Mathematisch-
Naturwissenschaftliche
Fakultät

Lernbasierte Computer Vision

Bachelorarbeit

Object Detection using the Scattering Transform

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Autonomous Computer Vision
Marius Hobbhahn, marius.hobbhahn@student.uni-tuebingen.de, 2018/19

Bearbeitungszeitraum: von-20.02.19 bis 20.06.19

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter: Dr. Benjamin Coors, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Marius Hobbhahn (Matrikelnummer 4003731), June 12, 2019

Abstract

TODO

Zusammenfassung

TODO

Contents

1 Introduction

Object detection describes the task of detecting instances of semantic objects in visual data, i.e. images and videos in two or three dimensions respectively. Even though the task is very easy for humans in most situations it is very hard for computers. However, in recent years object detection algorithms have gotten significantly better for many different applications like face recognition, object tracking (e.g. the ball in a football match) and especially semantic segmentation of traffic scenes, pedestrian and car tracking.

For most state of the art (SOTA) object detection algorithm convolutional neural networks (CNNs) are used. In many implementations the filters used in those CNNs are all trained during the training period. [BM12] introduced a new technique called the Scattering Transform that uses wavelet operations on the image and performs classification tasks on those. They also show that the technique is essentially equivalent to using CNNs with fixed weights for some or all filters. It has been applied successfully in a variety of tasks. [SM13] showed that the scattering transform is applicable to texture discrimination. [OM14] have demonstrated that the scattering transform also produces results similar to other SOTA algorithms for unsupervised learning. [ACC+17] improved the classification of diseases from neuroimages considerably. Lastly, [OBZ17] shows that substituting the first layer filters of CNN approaches with the scattering transform yields equivalent results compared to these filters being trained.
The reason why the scattering transform has proven so successful are the properties it provides. It is invariant to deformation and equivariant to translation. In this work it is also argued that it has desirable properties w.r.t rotation and scaling. These properties are important for image classification but also necessary for object detection. For example, when detecting pedestrians in real traffic situations, the object detection algorithm must be able to identify them independent of their location, size or rotation within the image.
This work tries to harvest the useful properties of the scattering transform and

combine it with already established state of the art object detection algorithms. This will be done primarily in two ways. First, the techniques are combined sequentially, i.e. the SOTA algorithms are applied only to the outputs of the scattering transform. [OBZ17] have already shown that sequential combination is able to produce SOTA results for image recognition. This is the attempt to reproduce these findings for object detection.
Second, the techniques are combined in parallel, i.e. the information of the scattering transform are used as additional inputs for the object detection algorithm or merged at later stages. This has not been tested yet and is the primary extension of the just described related work.
If the approaches are successful, three specific advantages arise. First, the scattering transform might yield information that was currently not available to the network and therefore increasing its accuracy. Even if that might only be a marginal increase, it is meaningful for application. Every little reduction of the error in object detection, especially for autonomous driving, means a reduction of risk of self driving cars. This is directly translated to lives being saved in the longterm. Second, fixed weights imply no additional training time for them. If, for example, one layer can be substituted that would reduce the length of training and save cost and energy while creating access for people who currently do not own multiple GPUs. Third, fixed weights cