essential aspect since it may, in practice, be hard to actually identify all sources of image corruptions in the test scenario of a practical application. The performance of a model that is trained on image

noise with respect to image noise models is clearly out of range for a model trained with Painting-by-Numbers. Therefore, we demonstrated that the best of both worlds, meaning the generality of Painting-by-Numbers and robustness against image noise, can be achieved when Painting-by-Numbers is combined with a simple noise augmentation.

The research field of inducing human-vision based image processing strategy, or, in general, inducing-shape bias is growing. Painting-by-Numbers is undoubtedly a significant contribution to this field, as the trade-off between simplicity and computational efficiency is enormous. The principle of Painting-by-Numbers can be a basis to transfer to other research domains and computer vision tasks.

5

Conclusions and Outlook

5.1 Conclusions

Several years ago, when a convolutional network succeeded the ImageNet Large-Scale Visual Recognition Challenge for the first time, the deep learning research was kick-started [66]. Convolutional neural network-based vision algorithms set the de-facto state-of-the-art for many image understanding tasks. Throughout this thesis, it is shown that convolutional networks are able to segment images extraordinarily well; however, they struggle for test data having different distributions to the training data. With the emergence of autonomous driving, a prominent candidate for a safety-critical computer vision application, robust convolutional networks are essential. Especially in the field of environment perception, the variety of possible test scenarios is hardly feasible to test, and collecting sufficient and qualitatively appropriate data can be a major issue.

In chapter 3 we evaluated the behavior of convolutional networks for semantic image segmentation on corrupted image data extensively in terms of network architecture and the training data by utilizing an image database of almost 400,000 images generated from three datasets. Among the broad diversity of different convolutional network architectures benchmarked in this chapter, the performance for corrupted data differs vastly for particular image corruptions, even though the performance on clean data is often similar. Such results indicate that segmentation mechanics within networks may differ enormously, although the network's backbone is at the most ResNet-based.

Our experimental setup of carefully ablating architectural properties allowed us to rate the influence of the particular properties on network robustness, supporting the development of robust convolutional neural network architectures.

We demonstrated that semantic image segmentation models are able to generalize well to a broad range of image corruptions, even between two fundamentally different types of

corruptions (image smearing and image noise). However, applying the ideal training strategy ultimately is not straight-forward. It must be considered that the training duration has a regularization effect on the training algorithm [39], which affects, in turn, the generalization performance. The performance on clean data may decrease significantly when corrupted data is added to the training set, and that the number of training epochs ramps up quickly when a similar performance on the original data is also desired. Unfortunately, training for too many epochs may then hamper the performance again with respect to corrupted data.

In chapter 4 we proposed a simple yet effective training schema, Painting-by-Numbers, that increases the robustness of a diverse set of convolutional architectures for semantic segmentation in a generic way. The basis of the training schema is that the input data is altered so that the convolutional network is forced to trade-off its texture and shape bias to benefit higher network robustness against real-world image corruptions. The advantage of this strategy is, besides its computational efficiency, the fact that no additional data is needed as it requires the semantic segmentation ground truth, which is anyway given.

We showed that our training strategy is most effective against image corruptions of category noise. At a certain severity of image noise, the effectiveness of Painting-by-Numbers is limited, and it is unlikely that our strategy delivers satisfying results. However, we demonstrated that Painting-by-Numbers can easily be mixed with other augmentation schemes, such as a data augmentation with image noise, to combine the best of both worlds. We demonstrated the effectiveness of Painting-by-Numbers for many convolutional neural network architectures and image corruptions for the task of semantic image segmentation. We further presented a methodology for validating an increased shape-bias, which is based on the textural corruption on the class-level of the respective semantic segmentation dataset.

5.2 Outlook

In the following, we discuss several directions for future work with respect to benchmarking and increasing the robustness of convolutional neural networks with respect to common and real-world image corruptions. Possible directions of future work cover the topics of architectures found by neural architecture searches, the generalization behavior of segmentation networks, and the challenges of the proposed data augmentation method, increasing a model's shape-bias.

Neural-architecture-search on corrupted data. A specific convolutional neural network module that is developed by a neural architecture search (NAS) is automatically designed instead of hand-crafted. This thesis showed that such a module, referred to as the dense prediction cell, is especially vulnerable to corrupted image data. The research field of neural architecture searches has a significant research interest. It would be interesting to benchmark more of these kinds of architectures in the future to understand their robustness

properties. Further, instead of conducting a neural architecture search on clean data, with the goal to maximize performance on clean data, it would be interesting to conduct a neural architecture search on corrupted data, such as the image corruption categories as utilized in this thesis, which might also reveal novel insights for robust architectural network designs.

Influence of corruptions to generalization. Whereas we have seen that a decent level of network generalization can be reached when trained on noise, the generalization effects of using a specific noise model for the training differs enormously. Understanding this influence remains a challenge and offers a potential direction for future work. We mainly explored the generalization behavior of ResNet-based network backbones, and it is also important to explore the generalization behavior of other networks.

Architectural properties for other computer vision tasks. While this thesis can help increase the state-of-the-art of robust semantic segmentation models, it is interesting to develop similar benchmarks for the variety of other computer vision tasks such as object detection.

Challenges of Painting-by-Numbers w.r.t. ground-truth completeness. Our proposed training schema, Painting-by-Numbers, offers a cheap and effective way to improve the model robustness generically. Since the basis of Painting-by-Numbers is an increase in the network's bias towards shape, the ground truth image data underlies certain requirements. Firstly, the boundaries must be accurately labeled to allow for a meaningful texture-free representation of Painting-by-Numbers. Whereas this requirement is fulfilled for the Cityscapes dataset, it is not for a dataset such as PASCAL VOC 2012. Both the object borders and the semantic classes making up the background of this dataset's images are not labeled, which is essential for Painting-by-Numbers.

Challenges w.r.t. dataset properties. Increasing the network bias towards shape may also only be meaningful to a point where classes have a distinct shape, as we demonstrated in detail. The shape unambiguity is given for an application like urban driving due to the respective vulnerable key-classes of pedestrians or cars. However, for datasets consisting of many classes with highly similar shapes, it reduces the effectiveness of Painting-by-Numbers, and probably of other methods that also aim to increase the shape bias.

Challenges w.r.t. ground-truth quality. An interesting future work would be to transfer the concepts of that training schema to other computer vision tasks that rely on macroscopic features. This outlook is coupled with the challenge that semantic segmentation ground-truth data is not always given, and hence Painting-by-Numbers cannot be applied out of the box. One possible solution that avoids the expensive hand-label process is to utilize predicted segmentation maps for the respective dataset, which comes along with the challenge of estimating and rating the required segmentation quality for the task at hand.

Human-inspired classification strategies. Increasing the network bias towards object shape aims to incorporate the general strategy of how humans classify images. Building

on top of this, a fruitful way to go might be to incorporate more natural/human vision principles, which are probably biases, such as the context or scene layout.

Novel architectures and real-world image corruptions. The rapid progress in deep learning comes with a broad range of novel architectural design choices and architectural subtleties that should be explored in terms of robustness with respect to common and realistic image corruptions. Finally, benchmarks with respect to common image corruptions can become much more significant, which take into account a great selection of image corruptions, where our proposed, more realistic ones, might contribute.

A

Appendix

A.1 HDR Camera Model

The previous chapters show that convolutional networks are very prone to image noise. As described in section 2.1.5, real-world image noise is significantly more complex than the noise simulated by simple noise models since a variety of noise sources contribute to the total noise. In section 3.2, we propose a noise model that mimics commonly observable image noise caused by real-world cameras. The noise value added to a clean pixel in the linear color space is a sum of two features: Random *chrominance noise* and *luminance noise* values are drawn, which depends on the pixel's intensity of the respective color gamut. We model the noisy pixel intensity for a color channel c as a random variable $I_{noise,c}$:

$$\begin{aligned} I_{noise,c}(\Phi_c, N_{luminance}, N_{chrominance,c}; w_s) = \\ log_2(2^{\Phi_c} + w_s \cdot (N_{luminance} + N_{chrominance,c})) \end{aligned} \tag{A.1}$$

where Φ_c is the normalized pixel intensity of color channel c, $N_{luminance}$ and $N_{chrominance}$ are random variables following a Normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$, w_s is a weight factor, parameterized by one of the five chosen severity level s.

We extend this model to a bracketing HDR noise model with $n = 4$ brackets through selecting four normalized intensity values $I_{B,i}$, where $i \in \{1, 2, 3, 4\}$, where $I_{B,4} = 1$ and consider commonly observable features. Firstly, we incorporate an additional weight component for bracket w_e, where the smallest component is assigned to the bracket with the longest exposure and vice-versa. Secondly, another weight, w_e, is added within each exposure which is high at the transition point of subsequent exposures and decreases with increasing exposure time. In summary, we model the noisy pixel intensity for a color channel c as a random

variable $I_{HDR-noise,c}$:

$$I_{HDR-noise,c}(\Phi_c, N_{luminance}, N_{chrominance,c}; w_s; w_b; w_e) = \\ log_2(2^{\Phi_c} + w_e \cdot (w_s + w_b) \cdot (N_{luminance} + N_{chrominance,c})), \tag{A.2}$$

where w_e is a exposure-dependent weight of bracket i factor modeled by $w_e = 1 - \frac{\Phi_c}{\Phi_{B,i}}$, and w_b is the weight component assigned to each bracket.

A comparison of how this HDR noise model manifests in images is shown in Figure A.1. It illustrates noisy variants of our proposed noise models of an image-crop of the Cityscapes dataset. The HDR noise model causes smear artifacts, which are well visible at the bumper and road lane.

(a) Intensity noise (b) HDR noise

Figure A.1: Comparison of our proposed image noise models. (left) The intensity noise model, imitating single bracket noise. (right) The HDR noise model, imitating bracketing. The HDR noise model incorporates smear artifacts, mostly visible at darker, homogeneous areas near the road lane and the bumper, which originates from selecting and blending. For better visibility of the noise patterns, the contrast of the images is modified.

A.2 Detailed Quantitative Results

We list in the following the individual evaluation metrics score of the results reported in chapter 3.

We list the individual CD and rCD scores for non-DeepLabv3+ based models in Table A.1, evaluated on Cityscapes. Table A.2 contains the mIoU for clean and corrupted variants of the validation set of the Cityscapes dataset for several network backbones of the DeepLabv3+ architecture and each respective architectural modification. The individual CD and rCD scores, evaluated on Cityscapes, are in Table A.3 and Table A.4.

Table A.5 contains the mIoU for clean and corrupted variants of the validation set of PASCAL VOC 2012 for several network backbones of the DeepLabv3+ architecture. Figure A.3 illustrates the rCD for each ablated variant evaluated on PASCAL VOC 2012. The

individual CD and rCD scores, evaluated on PASCAL VOC 2012, are in Table A.6 and Table A.7.

Table A.8 contains the mIoU for clean and corrupted variants of the validation set of ADE20K for several network backbones of the DeepLabv3+ architecture. Figure A.4 illustrates the rCD for each ablated variant evaluated on ADE20K. The individual CD and rCD scores, evaluated on ADE20K, are in Table A.9 and Table A.10.

	Blur					**Noise**					**Digital**				**Weather**				
Backbone	Motion	Defocus	Glass	Gaussian	PSF	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
ICNet	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	**100.0**	100.0	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
VGG16	**105.6**	**124.3**	119.6	**119.1**	110.8	101.8	103.0	103.2	104.1	**115.6**	79.2	91.2	88.2	119.4	94.7	98.4	85.8	90.2	86.2
DilatedNet	102.6	115.1	**128.3**	111.4	**111.8**	92.2	93.9	91.3	93.3	91.9	80.2	**100.7**	85.4	107.3	93.4	97.3	89.8	90.7	84.4
ResNet-38	83.7	99.2	107.8	95.5	72.0	94.3	91.8	91.5	85.5	91.2	67.8	73.8	73.3	129.2	92.4	77.9	64.7	87.4	61.7
PSPNet	74.1	84.6	105.7	83.3	66.3	97.1	92.4	94.7	91.1	96.2	67.1	72.1	95.7	119.1	97.8	82.5	90.2	94.1	60.7
GSCNN	75.9	75.1	110.4	72.2	56.5	**103.2**	**106.4**	**104.3**	**104.4**	100.0	40.9	57.0	40.4	**133.2**	93.5	75.8	44.1	75.7	54.3
ICNet	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**
VGG16	119.1	**167.3**	160.1	**153.8**	779.8	104.3	106.1	106.6	109.9	132.5	54.0	84.6	79.9	142.7	93.1	99.1	75.3	85.5	44.6
DilatedNet	**120.5**	152.2	195.4	142.8	**1117.9**	92.3	95.0	90.8	94.4	91.8	64.0	**109.9**	79.5	123.8	94.3	102.5	87.8	89.9	56.6
ResNet-38	114.2	152.9	185.3	143.5	388.9	111.2	107.2	107.4	103.1	115.2	70.6	82.1	80.0	197.2	107.7	89.5	63.7	100.8	46.4
PSPNet	93.7	120.0	185.3	116.8	256.0	117.7	110.2	114.6	116.8	127.9	73.5	82.1	**125.2**	179.7	**117.9**	**101.7**	**114.7**	**113.7**	54.0
GSCNN	109.7	105.9	**211.0**	98.3	85.1	**131.2**	**136.4**	**134.2**	**147.9**	**141.6**	20.2	58.1	26.5	**216.0**	115.0	95.0	33.7	87.9	48.1

Table A.1: CD (top) and rCD (bottom) for corrupted variants of the validation set of the Cityscapes dataset for several non-Deeplabv3+ based architectures. ICNet is used as reference model. Highest CD and rCD per corruption is bold.

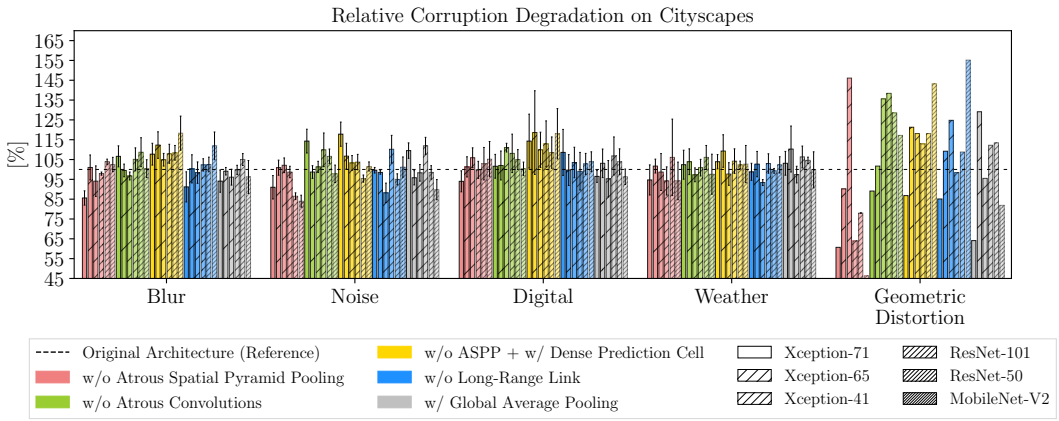


Figure A.2: Relative CD evaluated on Cityscapes for the proposed ablated variants of the DeepLabv3+ architecture with respect to image corruptions, employing six different network backbones. Each bar except for geometric distortion is averaged within a corruption category (error bars indicate the standard deviation). Bars above 100 % represent a relative decrease in performance compared to the respective reference architecture. Each ablated architecture is re-trained on the original training dataset. Removing ASPP may decrease performance significantly. The low rCD for geometric distortion indicates that the relative decrease of performance for this ablated variant is low. AC affect model performance, particularly against geometric distortion. The relative CD is often high against most image corruptions when DPC is used. The effect of GAP depends strongly on the network backbone. Best viewed in color.

		Blur					Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	PSF	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	**72.0**	**53.5**	49.0	**45.3**	49.1	**70.5**	6.4	7.0	6.6	16.6	26.9	51.7	46.7	32.4	27.2	**13.7**	38.9	**47.4**	**17.3**	65.5
w/o ASPP	64.9	45.5	40.4	39.0	41.5	63.3	7.7	8.7	8.9	19.9	28.4	41.7	36.1	27.6	20.7	13.0	36.8	37.8	14.0	61.9
w/o AC	71.2	52.1	49.1	42.8	49.3	69.8	3.6	7.2	4.5	19.6	29.2	49.8	46.2	31.4	**28.1**	10.0	**44.6**	45.2	16.7	63.5
w/ DPC	71.6	49.4	42.2	43.7	43.8	69.2	3.5	4.9	3.9	16.1	27.4	45.0	38.6	30.1	24.1	9.8	42.8	43.9	14.0	62.2
w/o LRL	71.1	49.7	43.9	44.4	45.1	68.9	1.9	2.6	2.5	19.6	26.6	49.1	43.5	32.2	26.3	10.4	39.5	44.9	14.7	60.9
w/ GAP	71.4	52.2	**50.6**	43.3	**51.8**	69.8	**8.5**	**10.9**	**10.8**	**26.0**	**32.5**	**51.8**	**47.3**	**35.3**	25.7	12.7	43.4	45.1	12.6	**66.0**
RN-50	76.6	**58.5**	**56.6**	47.2	**57.7**	74.8	6.5	7.2	10.0	31.1	30.9	**58.2**	**54.7**	41.3	27.4	**12.0**	**42.0**	55.9	**22.8**	**69.5**
w/o ASPP	71.4	52.3	50.7	41.2	52.0	69.7	10.1	11.3	**13.8**	31.2	33.3	50.2	48.4	37.0	25.3	12.0	38.6	42.7	18.7	65.9
w/o AC	76.0	56.7	53.1	**47.3**	54.1	73.8	2.4	6.1	5.1	25.5	25.7	56.8	51.4	38.9	27.6	9.7	40.8	52.0	20.1	66.9
w/ DPC	**76.9**	57.0	54.7	46.9	56.2	74.2	**10.7**	**12.6**	13.6	**33.1**	32.0	54.5	53.6	**41.5**	25.1	11.4	41.3	**56.3**	20.4	68.6
w/o LRL	75.6	57.9	54.4	46.4	55.6	73.8	7.9	9.3	11.2	31.8	**34.7**	56.2	51.6	40.2	**28.5**	11.9	41.4	55.4	21.1	67.9
w/ GAP	76.5	56.7	55.7	45.8	57.4	**75.2**	5.5	7.8	9.5	31.3	34.5	57.7	51.4	41.1	28.3	10.5	40.4	54.5	20.1	68.5
RN-101	77.1	**59.1**	56.3	47.7	57.3	75.2	**13.2**	**13.9**	**16.3**	**36.9**	**39.9**	**59.2**	54.5	**41.5**	**37.4**	**11.9**	**47.8**	55.1	**22.7**	**69.7**
w/o ASPP	71.1	53.8	50.6	42.2	51.7	68.8	9.5	9.8	12.7	30.7	32.5	52.1	48.3	36.7	33.2	13.3	43.5	47.8	23.2	66.4
w/o AC	75.7	57.9	52.5	46.6	53.9	73.3	8.4	11.0	11.6	31.5	28.8	53.5	53.1	39.1	34.2	9.9	44.7	55.0	20.0	65.5
w/ DPC	77.0	58.5	53.5	46.7	54.8	75.3	11.7	12.1	15.6	36.4	35.5	53.7	54.3	39.8	30.9	10.1	44.0	**56.0**	19.3	68.6
w/o LRL	76.5	58.7	54.6	47.5	55.7	74.3	9.1	8.3	12.1	33.5	30.3	57.0	**57.6**	40.9	35.7	9.3	44.3	55.4	20.8	69.2
w/ GAP	**77.3**	58.7	**56.9**	**48.4**	**57.8**	**75.9**	8.2	7.4	11.6	32.0	32.8	55.6	55.8	39.3	36.4	11.5	44.8	52.5	22.6	69.0
XC-41	**77.8**	61.6	54.9	51.0	**54.7**	**76.1**	17.0	17.3	21.6	43.7	**48.6**	63.6	56.9	**51.7**	**38.5**	18.2	46.6	57.6	20.6	**73.0**
w/o ASPP	75.4	59.7	55.5	47.4	55.4	73.1	15.1	14.4	19.7	40.7	43.6	60.4	52.5	46.8	37.0	18.0	47.2	52.4	22.1	68.4
w/o AC	77.4	62.2	**55.6**	51.3	54.5	75.4	17.7	15.7	22.1	42.8	46.5	61.6	54.9	47.8	34.3	17.8	46.6	**59.1**	20.9	70.9
w/ DPC	77.5	60.6	53.0	50.8	52.5	75.8	15.1	10.7	20.3	42.7	48.4	63.6	53.4	46.0	36.0	17.6	**50.0**	56.7	20.6	71.8
w/o LRL	76.8	**62.3**	53.2	50.6	53.0	75.1	**21.3**	**19.2**	**27.6**	**49.3**	51.7	63.9	55.2	48.0	33.8	20.5	48.3	57.6	23.9	70.8
w/ GAP	77.1	61.5	54.8	**53.1**	53.9	75.6	20.0	16.4	24.8	43.4	46.6	**65.7**	**57.6**	50.4	36.2	16.5	48.6	56.8	22.6	72.5
XC-65	**78.4**	**63.9**	59.1	52.8	**59.2**	**76.8**	15.0	10.6	19.8	42.4	46.5	**65.9**	**59.1**	46.1	31.4	**19.3**	50.7	**63.6**	**23.8**	**72.7**
w/o ASPP	75.8	61.6	56.1	51.8	54.6	74.1	14.3	7.7	18.8	39.0	41.6	62.0	57.2	43.1	29.7	15.6	46.9	60.3	23.4	70.6
w/o AC	77.7	63.9	58.7	51.5	57.8	75.7	14.1	14.8	19.5	41.9	45.1	63.9	58.3	42.9	35.0	15.7	**51.4**	60.9	21.4	71.8
w/ DPC	77.7	62.4	55.0	50.4	54.5	74.7	8.9	4.8	13.2	37.1	**47.7**	62.5	48.4	45.4	30.3	17.3	47.1	59.6	21.9	70.7
w/o LRL	77.7	64.5	58.6	49.5	57.9	75.9	15.1	12.0	19.9	42.1	45.9	63.8	57.9	46.1	**35.9**	18.4	46.3	63.5	22.0	71.4
w/ GAP	78.4	63.9	**59.4**	**53.5**	58.8	76.2	**18.8**	**15.4**	**23.7**	**43.7**	45.7	65.2	56.5	**48.0**	31.5	18.8	49.4	59.1	20.7	71.0

Table A.2: Mean IoU for clean and corrupted variants of the validation set of the Cityscapes dataset for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Every mIoU is averaged over all available severity levels, except for corruptions of category noise where only the first three severity levels are considered. The standard deviation for image corruptions of category noise is 0.2 or less. Highest mIoU per corruption is bold.

	Blur					Noise					Digital				Weather				
Backbone	Motion	Defocus	Glass	Gaussian	PSF	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**117.2**	**116.8**	**111.5**	**115.0**	**124.2**	98.6	98.1	97.6	96.1	97.9	**120.6**	**120.0**	**107.1**	**109.0**	100.7	**103.5**	**118.2**	104.0	**110.4**
w/o AC	103.1	99.9	104.6	99.8	102.2	103.0	99.7	102.2	96.4	96.9	104.0	101.1	101.5	98.8	104.2	90.8	104.1	100.7	105.7
w/ DPC	108.9	113.4	102.9	110.4	104.2	103.1	102.3	102.9	**100.6**	99.3	113.9	115.3	103.4	104.3	**104.5**	93.7	106.7	103.9	109.4
w/o LRL	108.2	110.0	101.7	108.0	105.3	**104.8**	**104.8**	**104.4**	96.3	**100.0**	105.5	106.1	100.3	101.2	103.8	99.1	104.8	103.1	113.2
w/ GAP	102.8	96.9	103.6	94.8	102.2	97.7	95.8	95.5	88.7	92.3	99.9	98.9	95.7	102.1	101.1	92.8	104.3	**105.6**	98.4
RN-50	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**115.1**	**113.6**	**111.3**	**113.5**	**120.3**	96.2	95.5	95.8	99.9	96.4	**119.0**	**113.9**	**107.3**	102.9	100.1	**105.8**	**130.1**	**105.3**	**111.8**
w/o AC	104.4	108.1	99.7	108.5	104.2	**104.3**	**101.2**	**105.4**	**108.1**	**107.5**	103.2	107.4	104.1	99.8	**102.6**	102.0	109.0	103.5	108.6
w/ DPC	103.6	104.3	100.5	103.7	102.3	95.5	94.2	96.0	97.1	98.3	108.6	102.4	99.7	**103.3**	100.7	101.0	99.1	103.1	103.0
w/o LRL	101.5	104.9	101.5	105.1	104.2	98.5	97.7	98.6	99.1	94.4	104.6	106.8	101.8	98.5	100.1	101.0	101.3	102.2	105.1
w/ GAP	104.3	102.1	102.7	100.7	98.3	101.1	99.3	100.6	99.7	94.7	101.2	107.3	100.3	98.8	101.8	102.8	103.3	103.4	103.2
RN-101	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**113.2**	**113.1**	**110.5**	**113.3**	**125.6**	104.3	104.8	104.4	**109.9**	112.2	**117.6**	**113.4**	**108.3**	106.7	98.4	**108.1**	**116.2**	99.4	111.0
w/o AC	103.1	108.6	102.2	108.0	107.5	105.6	103.3	105.7	108.6	118.5	114.2	103.1	104.2	105.2	102.4	105.9	100.3	103.5	**113.9**
w/ DPC	101.5	106.5	101.9	105.8	99.8	101.8	102.1	100.8	100.8	107.2	113.5	100.4	103.0	**110.5**	102.1	107.2	98.0	**104.4**	103.5
w/o LRL	101.2	103.9	100.5	103.8	103.8	104.8	106.4	105.1	105.4	**115.9**	105.6	93.1	101.1	102.8	**103.1**	106.7	99.5	102.5	101.5
w/ GAP	101.1	98.6	98.7	98.9	97.1	**105.8**	**107.5**	**105.7**	107.8	111.8	108.9	97.1	103.7	101.7	100.5	105.6	105.8	100.2	102.2
XC-41	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0
w/o ASPP	**105.1**	98.5	**107.5**	98.5	**112.2**	**102.4**	103.5	**102.4**	**105.3**	**109.8**	**108.9**	**110.2**	110.2	102.3	100.2	98.8	**112.4**	98.2	**116.9**
w/o AC	98.5	98.3	99.5	100.4	102.6	99.1	102.0	99.3	101.6	104.0	105.6	104.7	108.2	106.8	100.5	**100.0**	96.6	99.6	107.5
w/ DPC	102.8	**104.2**	100.5	**104.9**	101.0	102.4	**108.0**	101.6	101.9	100.4	100.0	108.1	**111.9**	104.1	100.8	93.6	102.1	100.0	104.4
w/o LRL	98.2	103.6	100.9	103.7	103.9	94.9	97.7	92.3	90.1	94.0	99.2	103.9	107.7	**107.6**	97.2	96.9	100.0	95.9	108.0
w/ GAP	100.3	100.2	95.7	101.7	101.8	96.5	101.1	95.9	100.5	103.9	94.3	98.5	102.6	103.8	**102.1**	96.2	101.9	97.5	101.6
XC-65	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**106.4**	107.2	102.0	111.4	**111.7**	100.8	103.3	101.4	105.9	**109.3**	**111.2**	104.7	105.5	**102.5**	**104.5**	107.7	109.1	100.6	**107.5**
w/o AC	100.1	101.0	102.6	103.3	104.8	101.1	95.3	100.5	100.9	102.7	105.6	102.1	**106.0**	94.8	104.4	98.7	107.4	103.1	103.1
w/ DPC	104.0	**109.8**	105.1	**111.6**	108.9	**107.2**	**106.5**	**108.3**	**109.3**	97.9	109.9	**126.3**	101.4	101.7	102.4	107.4	111.0	102.5	107.1
w/o LRL	98.3	101.1	**107.0**	103.2	104.0	99.8	98.4	99.9	100.6	101.3	106.1	103.1	99.9	93.4	101.1	**108.9**	100.3	102.4	104.8
w/ GAP	100.0	99.1	98.5	101.0	102.5	95.5	94.6	95.2	97.8	101.5	101.9	106.6	96.5	99.9	100.6	102.6	**112.4**	**104.2**	106.3
XC-71	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0
w/o ASPP	**109.6**	103.6	101.1	105.2	**115.1**	95.7	95.6	96.5	102.4	110.9	114.6	106.0	**108.3**	100.5	101.6	102.9	**116.1**	97.9	**105.9**
w/o AC	105.3	**107.6**	100.4	**105.7**	100.9	**108.4**	105.8	110.2	**114.3**	114.0	101.4	**107.4**	102.0	99.1	100.9	100.8	108.2	98.2	99.7
w/ DPC	103.7	103.7	98.8	105.6	97.9	108.9	**109.0**	**110.8**	113.9	**115.5**	110.2	98.2	103.4	**113.7**	**105.5**	**103.7**	100.2	99.2	96.0
w/o LRL	99.8	94.0	**102.8**	95.4	98.7	101.1	101.7	101.5	99.9	100.5	**111.0**	98.7	105.4	106.9	102.4	103.6	99.3	98.5	98.7
w/ GAP	99.8	97.8	91.8	99.3	94.0	106.1	102.8	106.9	107.3	109.1	100.1	96.2	97.7	99.3	102.6	98.9	105.4	99.0	90.8

Table A.3: CD for corrupted variants of the validation set of the Cityscapes dataset for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Highest CD per corruption is bold.

	Blur					Noise					Digital				Weather				
Backbone	Motion	Defocus	Glass	Gaussian	PSF	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0
w/o ASPP	104.8	106.3	96.8	102.1	101.3	87.2	86.3	85.7	81.2	80.8	114.0	114.0	94.1	98.7	88.9	84.9	109.9	93.0	46.3
w/o AC	103.2	96.1	**106.4**	95.8	89.0	103.0	98.3	101.9	93.1	93.1	105.5	99.0	100.5	96.3	104.9	80.5	105.4	99.6	117.2
w/ DPC	**119.9**	**127.8**	104.3	**121.3**	**152.5**	103.8	102.6	103.4	**100.2**	97.9	**130.9**	**130.5**	**104.7**	**106.0**	**105.9**	87.1	**112.5**	105.1	143.2
w/o LRL	115.5	118.1	100.0	113.8	141.6	**105.4**	**105.4**	**104.8**	92.8	97.9	108.5	109.2	98.1	100.0	104.1	95.5	106.5	103.0	**155.2**
w/ GAP	103.5	90.4	105.0	85.8	100.6	95.8	93.0	92.6	81.9	86.1	96.7	95.2	91.1	102.0	100.5	84.7	106.6	**107.3**	81.9
RN-50	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	106.1	103.7	102.8	102.9	98.2	87.5	86.6	86.5	88.4	83.3	**115.1**	105.2	97.5	93.8	92.1	94.9	**139.3**	98.0	78.0
w/o AC	106.7	**114.5**	97.5	**115.9**	126.3	**104.9**	**100.7**	**106.5**	**111.0**	**110.0**	103.9	112.5	**105.0**	98.4	102.6	101.7	116.3	103.9	**128.5**
w/ DPC	**110.3**	111.1	102.1	110.2	**154.5**	94.5	92.8	95.2	96.3	98.3	121.6	106.6	100.5	**105.5**	101.5	102.8	99.8	**105.1**	118.1
w/o LRL	98.2	106.0	99.5	106.5	106.5	96.7	95.5	96.8	96.5	89.5	105.4	109.8	100.3	95.8	98.7	99.0	98.3	101.4	108.8
w/ GAP	109.7	104.3	**104.7**	101.3	73.1	101.4	99.0	100.7	99.4	91.9	102.5	**114.9**	100.3	98.1	**102.3**	**104.5**	106.9	104.9	113.4
RN-101	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	96.5	98.6	98.3	98.3	119.0	96.4	97.0	96.1	100.6	103.6	106.5	**100.5**	96.7	95.5	88.7	94.0	105.9	88.1	63.9
w/o AC	99.3	111.4	99.2	110.4	**125.1**	105.5	102.3	105.5	110.1	**126.1**	124.6	100.1	103.0	104.7	101.1	105.8	94.3	102.4	**138.4**
w/ DPC	103.0	**113.2**	**103.1**	**112.2**	92.1	102.3	102.7	101.0	101.0	111.4	**130.4**	100.4	104.7	**116.3**	102.7	**112.6**	95.5	**106.1**	112.9
w/o LRL	99.5	105.5	98.9	105.4	120.0	105.7	107.9	106.1	107.1	124.1	109.5	83.7	100.3	102.9	**103.3**	110.0	96.3	102.4	98.4
w/ GAP	**103.8**	98.2	98.5	98.9	74.0	**108.3**	**110.6**	**108.3**	**112.8**	119.8	121.7	95.1	**106.8**	103.4	101.0	110.8	**112.9**	100.7	112.2
XC-41	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	**100.0**	100.0
w/o ASPP	97.7	86.9	**104.9**	86.8	**133.6**	99.4	100.9	99.1	101.9	**109.2**	106.3	109.9	109.9	97.7	96.4	90.5	**114.4**	93.4	**146.1**
w/o AC	94.3	95.3	97.8	99.4	116.9	98.3	102.2	98.5	101.6	106.0	**112.1**	108.2	114.0	**109.8**	100.1	98.9	91.2	98.9	135.6
w/ DPC	**104.7**	**106.9**	99.8	**108.2**	96.2	**102.7**	**110.4**	**101.7**	**102.2**	99.7	97.8	**115.3**	**120.9**	105.6	100.5	88.1	102.8	99.5	118.1
w/o LRL	89.6	102.9	98.0	103.0	97.2	91.4	95.3	87.5	80.8	86.0	91.0	103.3	110.5	109.4	94.5	91.5	95.2	92.6	124.8
w/ GAP	96.7	97.6	89.7	100.5	88.3	94.1	100.4	93.1	99.0	104.6	80.9	93.8	102.4	104.3	**101.8**	91.4	100.9	95.4	95.5
XC-65	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	98.0	101.7	93.6	**110.6**	106.5	97.0	100.4	97.4	102.1	**107.4**	109.7	96.4	101.1	98.1	101.7	104.3	104.8	96.1	90.2
w/o AC	95.0	98.1	101.8	103.1	121.2	100.2	92.7	99.3	99.3	102.2	109.3	100.5	**107.7**	90.7	**104.7**	94.9	113.0	103.0	101.7
w/ DPC	**105.0**	**117.0**	106.5	120.8	**181.5**	**108.5**	**107.4**	**110.2**	**112.8**	94.2	**121.2**	**152.0**	100.0	**100.9**	102.1	110.6	122.2	102.2	121.3
w/o LRL	90.4	98.3	**109.8**	102.7	108.8	98.6	96.8	98.5	98.8	99.7	110.2	102.5	97.5	88.8	100.1	**113.1**	95.5	101.9	109.2
w/ GAP	99.7	97.8	97.0	102.0	132.6	93.9	92.8	93.4	96.4	102.4	104.8	113.8	94.1	99.7	100.7	104.4	**130.2**	**105.7**	**129.1**
XC-71	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**
w/o ASPP	91.2	81.3	84.3	85.5	49.1	86.9	87.3	87.3	91.1	102.4	99.6	88.8	99.0	88.4	94.2	88.3	107.3	89.1	60.7
w/o AC	107.9	**112.4**	97.9	108.1	76.3	110.0	106.5	112.6	120.5	121.8	97.1	**111.5**	101.0	96.7	99.9	98.7	**114.9**	96.3	89.1
w/ DPC	**110.3**	109.0	98.5	**113.1**	**84.6**	**112.2**	**112.1**	**115.0**	**122.3**	**127.6**	**132.1**	97.1	106.2	**121.8**	**107.7**	**107.1**	101.6	**99.2**	86.8
w/o LRL	94.2	82.5	**102.1**	85.8	53.0	100.3	101.1	100.7	97.7	98.1	125.9	93.4	**106.6**	108.7	102.0	103.6	92.9	96.7	85.1
w/ GAP	98.9	94.6	85.0	98.1	33.6	108.0	103.5	109.2	111.2	115.6	99.5	91.8	95.8	98.8	103.3	97.8	112.7	98.5	64.1

Table A.4: Relative CD for corrupted variants of the validation set of the Cityscapes dataset for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Highest rCD per corruption is bold.

		Blur				Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
RN-50	69.6	38.7	**43.5**	31.1	45.5	43.2	40.7	44.2	50.9	59.8	63.5	50.3	63.8	58.2	31.3	47.0	56.9	39.8	67.2
w/o ASPP	57.6	28.8	28.8	21.9	31.5	31.7	29.2	32.3	38.4	45.9	49.8	36.0	51.1	45.0	23.1	37.6	42.0	26.7	56.3
w/o AC	68.9	39.3	41.6	29.0	43.6	43.6	42.0	44.1	50.8	59.3	62.7	48.9	62.9	56.9	31.2	46.4	55.9	38.3	65.9
w/ DPC	68.0	38.6	40.6	29.7	**42.5**	44.0	42.2	45.2	51.6	59.8	61.5	48.9	62.6	56.5	30.4	46.8	56.0	37.9	64.9
w/o LRL	69.0	40.0	41.2	30.1	43.0	43.5	41.9	44.3	50.8	59.7	62.4	48.5	62.5	57.2	30.4	46.6	55.8	39.1	65.5
w/ GAP	**71.5**	**41.6**	42.9	**33.0**	**45.9**	**45.6**	**45.3**	**46.4**	**53.7**	**63.4**	**65.7**	**52.1**	**66.1**	**59.5**	**33.6**	**50.0**	**60.3**	**43.8**	**67.6**
RN-101	70.3	45.8	45.6	33.2	46.6	49.4	48.3	50.1	55.4	61.3	64.5	50.6	65.3	59.7	31.4	50.4	57.6	41.2	67.6
w/o ASPP	60.5	36.4	34.2	25.1	36.3	36.4	34.1	37.1	43.0	50.5	53.6	39.2	54.1	49.8	24.5	41.9	45.5	29.6	59.8
w/o AC	70.2	**46.8**	45.8	33.5	46.3	46.0	45.2	46.6	52.9	60.5	64.4	50.5	64.5	59.6	32.3	51.0	57.9	40.4	66.8
w/ DPC	69.5	44.5	44.8	32.3	46.2	48.4	45.0	49.4	54.8	61.5	63.5	51.3	64.0	59.4	32.3	49.9	58.3	40.5	65.7
w/o LRL	69.6	44.2	44.8	33.5	45.8	47.4	45.2	48.5	53.8	61.3	64.1	50.7	64.6	58.4	32.2	50.6	57.5	40.4	65.9
w/ GAP	**72.5**	46.7	**46.3**	**36.5**	**47.6**	**50.5**	**48.5**	**51.3**	**56.6**	**64.3**	**66.7**	**53.6**	**66.0**	**61.1**	**36.4**	**52.6**	**61.7**	**44.7**	**68.4**
XC-41	75.5	52.9	54.7	35.5	53.9	55.8	53.3	56.7	62.8	67.6	70.8	51.9	70.9	64.6	42.5	59.0	63.1	48.4	**73.0**
w/o ASPP	66.9	45.9	45.3	30.4	45.6	47.2	45.5	48.0	52.9	58.5	61.0	43.1	61.5	56.0	34.6	50.6	53.1	39.3	65.4
w/o AC	75.0	53.2	54.9	36.5	54.9	54.1	52.6	55.5	61.4	67.1	69.7	50.5	70.5	64.5	40.9	60.1	62.3	47.0	71.8
w/ DPC	75.3	51.6	54.8	**37.5**	54.3	**56.8**	55.1	58.1	63.0	67.8	70.0	50.8	70.7	65.5	40.6	58.1	61.9	47.7	72.0
w/o LRL	76.1	52.9	**56.7**	36.7	**55.8**	56.7	**56.6**	**58.3**	**63.9**	68.8	70.9	**53.8**	**71.9**	65.1	41.4	59.1	63.3	48.4	72.9
w/ GAP	**76.5**	**55.0**	55.2	36.3	55.0	55.3	55.7	56.6	62.7	**68.8**	**71.1**	52.8	71.5	**66.1**	**43.3**	**61.4**	**63.7**	**48.9**	72.3
XC-65	76.5	53.5	58.3	37.7	57.2	56.6	54.7	57.4	62.5	69.3	71.8	55.9	72.1	66.7	40.2	58.5	64.0	47.5	**73.6**
w/o ASPP	70.6	47.5	47.8	29.1	48.6	45.4	44.2	45.6	51.8	62.4	64.7	48.1	64.6	58.3	35.7	52.6	56.4	39.4	68.7
w/o AC	76.4	**57.6**	57.3	38.4	56.9	56.5	54.5	57.0	62.2	69.6	71.4	55.0	72.3	66.3	42.5	60.4	63.6	46.4	73.6
w/ DPC	76.1	53.7	55.0	34.6	55.0	54.8	54.0	56.0	61.6	68.9	70.9	54.0	71.1	66.0	40.9	58.3	61.9	46.5	73.4
w/o LRL	76.2	55.2	55.1	36.6	55.6	56.5	55.4	56.8	61.8	68.9	71.1	56.1	71.1	64.3	40.3	58.4	64.2	46.1	73.0
w/ GAP	**77.5**	56.8	**59.8**	**41.8**	**59.1**	**57.9**	**57.6**	**57.6**	**62.6**	**71.0**	**73.1**	**57.4**	**73.0**	**67.3**	**42.8**	**61.1**	**65.3**	**49.7**	73.2
XC-71	76.7	56.5	59.1	40.2	59.5	56.6	57.8	57.6	63.2	69.9	72.1	57.1	72.6	68.1	43.9	60.9	66.1	50.9	**73.6**
w/o ASPP	70.5	48.3	49.2	33.3	50.1	47.5	47.1	48.2	54.6	62.7	65.1	48.8	65.6	60.2	37.0	53.4	57.3	44.1	69.3
w/o AC	75.7	55.9	58.8	**41.8**	59.0	57.1	58.2	57.3	62.6	69.5	71.0	56.9	71.4	67.6	41.9	60.9	64.1	48.2	73.0
w/ DPC	76.8	53.5	54.6	35.8	55.4	55.5	55.6	54.7	60.5	69.0	71.4	54.0	71.0	66.3	42.5	58.3	63.3	49.7	73.3
w/o LRL	76.3	56.4	56.4	40.5	55.9	**59.9**	**59.3**	**60.2**	**64.6**	71.1	72.1	53.0	72.3	67.7	43.8	59.3	64.1	50.8	73.2
w/ GAP	**77.7**	**57.8**	**58.7**	38.3	**59.1**	58.8	55.2	58.4	63.7	**71.9**	**73.8**	**60.4**	**73.9**	**69.2**	**46.9**	**61.6**	**67.9**	**53.5**	73.5

Table A.5: Mean IoU for clean and corrupted variants of the validation set of PASCAL VOC 2012 for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Every mIoU is averaged over all available severity levels, except for corruptions of category noise where only the first three severity levels are considered. The standard deviation for image corruptions of category noise is 0.3 or less. Highest mIoU per corruption is bold.

	Blur				Noise					Digital				Weather				
Backbone	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
RN-50	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**116.1**	**126.1**	**113.3**	**125.7**	**120.3**	**119.4**	**121.4**	**125.6**	**134.7**	**137.7**	**128.7**	**135.0**	**131.4**	**111.9**	**117.8**	**134.5**	**121.7**	**133.3**
w/o AC	99.1	103.4	103.0	103.5	99.3	97.9	100.2	100.4	101.3	102.3	102.6	102.4	103.1	100.0	101.1	102.2	102.6	104.1
w/ DPC	100.2	105.2	101.9	105.5	98.6	97.6	98.2	98.6	100.1	105.5	102.6	103.2	104.0	101.3	100.5	101.9	103.3	107.0
w/o LRL	98.0	104.1	101.5	104.6	99.4	98.0	99.9	100.3	100.3	103.0	103.6	103.4	102.4	101.2	100.9	102.4	101.2	105.2
w/ GAP	95.4	101.1	97.2	99.4	95.9	92.3	96.2	94.3	91.2	94.0	96.3	93.7	96.7	96.6	94.4	92.1	93.5	98.9
RN-101	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**117.2**	**120.9**	**112.2**	**119.4**	**125.7**	**127.4**	**126.1**	**127.8**	**127.9**	**130.9**	**123.0**	**132.3**	**124.5**	**110.1**	**117.1**	**128.5**	**119.8**	**124.0**
w/o AC	98.1	99.6	99.5	100.7	106.7	106.0	107.0	105.7	102.1	100.5	100.1	102.3	100.4	98.7	98.8	99.3	101.4	102.3
w/ DPC	102.3	101.4	101.4	100.8	102.0	106.2	101.5	101.4	99.5	103.0	98.6	103.6	100.9	98.7	101.2	98.3	101.3	105.9
w/o LRL	102.9	101.4	99.6	101.5	103.9	105.9	103.3	103.5	100.0	101.2	99.7	102.0	103.3	99.0	99.6	100.2	101.5	105.2
w/ GAP	98.3	98.6	95.0	98.2	97.8	99.5	97.7	97.2	92.2	93.8	93.9	97.8	96.6	92.8	95.6	90.4	94.0	97.6
XC-41	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**114.8**	**120.8**	**107.9**	**117.9**	**119.3**	**116.6**	**120.1**	**126.5**	**127.9**	**133.7**	**118.3**	**132.3**	**124.4**	**113.6**	**120.6**	**127.2**	**117.6**	**128.0**
w/o AC	99.4	99.7	98.5	97.8	103.9	101.5	103.0	103.6	101.6	103.7	103.0	101.4	100.2	102.7	97.3	102.2	102.7	104.3
w/ DPC	102.9	99.8	97.0	99.1	97.8	96.1	96.8	99.4	99.4	102.6	102.3	100.5	97.4	103.2	102.3	103.2	101.5	103.7
w/o LRL	100.0	95.6	98.2	95.8	97.9	92.9	96.5	97.0	96.1	99.6	96.0	96.6	98.6	101.8	99.8	99.5	100.0	100.1
w/ GAP	95.5	99.0	98.8	97.5	101.2	94.9	100.3	100.3	96.4	99.0	98.1	97.8	95.7	98.6	94.3	98.4	99.0	102.4
XC-65	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**112.8**	**125.2**	**113.9**	**120.2**	**125.9**	**123.1**	**127.5**	**128.7**	**122.3**	**125.3**	**117.7**	**126.7**	**125.3**	**107.4**	**114.0**	**121.1**	**115.4**	**118.3**
w/o AC	91.1	102.5	98.9	100.7	100.3	100.5	100.9	101.0	98.8	101.3	102.2	99.1	101.2	96.2	95.4	101.2	102.1	99.8
w/ DPC	99.5	108.0	104.9	105.3	104.1	101.5	103.2	102.6	101.2	103.4	104.4	103.5	102.3	98.8	100.3	105.8	101.9	100.5
w/o LRL	96.3	107.7	101.8	103.8	100.3	98.4	101.2	101.9	101.2	102.6	99.7	103.4	107.1	99.7	100.3	99.4	102.7	102.2
w/ GAP	92.8	96.4	93.4	95.6	97.0	93.6	99.4	99.8	94.3	95.3	96.6	96.7	98.2	95.6	93.6	96.5	95.7	101.4
XC-71	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**118.9**	**124.3**	**111.5**	**123.0**	**121.0**	**125.2**	**122.0**	**123.4**	**123.8**	**125.0**	**119.2**	**125.6**	**124.9**	**112.3**	**119.1**	**126.0**	**113.9**	**116.6**
w/o AC	101.3	100.9	97.3	101.2	98.8	99.0	100.7	101.8	101.4	103.9	100.4	104.4	101.5	103.5	100.1	105.9	105.4	102.3
w/ DPC	107.0	111.1	107.4	110.1	102.5	105.1	106.7	107.5	102.9	102.4	107.1	105.8	105.5	102.5	106.7	108.1	102.5	101.1
w/o LRL	100.2	106.7	99.5	108.7	92.4	96.5	93.9	96.3	95.9	99.9	109.4	101.2	101.2	100.2	104.0	105.8	100.2	101.5
w/ GAP	97.1	101.1	103.2	100.8	94.8	106.1	98.2	98.7	93.2	93.9	92.3	95.2	96.5	94.7	98.3	94.7	94.7	100.6

Table A.6: Mean CD for corrupted variants of the validation set of PASCAL VOC 2012 for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Highest CD per corruption is bold.

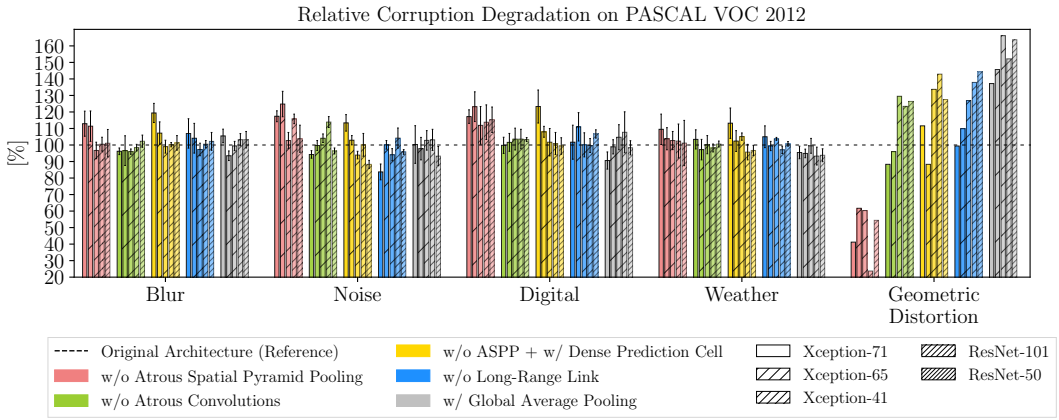


Figure A.3: Relative CD evaluated on PASCAL VOC 2012 for the proposed ablated variants of the DeepLabv3+ architecture with respect to image corruptions, employing five different network backbones. Each bar except for geometric distortion is averaged within a corruption category (error bars indicate the standard deviation). Bars above 100 % represent a relative decrease in performance compared to the respective reference architecture. Each ablated architecture is re-trained on the original training dataset. Removing ASPP decreases performance oftentimes significantly. The low rCD for geometric distortion indicates that the relative decrease of performance for this ablated variant is low. AC aids the robustness against geometric distortion for several backbones. The harming effect of DPC with respect to image corruptions is especially pronounced for Xception-71. The rCD of LRL is large against geometric distortion for ResNet-50. The rCD of GAP has, oftentimes, a contrary tendency as the CD. Best viewed in color.

	Blur				Noise					Digital				Weather				
Backbone	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
RN-50	**100.0**	100.0	100.0	100.0	**100.0**	**100.0**	**100.0**	**100.0**	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0
w/o ASPP	93.1	110.4	92.6	108.2	98.3	98.3	99.8	103.0	**119.8**	**128.6**	**111.6**	**111.1**	**109.9**	90.0	88.7	**122.6**	**103.6**	54.4
w/o AC	95.9	104.7	**103.6**	105.0	95.8	93.2	97.7	97.2	98.2	102.1	103.2	102.8	105.2	98.2	99.5	102.0	102.8	126.4
w/ DPC	95.1	105.0	99.2	105.7	90.7	89.4	89.7	87.7	83.8	106.3	98.4	92.2	100.5	98.1	94.0	93.6	101.2	127.5
w/o LRL	93.9	106.4	100.9	107.7	96.3	93.7	97.1	97.4	94.7	107.4	106.0	110.1	103.1	**100.5**	99.1	103.0	100.2	144.3
w/ GAP	96.9	**109.5**	99.8	**106.4**	98.2	90.8	99.1	95.2	83.2	95.3	100.1	93.5	104.5	98.7	95.1	88.1	93.2	**163.7**
RN-101	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	**100.0**	100.0	100.0	100.0
w/o ASPP	98.0	**106.3**	95.4	102.3	115.3	**119.7**	115.8	**117.3**	111.0	**119.4**	**107.8**	127.7	100.5	92.6	93.3	**117.7**	**106.2**	23.7
w/o AC	95.4	98.5	98.7	101.1	**115.7**	113.4	**116.6**	116.1	**107.8**	100.9	99.5	113.7	100.1	97.4	96.4	96.8	102.4	123.4
w/ DPC	102.0	99.9	**100.4**	98.5	101.0	111.1	99.9	99.1	89.4	105.3	92.7	109.4	96.1	95.8	99.1	88.1	99.9	142.9
w/o LRL	103.7	100.3	97.4	100.6	106.3	110.9	105.0	106.1	92.7	96.1	95.9	100.7	106.3	96.5	95.6	95.3	100.8	137.9
w/ GAP	**105.1**	105.7	96.8	**105.1**	105.0	108.7	105.2	106.3	90.6	99.6	95.7	**127.9**	**107.4**	92.9	99.9	85.0	95.4	**152.1**
XC-41	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	93.0	**104.0**	91.3	98.6	99.9	96.4	100.7	**110.4**	**106.0**	**127.2**	101.1	**118.5**	100.7	97.8	99.3	**111.8**	102.0	60.3
w/o AC	96.9	97.3	96.6	93.4	106.5	**101.1**	104.5	107.1	101.1	113.9	**104.2**	99.4	96.7	103.4	90.6	103.1	**103.6**	129.5
w/ DPC	**105.3**	98.9	94.8	97.2	94.2	91.0	91.7	97.0	95.5	112.9	103.9	99.9	89.9	**105.2**	**104.7**	108.4	102.2	133.7
w/o LRL	102.9	93.5	98.8	94.0	98.5	87.9	95.4	96.2	92.4	111.5	94.6	92.8	**101.3**	105.1	103.5	103.7	102.5	126.9
w/ GAP	95.0	102.8	**100.6**	99.3	**107.8**	93.7	**106.0**	108.7	97.9	115.4	100.4	107.8	95.3	100.6	92.0	103.5	101.9	**166.2**
XC-65	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	**100.0**	100.0	100.0	100.0
w/o ASPP	**100.1**	**124.9**	106.9	**113.8**	**126.4**	**120.7**	**130.2**	**134.3**	**112.4**	**124.8**	**109.0**	**133.9**	**125.0**	95.8	99.3	**113.0**	**107.3**	61.7
w/o AC	81.7	105.2	98.1	101.3	100.4	100.6	101.7	102.0	93.8	106.5	104.3	92.5	103.3	93.5	89.0	103.0	103.5	96.0
w/ DPC	96.9	115.7	**106.7**	109.3	106.5	101.1	104.7	103.5	98.4	110.5	107.2	111.3	102.9	96.7	98.1	112.9	101.8	88.3
w/o LRL	91.2	116.1	102.2	106.9	99.3	95.5	101.3	103.1	101.1	109.5	98.0	115.1	121.4	98.7	99.1	96.0	103.8	109.9
w/ GAP	89.7	97.1	91.9	95.4	98.5	91.1	103.8	106.6	89.3	92.7	97.5	101.6	103.9	95.5	90.6	97.8	95.6	**145.7**
XC-71	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	110.1	121.4	101.9	118.3	**114.5**	**123.5**	**116.5**	118.0	**114.6**	**117.1**	110.6	120.5	120.6	102.2	108.2	124.9	102.6	41.2
w/o AC	98.1	96.6	92.9	97.3	92.7	92.7	96.6	97.7	91.9	102.8	96.0	106.3	94.3	103.1	94.3	109.8	**106.6**	88.3
w/ DPC	**115.3**	**126.1**	**112.2**	**124.2**	105.6	111.6	115.1	**120.9**	113.5	115.8	115.9	**140.2**	**121.2**	**104.4**	**116.9**	**126.6**	104.9	**111.6**
w/o LRL	98.4	113.3	98.1	118.1	81.6	90.0	84.3	86.8	75.8	90.7	**118.6**	97.8	99.8	99.0	107.4	114.8	98.7	99.4
w/ GAP	98.5	108.2	107.9	107.5	93.7	118.8	101.0	103.7	84.2	84.3	88.1	91.6	98.3	93.9	102.0	92.3	93.7	137.2

Table A.7: Relative CD for corrupted variants of the validation set of PASCAL VOC 2012 for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Highest rCD per corruption is bold.

		Blur				Noise					Digital				Weather				
Backbone	Clean	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	33.1	16.1	16.6	14.9	16.5	12.1	11.5	12.4	17.0	24.7	27.2	14.8	26.5	25.1	7.8	18.5	20.1	10.7	28.3
w/o ASPP	27.3	12.2	11.3	10.5	11.6	9.6	9.6	9.9	13.3	19.0	22.1	10.8	20.8	19.5	5.7	15.6	14.5	7.8	22.9
w/o AC	32.1	15.2	15.9	14.2	15.7	11.2	11.5	11.4	15.6	23.0	27.1	13.7	25.1	25.1	7.6	18.8	19.2	10.7	27.9
w/ DPC	**34.7**	**17.3**	**18.9**	**15.6**	**18.0**	**13.9**	**13.7**	**13.8**	**18.4**	**25.8**	**28.9**	**15.8**	**27.8**	26.2	8.6	20.8	**21.5**	11.6	**29.4**
w/o LRL	32.2	15.7	16.5	14.3	16.3	12.9	11.6	13.2	17.3	24.4	26.7	14.2	25.1	24.6	7.5	18.9	19.7	10.7	27.6
w/ GAP	33.9	17.2	17.9	15.1	17.2	12.5	12.8	12.8	16.8	25.3	28.7	15.0	27.4	**26.6**	**8.7**	**20.9**	20.8	**11.7**	29.1
RN-50	37.4	18.0	19.7	16.9	19.2	14.1	12.8	14.4	19.4	28.5	31.1	18.0	30.1	29.5	8.8	21.5	23.9	13.6	32.9
w/o ASPP	29.7	13.5	13.8	11.6	13.5	11.1	10.1	11.6	15.5	21.6	24.8	13.3	23.4	22.7	6.7	17.0	17.6	9.9	25.5
w/o AC	36.5	18.2	19.3	16.6	18.7	13.7	11.8	13.8	18.8	27.5	30.2	17.3	29.1	28.7	7.9	20.4	23.3	12.8	31.3
w/ DPC	37.9	18.9	20.3	**17.5**	19.8	13.4	12.2	13.7	19.2	29.0	31.6	18.9	30.3	30.1	8.6	20.9	24.5	13.6	33.0
w/o LRL	36.6	18.3	19.8	16.1	18.8	13.6	12.2	13.8	18.9	27.4	31.0	19.1	30.1	29.3	8.1	21.2	24.3	13.4	32.0
w/ GAP	**38.2**	**19.3**	**21.1**	17.2	**19.9**	**15.5**	**12.8**	**15.8**	**21.3**	**30.4**	**32.9**	**19.5**	**31.7**	**30.8**	**9.9**	**23.2**	**25.8**	**14.9**	**33.0**
RN-101	38.1	19.1	20.6	17.3	19.8	15.4	14.6	15.7	20.7	28.8	31.6	19.7	31.2	31.4	10.2	22.9	25.6	14.0	32.8
w/o ASPP	30.7	14.3	14.1	12.8	14.2	13.3	11.8	13.7	17.7	23.4	25.9	14.4	24.7	24.1	7.3	18.5	18.8	10.7	26.2
w/o AC	37.3	18.3	19.9	16.9	19.0	14.4	14.4	14.7	19.4	27.5	31.4	18.1	30.1	30.5	9.4	22.9	24.6	13.6	32.2
w/ DPC	37.6	19.6	21.0	17.7	20.0	15.9	**15.1**	16.4	21.6	28.7	32.1	19.5	31.5	31.2	9.7	23.3	25.4	14.0	32.6
w/o LRL	37.5	18.9	20.5	17.7	19.9	16.5	14.6	**16.8**	**21.7**	29.0	31.6	19.8	30.7	30.1	9.8	22.2	25.9	14.0	32.2
w/ GAP	**39.3**	**20.2**	**21.7**	**17.9**	**20.6**	**15.9**	14.2	16.1	21.4	**29.9**	**33.2**	**20.4**	**32.8**	**32.8**	**10.8**	**23.3**	**27.0**	**15.6**	**34.2**
XC-41	39.7	22.1	22.7	17.4	**20.8**	20.8	18.1	20.5	24.8	**33.7**	34.2	20.9	32.5	32.6	13.0	25.0	28.4	17.0	**34.4**
w/o ASPP	35.4	19.4	20.0	15.3	18.4	18.2	16.3	17.9	21.7	29.2	30.3	17.7	28.5	28.3	11.2	22.7	23.5	14.4	31.3
w/o AC	38.4	21.8	22.2	**17.7**	20.6	21.8	18.3	21.1	25.0	32.9	33.4	20.0	31.7	32.0	12.2	24.8	26.4	15.9	33.0
w/ DPC	**40.2**	21.9	22.5	17.4	20.8	20.2	17.5	19.6	23.9	33.3	**34.8**	20.3	**32.6**	**32.9**	13.8	25.6	26.9	17.4	34.5
w/o LRL	39.1	21.4	22.6	17.2	20.6	20.8	17.6	20.5	25.0	32.6	34.1	21.1	32.1	32.2	13.8	25.5	27.1	16.9	34.2
w/ GAP	39.0	**22.7**	**22.9**	17.5	21.0	**21.9**	**18.5**	**21.6**	**25.4**	33.1	34.1	**21.6**	32.3	32.6	**14.3**	**25.8**	**28.9**	**17.7**	34.1
XC-65	41.4	23.4	25.2	18.9	22.7	23.2	19.8	22.9	27.1	35.4	36.1	23.5	34.8	**34.2**	14.8	27.7	30.0	18.4	35.6
w/o ASPP	40.2	21.4	23.3	18.1	21.6	20.4	16.7	20.1	24.7	33.0	34.1	21.4	32.6	31.4	12.1	25.3	27.4	15.6	35.1
w/o AC	40.0	22.7	24.4	18.6	22.1	23.6	20.9	22.8	26.4	34.6	35.1	23.4	33.9	33.3	13.9	27.2	29.1	17.8	35.0
w/ DPC	40.9	23.7	24.9	18.4	22.8	23.0	18.8	22.9	27.0	35.6	35.7	23.0	34.1	33.9	14.6	28.0	29.5	17.9	35.7
w/o LRL	41.0	23.2	25.0	18.6	22.7	24.2	19.8	23.9	27.6	35.6	35.7	23.5	34.1	33.8	14.9	27.5	29.3	19.0	**36.1**
w/ GAP	**41.7**	**23.9**	**25.6**	**19.2**	**23.4**	**24.7**	**20.8**	**24.2**	**28.1**	**36.2**	**36.1**	**23.8**	**35.0**	34.1	**15.4**	**28.2**	**30.5**	**20.1**	36.0
XC-71	42.4	**24.4**	26.4	19.5	23.9	**24.0**	**20.3**	**23.3**	**27.5**	**36.8**	**37.2**	**25.3**	**35.7**	**34.7**	16.1	**29.4**	31.3	19.8	**37.1**
w/o ASPP	40.6	21.9	24.2	17.5	21.9	20.8	16.6	20.0	24.2	34.0	34.8	22.5	33.1	32.4	12.9	26.3	28.9	16.5	35.2
w/o AC	41.8	24.3	25.4	**19.6**	23.6	24.0	19.9	22.9	26.2	35.7	36.1	23.2	34.8	33.7	15.7	28.3	29.9	19.7	35.8
w/ DPC	**42.5**	23.3	25.9	18.4	23.1	23.4	18.9	22.4	26.5	36.3	36.5	24.1	34.9	34.2	15.8	28.3	30.6	18.6	36.3
w/o LRL	42.2	22.9	25.9	18.7	23.5	21.7	18.7	21.1	25.4	35.5	36.3	24.3	34.5	34.0	15.1	28.6	30.6	19.7	36.4
w/ GAP	42.0	24.0	**26.6**	19.1	**24.0**	23.6	19.8	22.8	26.7	35.9	37.0	25.0	35.2	34.6	**16.7**	29.3	**31.6**	**20.9**	36.3

Table A.8: Mean IoU for clean and corrupted variants of the validation set of ADE20K for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Every mIoU is averaged over all available severity levels, except for corruptions of category noise where only the first three severity levels are considered. The standard deviation for image corruptions of category noise is 0.2 or less. Highest mIoU per corruption is bold.

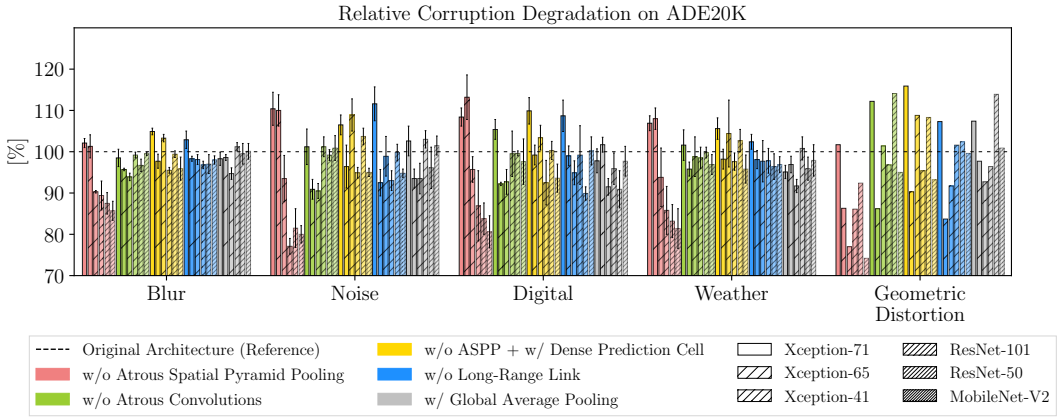


Figure A.4: Relative CD evaluated on ADE20K for the proposed ablated variants of the DeepLabv3+ architecture with respect to image corruptions, employing six different network backbones. Bars above 100 % represent a relative decrease in performance compared to the respective reference architecture. Each bar except for geometric distortion is averaged within a corruption category (error bars indicate the standard deviation). Each ablated architecture is re-trained on the original training dataset. Removing ASPP decreases performance oftentimes significantly. The low rCD for geometric distortion indicates that the relative decrease of performance for this ablated variant is low (except for Xception-71). The rCD of DPC and LRL are oftentimes highest for Xception-71. GAP increases the robustness for most backbones against many image corruptions. Best viewed in color.

	Blur				Noise					Digital				Weather				
Backbone	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**104.7**	**106.4**	**105.2**	**105.9**	**102.9**	**102.1**	**102.8**	**104.5**	**107.5**	**107.0**	**104.8**	**107.8**	**107.5**	**102.3**	**103.6**	**107.1**	**103.3**	**107.5**
w/o AC	101.1	100.9	100.8	101.0	101.1	100.0	101.1	101.7	102.1	100.1	101.3	102.0	100.0	100.2	99.6	101.1	100.0	100.6
w/ DPC	98.7	97.3	99.2	98.3	98.0	97.5	98.4	98.3	98.5	97.7	98.8	98.3	98.5	99.2	97.2	98.2	99.0	98.4
w/o LRL	100.6	100.1	100.8	100.3	99.1	99.8	99.1	99.6	100.3	100.8	100.8	102.0	100.7	100.4	99.5	100.6	100.0	101.1
w/ GAP	98.8	98.5	99.8	99.3	99.6	98.5	99.5	100.2	99.1	98.0	99.8	98.8	98.0	99.0	97.0	99.2	98.9	99.0
RN-50	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**105.4**	**107.4**	**106.4**	**107.1**	**103.5**	**103.1**	**103.3**	**104.8**	**109.7**	**109.2**	**105.8**	**109.5**	**109.6**	**102.2**	**105.8**	**108.3**	**104.3**	**111.0**
w/o AC	99.7	100.6	100.5	100.6	100.5	101.1	100.8	100.6	101.4	101.3	100.9	101.4	101.2	101.0	101.5	100.8	100.9	102.3
w/ DPC	98.9	99.3	99.3	99.4	100.8	100.7	100.9	100.2	99.3	99.3	98.9	99.7	99.2	100.1	100.8	99.2	100.0	99.8
w/o LRL	99.6	99.9	101.0	100.6	100.6	100.7	100.7	100.5	101.6	100.1	98.7	99.9	100.3	100.7	100.4	99.5	100.3	101.3
w/ GAP	98.4	98.3	99.7	99.2	98.4	100.0	98.4	97.7	97.4	97.4	98.1	97.7	98.1	98.7	97.9	97.5	98.5	99.7
RN-101	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**105.9**	**108.1**	**105.5**	**107.0**	**102.5**	**103.2**	**102.3**	**103.8**	**107.5**	**108.4**	**106.5**	**109.4**	**110.6**	**103.2**	**105.7**	**109.2**	**103.9**	**109.9**
w/o AC	101.0	100.8	100.5	101.0	101.2	100.2	101.2	101.6	101.8	100.3	101.9	101.5	101.3	100.9	100.0	101.4	100.5	101.0
w/ DPC	99.3	99.4	99.6	99.7	99.4	99.3	99.1	98.9	100.1	99.3	100.3	99.6	100.4	100.6	99.5	100.3	100.1	100.3
w/o LRL	100.2	100.1	99.6	99.9	98.7	100.0	98.7	98.8	99.6	100.0	99.9	100.7	101.9	100.5	100.9	99.6	100.0	101.0
w/ GAP	98.7	98.6	99.3	98.9	99.5	100.4	99.5	99.2	98.3	97.6	99.1	97.6	97.9	99.4	99.5	98.1	98.1	98.0
XC-41	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**103.4**	**103.4**	**102.5**	**103.0**	**103.4**	**102.3**	**103.3**	**104.1**	**106.9**	**105.9**	**104.0**	**105.9**	**106.4**	**102.1**	**103.1**	**106.9**	**103.1**	**104.7**
w/o AC	100.3	100.6	99.6	100.3	98.7	99.8	99.2	99.8	101.3	101.2	101.1	101.1	100.9	100.9	100.3	102.8	101.2	102.1
w/ DPC	100.3	100.2	99.9	100.0	100.8	100.8	101.2	101.2	100.7	99.1	100.8	99.8	99.6	99.1	99.3	102.2	99.4	99.9
w/o LRL	100.8	100.1	100.2	100.2	100.0	100.7	100.0	99.7	101.7	100.1	99.8	100.6	100.6	99.1	99.4	101.8	100.0	100.3
w/ GAP	99.2	99.6	99.9	99.7	98.6	99.5	98.5	99.2	101.0	100.1	99.1	100.3	100.0	98.5	99.0	99.4	99.2	100.4
XC-65	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**102.7**	**102.6**	**101.0**	**101.5**	**103.5**	**104.0**	**103.7**	**103.2**	**103.6**	**103.2**	**102.8**	**103.4**	**104.2**	**103.2**	**103.3**	**103.8**	**103.5**	100.8
w/o AC	100.9	101.0	100.4	100.8	99.4	98.7	100.1	100.9	101.1	101.6	100.2	101.3	101.2	101.0	100.7	101.3	100.7	**100.9**
w/ DPC	99.6	100.3	100.7	99.9	100.2	101.3	100.1	100.2	99.6	100.6	100.6	101.1	100.4	100.2	99.6	100.7	100.6	99.9
w/o LRL	100.3	100.2	100.4	100.0	98.6	100.1	98.8	99.2	99.6	100.5	100.1	101.1	100.6	99.9	100.2	101.0	99.3	99.3
w/ GAP	99.4	99.4	99.7	99.2	98.0	98.8	98.3	98.6	98.7	99.9	99.6	99.7	100.1	99.3	99.3	99.3	97.9	99.5
XC-71	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	**103.3**	**103.1**	**102.6**	**102.6**	**104.3**	**104.6**	**104.3**	**104.6**	**104.5**	**103.8**	**103.8**	**103.9**	**103.6**	**103.8**	**104.4**	**103.6**	**104.1**	**103.0**
w/o AC	100.2	101.4	100.0	100.4	100.0	100.5	100.6	101.8	101.8	101.7	102.8	101.3	101.5	100.5	101.5	102.1	100.1	102.1
w/ DPC	101.4	100.8	101.4	101.0	100.7	101.8	101.2	101.4	100.9	101.1	101.7	101.3	100.7	100.4	101.5	101.1	101.5	101.3
w/o LRL	101.9	100.7	101.0	100.6	103.0	102.0	103.0	102.9	102.1	101.3	101.5	101.9	101.1	101.3	101.0	101.0	100.2	101.1
w/ GAP	100.5	99.7	100.5	99.9	100.5	100.6	100.8	101.1	101.5	100.2	100.4	100.8	100.1	99.3	100.1	99.6	98.6	101.3

Table A.9: CD for corrupted variants of the validation set of ADE20K for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Highest CD per corruption is bold.

	Blur				Noise					Digital				Weather				
Backbone	Motion	Defocus	Glass	Gaussian	Gaussian	Impulse	Shot	Speckle	Intensity	Brightness	Contrast	Saturate	JPEG	Snow	Spatter	Fog	Frost	Distortion
MN-V2	100.0	**100.0**	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0	**100.0**	100.0	100.0	**100.0**	**100.0**	**100.0**	100.0	**100.0**	100.0
w/o ASPP	82.3	88.6	85.4	86.6	79.5	77.4	78.9	80.2	83.8	74.0	83.8	81.9	82.9	81.0	74.5	88.0	82.0	74.2
w/o AC	**100.1**	99.4	98.9	99.6	100.0	96.5	100.2	102.4	**105.5**	91.8	100.9	105.0	92.8	97.7	93.3	100.0	96.7	95.0
w/ DPC	97.2	91.7	99.2	95.5	95.1	93.4	96.4	95.5	94.9	88.5	97.8	93.2	95.0	99.2	90.7	94.8	98.5	93.2
w/o LRL	98.5	96.7	99.4	97.5	93.6	96.3	93.8	94.5	95.7	97.7	99.3	**106.0**	97.7	98.6	93.8	98.1	97.1	99.6
w/ GAP	99.0	97.8	**102.8**	**100.9**	**101.7**	97.9	**101.7**	**105.0**	101.4	94.0	**102.8**	99.3	94.6	99.8	91.4	**101.0**	99.3	**100.9**
RN-50	**100.0**	**100.0**	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	**100.0**	100.0	100.0	**100.0**	100.0	100.0
w/o ASPP	83.1	89.8	88.1	89.0	79.6	79.7	78.6	78.6	90.9	77.6	84.5	85.4	87.9	80.2	79.9	89.8	83.1	92.4
w/o AC	94.1	97.4	97.4	97.8	98.0	100.1	98.9	97.8	101.1	100.0	99.0	100.4	99.0	99.9	101.6	98.0	99.6	114.1
w/ DPC	98.0	99.6	99.8	100.0	**105.0**	**104.5**	**105.5**	**103.9**	99.8	**100.5**	98.1	**103.8**	98.9	**102.3**	**107.0**	99.4	**102.1**	108.3
w/o LRL	94.3	95.1	100.0	98.4	98.8	99.4	99.2	98.0	**103.8**	88.5	90.4	88.3	92.4	99.6	97.1	91.4	97.6	102.4
w/ GAP	97.3	97.1	**102.9**	**100.8**	97.6	103.2	97.6	94.1	88.1	84.4	96.4	89.5	93.5	98.9	94.7	91.8	98.0	**113.9**
RN-101	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0
w/o ASPP	86.3	94.7	86.3	90.3	76.9	80.3	75.8	74.7	77.9	74.6	88.3	86.3	98.5	83.9	80.6	95.7	83.2	86.1
w/o AC	99.9	99.1	98.2	99.8	100.8	97.1	100.8	102.7	**104.5**	90.1	**103.8**	**102.8**	**101.3**	99.9	94.4	**101.4**	98.2	96.8
w/ DPC	94.8	94.8	96.0	96.3	95.7	95.6	94.7	92.4	96.1	85.5	98.6	89.1	96.8	100.1	94.2	97.8	98.3	95.4
w/o LRL	98.0	97.1	95.6	96.4	92.6	97.6	92.6	91.3	90.8	91.2	96.5	98.6	110.6	99.6	101.1	93.0	97.6	**101.6**
w/ GAP	**100.5**	**100.0**	**102.6**	**101.6**	**103.1**	**106.2**	**103.2**	**102.9**	99.6	92.8	102.3	92.9	95.6	**102.3**	**104.8**	98.2	98.1	96.4
XC-41	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**	100.0	100.0	100.0	**100.0**	**100.0**	100.0	100.0	100.0
w/o ASPP	90.8	90.3	90.2	89.8	91.4	88.7	91.1	91.9	104.6	92.7	93.8	95.5	100.7	90.7	86.6	105.6	92.3	77.0
w/o AC	94.0	95.1	92.5	94.1	87.8	92.9	89.7	90.1	92.1	90.7	97.7	92.2	90.2	98.0	92.5	106.1	98.7	101.4
w/ DPC	**104.3**	**104.2**	**102.2**	**102.6**	**106.1**	**105.4**	**107.6**	**109.5**	**116.2**	98.5	**106.0**	**105.5**	**103.4**	99.0	100.1	**118.5**	**100.2**	**108.8**
w/o LRL	100.0	96.9	98.0	97.4	96.5	99.5	96.5	94.2	107.8	89.9	95.6	97.0	97.1	94.7	92.8	105.9	97.3	91.7
w/ GAP	92.8	94.4	96.6	95.1	90.7	95.1	90.4	91.3	100.1	88.6	92.6	93.8	90.9	92.5	90.3	90.2	94.0	92.7
XC-65	100.0	100.0	**100.0**	**100.0**	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**100.0**
w/o ASPP	**104.3**	**103.8**	97.9	99.3	**107.8**	**108.8**	**108.5**	**107.5**	**117.5**	**113.6**	**104.9**	**114.0**	**120.2**	**105.3**	**108.0**	**112.2**	**106.6**	86.3
w/o AC	96.0	95.8	95.1	95.8	89.8	88.7	92.8	94.7	88.5	92.3	92.8	91.9	91.8	98.0	93.3	95.6	96.3	86.2
w/ DPC	95.4	98.3	100.0	96.9	97.9	102.6	97.6	97.1	86.7	97.4	99.8	102.8	96.7	98.7	94.1	100.0	99.8	90.3
w/o LRL	98.6	97.9	99.2	97.5	91.4	98.0	92.2	92.6	88.2	97.0	97.6	103.0	98.5	97.9	97.5	101.6	95.4	83.7
w/ GAP	98.6	98.6	99.7	97.7	92.7	96.6	94.2	94.4	89.7	102.8	99.6	100.4	104.0	98.7	97.9	97.6	93.5	97.7
XC-71	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
w/o ASPP	103.6	102.7	101.1	100.8	107.6	**108.3**	107.8	109.9	118.3	**110.7**	106.0	110.5	**106.5**	**105.3**	**109.8**	105.8	**106.6**	101.7
w/o AC	97.1	102.1	97.0	98.0	96.2	98.7	98.9	104.1	108.0	107.2	**108.3**	102.6	103.7	99.2	103.0	107.0	97.3	112.2
w/ DPC	106.0	**103.9**	**105.1**	**104.4**	103.2	106.7	105.1	107.0	110.6	113.7	107.4	112.5	106.1	101.4	108.2	**107.2**	105.5	**115.9**
w/o LRL	**106.5**	101.7	102.3	101.0	**110.7**	105.9	**110.4**	**112.4**	**118.5**	110.3	104.8	**114.1**	105.5	103.0	103.5	103.7	99.4	107.3
w/ GAP	99.7	96.1	100.0	97.3	100.0	100.3	100.9	102.3	109.8	94.5	99.5	101.6	95.4	96.1	97.3	93.8	93.2	107.4

Table A.10: Relative CD for corrupted variants of the validation set of ADE20K for several network backbones of the DeepLabv3+ architecture and respective architectural ablations. Highest rCD per corruption is bold.

A.3 Implementation of Painting-by-Numbers in TensorFlow

The following code illustrates a function that implements Painting-by-Numbers in TensorFlow through supplying an image label pair of the mini-batch.

```
import tensorflow as tf

def pbn(image_in,
        label,
        num_classes,
        alpha_min=None,
        alpha_max=None):

  """
  Removes texture of input images through
  generating a randomized texture-free image.

  Args:
  image_in: A 3D tensor of shape [height, width, 3].
  label: A 3D tensor of shape [height, width, 1] (default)
  num_classes: (scalar) number of classes. 19 for Cityscapes
  alpha_min: lower bound of alpha_blending
  alpha_max: upper bound of alpha_blending

  Returns:
  outputs the image processed by Painting-by-Numbers

  Raises:
  ValueError: If num_classes is none
  """

  """ Following dependencies must be set """
  if num_classes is None:
  raise ValueError('num classes must not be None.')

  bottom_val = 0  # sRGB
  top_val = 255  # sRGB
```

```
random_value = tf.random_uniform([])

def remove():

  # create empty tensor
  tensor = tf.clip_by_value(image_in, 0, 0)

  for i in range(num_classes):
    # create random RGB
    random_color_values = tf.random_uniform([1, 1, 3],
                                            bottom_val,
                                            top_val,
                                            dtype=tf.float32)
    # boolean mask for i-th class
    mask_boolean = tf.equal(label, i)
    # numeric mask for i-th class
    mask_numeric = tf.cast(mask_boolean, tf.float32)

    # Painting-by-Numbers
    concat_numeric_mask = tf.concat([mask_numeric,
                                     mask_numeric,
                                     mask_numeric],
                                     axis=2)
    random_color_class = concat_numeric_mask * random_color_values
    tensor += random_color_class

  # draw alpha and alpha-blend
  alpha = tf.random_uniform([], alpha_min, alpha_max,
                            dtype=tf.float32)
  # alpha blend
  tensor = tf.multiply(alpha, tensor) + (1 - alpha) * image_in

  return tensor

texture_removed = tf.less_equal(random_value, 1)
outputs = tf.cond(texture_removed, remove, lambda: image_in)

return outputs
```

List of Tables

List of Figures

Bibliography

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Ahmet Ouz Akyz and Erik Reinhard. Noise reduction in high dynamic range imaging. *Journal of Visual Communication and Image Representation*, 18(5):366 – 376, 2007.

[3] A. Arnab, O. Miksik, and P. H. S. Torr. On the Robustness of Semantic Segmentation Models to Adversarial Attacks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 888–897, 2018.

[4] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20(184):1–25, 2019.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[6] A. A. Bell, C. Seiler, J. N. Kaftan, and T. Aach. Noise in high dynamic range imaging. In *2008 15th IEEE International Conference on Image Processing*, pages 561–564, 2008.

[7] M. Bigas, E. Cabruja, J. Forest, and J. Salvi. Review of CMOS image sensors. *Microelectronics Journal*, 37(5):433 – 451, 2006.

[8] Michael Brading, Brian Keelan, and Hieu Tran. Image Sensors for Camera Monitor Systems. In Anestis Terzis, editor, *Handbook of Camera Monitor Systems: The Automotive Mirror-Replacement Technology based on ISO 16505*, pages 175–201. Springer International Publishing, Cham, 2016. DOI: 10.1007/978-3-319-29611-1_5.

[9] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 3–14, New York, NY, USA, 2017. ACM.

[10] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[11] Boncelet Charles. 4.5 - Image Noise Models. In AL BOVIK, editor, *Handbook of Image and Video Processing (Second Edition)*, Communications, Networking and Multimedia, pages 397

– 409. Academic Press, Burlington, second edition, 2005. DOI: 10.1016/B978-012119792-6/50087-5.

[12] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[13] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for Efficient Multi-Scale Architectures for Dense Image Prediction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 8699–8710. Curran Associates, Inc., 2018.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume abs/1412.7062, 2015.

[15] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision ECCV 2018*, pages 833–851, Cham, 2018. Springer International Publishing.

[17] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[18] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval Networks: Improving Robustness to Adversarial Examples. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, International Convention Centre, Sydney, Australia, 2017. PMLR.

[19] Tomer Cohen, Noy Shulman, Hai Morgenstern, Roey Mechrez, and Erez Farhan. Self-Supervised Dynamic Networks for Covariate Shift Robustness. *arXiv preprint arXiv:2006.03952*, 2020.

[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[21] E. D. Cubuk, B. Zoph, D. Man, V. Vasudevan, and Q. V. Le. AutoAugment: Learning Augmentation Strategies From Data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019.

[22] Ekin D Cubuk, Barret Zoph, Samuel S Schoenholz, and Quoc V Le. Intriguing Properties of Adversarial Examples. *arXiv preprint arXiv:1711.02846*, 2017.

[23] D. Dai and L. V. Gool. Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3819–3824, 2018.

[24] Paul E. Debevec and Jitendra Malik. Recovering High Dynamic Range Radiance Maps from Photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[25] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[26] Terrance DeVries and Graham W Taylor. Improved Regularization of Convolutional Neural Networks with CutOut. *arXiv preprint arXiv:1708.04552*, 2017.

[27] S. Dodge and L. Karam. Understanding how image quality affects deep neural networks. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2016.

[28] S. Dodge and L. Karam. A Study and Comparison of Human and Deep Learning Recognition Performance under Visual Distortions. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2017.

[29] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[30] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the Landscape of Spatial Robustness. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of Machine Learning Research*, volume 97, pages 1802–1811, Long Beach, California, USA, June 2019. PMLR.

[31] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[32] H. Faraji and W. J. MacLean. CCD noise removal in digital images. *IEEE Transactions on Image Processing*, 15(9):2676–2685, 2006.

[33] Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 106.1–106.13. BMVA Press, Sept. 2015.

[34] Fei-Fei Li & Justin Johnson & Serena Yeung. Lecture 11: Detection and Segmentation, May 2017.

[35] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[36] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy

and robustness. In *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2019.

[37] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schtt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 7538–7550. Curran Associates, Inc., 2018.

[38] Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin Cubuk. Adversarial Examples Are a Natural Consequence of Test Error in Noise. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of Machine Learning Research*, volume 97, pages 2280–2289, Long Beach, California, USA, June 2019. PMLR.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[40] Miguel Granados, Tun Ozan Aydn, J Rafael Tena, Jean-Franois Lalonde, and Christian Theobalt. HDR image noise estimation for denoising tone mapped images. In *Proceedings of the 12th European Conference on Visual Media Production*, pages 1–8, 2015.

[41] K. Grauman and T. Darrell. The pyramid match kernel: discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1458–1465 Vol. 2, 2005.

[42] Shixiang Gu and Luca Rigazio. Towards Deep Neural Network Architectures Robust to Adversarial Examples. *arXiv preprint arXiv:1412.5068*, 2014.

[43] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973.

[44] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. DOI: 10.1017/CBO9780511811685.

[45] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[46] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[47] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[48] G. E. Healey and R. Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, 1994.

[49] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[50] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A Simple Data Processing Method to Improve Robustness and

Uncertainty. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[51] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, 2012.

[52] M. Holschneider, R. Kronland-Martinet, J. Morlet, and Ph. Tchamitchian. A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform. In Jean-Michel Combes, Alexander Grossmann, and Philippe Tchamitchian, editors, *Wavelets*, pages 286–297, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[53] A. Howard, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le. Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[54] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[55] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[56] T Huang. Computer Vision: Evolution And Promise. 1996.

[57] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety Verification of Deep Neural Networks. In Rupak Majumdar and Viktor Kunak, editors, *Computer Aided Verification*, pages 3–29, Cham, 2017. Springer International Publishing.

[58] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Francis Bach and David Blei, editors, *Proceedings of Machine Learning Research*, volume 37, pages 448–456, Lille, France, July 2015. PMLR.

[59] Bernd Jähne. *Digital Image Processing: Concepts, Algorithms, and Scientific Applications*. Springer-Verlag, Berlin, Heidelberg, 4th edition, 1997.

[60] Christoph Kamann and Carsten Rother. Benchmarking the Robustness of Semantic Segmentation Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[61] Christoph Kamann and Carsten Rother. Benchmarking the Robustness of Semantic Segmentation Models with Respect to Common Corruptions. *International Journal of Computer Vision*, pages 1–22, 2020.

[62] Christoph Kamann and Carsten Rother. Increasing the Robustness of Semantic Segmentation Models with Painting-by-Numbers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision ECCV 2020*, pages 369–387, Cham, 2020. Springer International Publishing.

[63] Christoph Kamann and Carsten Rother. Supplementary Material for Benchmarking the Robustness of Semantic Segmentation Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[64] Christoph Kamann and Carsten Rother. Supplementary Material for: Increasing the Robustness of Semantic Segmentation Models with Painting-by-Numbers. In Andrea Vedaldi,

Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision ECCV 2020*, Cham, 2020. Springer International Publishing.

[65] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial Logit Pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[66] Andrej Karpathy. *Connecting images and natural language*. PhD thesis, Stanford University, 2016.

[67] T. Ke, M. Maire, and S. X. Yu. Multigrid Neural Architectures. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4067–4075, 2017.

[68] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[69] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[70] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollr. Panoptic Segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9396–9405, 2019.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[72] Jan Laermann, Wojciech Samek, and Nils Strodthoff. Achieving Generalizable Robustness of Deep Neural Networks by Stability Training. In Gernot A. Fink, Simone Frintrop, and Xiaoyi Jiang, editors, *Pattern Recognition*, pages 360–373, Cham, 2019. Springer International Publishing.

[73] Steven M. LaValle. Geometric Representations and Transformations. In *Planning Algorithms*, pages 66–104. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877.005.

[74] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[75] J. W. Lee, R. Park, and S. Chang. Noise reduction and adaptive contrast enhancement for local tone mapping. *IEEE Transactions on Consumer Electronics*, 58(2):578–586, 2012.

[76] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[77] Wei Liu, Andrew Rabinovich, and Alexander C Berg. ParseNet: Looking Wider to See Better. *arXiv preprint arXiv:1506.04579*, 2015.

[78] A. LOBO, O. CHIC, and A. CASTERAD. Classification of Mediterranean crops with multisensor data: per-pixel versus per-object statistics and image segmentation. *International Journal of Remote Sensing*, 17(12):2385–2400, 1996.

[79] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[80] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D Cubuk. Improving Robustness Without Sacrificing Accuracy with Patch Gaussian Augmentation. *arXiv preprint arXiv:1906.02611*, 2019.

[81] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision ECCV 2018*, pages 185–201, Cham, 2018. Springer International Publishing.

[82] David Marr. Vision: A computational investigation into the human representation and processing of visual information, henry holt and co. *Inc., New York, NY*, 2(4.2), 1982.

[83] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[84] 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. In *Machine Learning for Autonomous Driving Workshop, NeurIPS 2019*, volume 190707484, July 2019.

[85] A. Milioto, P. Lottes, and C. Stachniss. Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2229–2235, 2018.

[86] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 374–380 Vol. 1, 1999.

[87] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, Haifa, Israel, June 2010. Omnipress.

[88] Junichi Nakamura. *Image Sensors and Signal Processing for Digital Still Cameras*. CRC press, 2017.

[89] H. Noh, S. Hong, and B. Han. Learning Deconvolution Network for Semantic Segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, 2015.

[90] A Emin Orhan. Robustness properties of Facebook's ResNeXt WSL models. *arXiv preprint arXiv:1907.07640*, 2019.

[91] G. Papandreou, I. Kokkinos, and P. Savalle. Modeling local and global deformations in Deep Learning: Epitomic convolution, Multiple Instance Learning, and sliding window detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399, 2015.

[92] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[93] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient Neural Architecture Search via Parameters Sharing. In Jennifer Dy and Andreas Krause, editors, *Proceedings of Machine Learning Research*, volume 80, pages 4095–4104, Stockholmsmssan, Stockholm Sweden, July 2018. PMLR.

[94] Haozhi Qi, Zheng Zhang, Bin Xiao, Han Hu, Bowen Cheng, Yichen Wei, and Jifeng Dai. Deformable convolutional networkscoco detection and segmentation challenge 2017 entry. In *ICCV COCO Challenge Workshop*, volume 15, 2017.

[95] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151, 1999.

[96] M. Rahnemoonfar, R. Murphy, M. V. Miquel, D. Dobbs, and A. Adams. Flooded Area Detection from Uav Images Based on Densely Connected Recurrent Neural Networks. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1788–1791, 2018.

[97] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[98] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[99] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 309–314, Los Angeles, California, 2004. Association for Computing Machinery.

[100] Avraham Ruderman, Neil C Rabinowitz, Ari S Morcos, and Daniel Zoran. Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs. *arXiv preprint arXiv:1804.04438*, 2018.

[101] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[102] Evgenia Rusak, Lukas Schott, Roland S Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A simple way to make neural networks robust against diverse image corruptions. *arXiv preprint arXiv:2001.06057*, 2020.

[103] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and others. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[104] C. Sakaridis, D. Dai, and L. Van Gool. Guided Curriculum Model Adaptation and Uncertainty-Aware Evaluation for Semantic Nighttime Image Segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7373–7382, 2019.

[105] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic Foggy Scene Understanding with Synthetic Data. *International Journal of Computer Vision*, 126(9):973–992, 2018.

[106] Shishir Shah and JK Aggarwal. Intrinsic parameter calibration procedure for a (high-distortion) fish-eye lens camera with distortion model and accuracy estimation. *Pattern Recognition*, 29(11):1775–1788, 1996.

[107] Neeraj Sharma and Lalit M Aggarwal. Automated medical image segmentation techniques. *Journal of Medical Physics*, 35(1):3–14, 2010.

[108] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[109] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[110] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, New York, 1st edition, 2010.

[111] R. Takahashi, T. Matsubara, and K. Uehara. Data Augmentation Using Random Image Cropping and Patching for Deep CNNs. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2917–2931, 2020.

[112] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler. Gated-SCNN: Gated Shape CNNs for Semantic Segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5228–5237, 2019.

[113] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of CCD imaging process. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 480–487 vol.1, 2001.

[114] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the Impact of Blur on Recognition by Convolutional Networks. *arXiv preprint arXiv:1611.05760*, 2016.

[115] G. Volk, S. Mller, A. v Bernuth, D. Hospach, and O. Bringmann. Towards Robust CNN-based Object Detection through Augmentation with Synthetic Rain Variations. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 285–292, 2019.

[116] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. *Pattern Recognition*, 90:119 – 133, 2019.

[117] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-Training With Noisy Student Improves ImageNet Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2020.

[118] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A Fourier Perspective on Model Robustness in Computer Vision. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alch-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13276–13286. Curran Associates, Inc., 2019.

[119] Ian T Young, Jan J Gerbrands, and Lucas J Van Vliet. *Fundamentals of image processing*, volume 841. Delft University of Technology Delft, 1998.

[120] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[121] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031, 2019.

[122] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[123] Oliver Zendel, Katrin Honauer, Markus Murschitz, Daniel Steininger, and Gustavo Fernndez Domnguez. WildDash - Creating Hazard-Aware Benchmarks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision ECCV 2018*, pages 407–421, Cham, 2018. Springer International Publishing.

[124] Oliver Zendel, Markus Murschitz, Martin Humenberger, and Wolfgang Herzner. How Good Is My Test Data? Introducing Safety Analysis for Computer Vision. *International Journal of Computer Vision*, 125(1):95–109, 2017.

[125] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond Empirical Risk Minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[126] L. Zhang, M. Yu, T. Chen, Z. Shi, C. Bao, and K. Ma. Auxiliary Training: Towards Accurate and Robust Models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 369–378, 2020.

[127] Richard Zhang. Making Convolutional Networks Shift-Invariant Again. In *International Conference on Machine Learning*, 2019.

[128] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision ECCV 2018*, pages 418–434, Cham, 2018. Springer International Publishing.

[129] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017.

[130] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the Robustness of Deep Neural Networks via Stability Training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4480–4488, 2016.

[131] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random Erasing Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):13001–13008, Apr. 2020.

[132] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene Parsing through ADE20k Dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017.

[133] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic Understanding of Scenes Through the ADE20k Dataset. *International Journal of Computer Vision*, 127(3):302–321, 2016.

[134] Y. Zhou, S. Song, and N. Cheung. On classification of distorted images with deep convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1213–1217, 2017.

[135] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. *arXiv preprint arXiv: 1611.01578*, 2017.

[136] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.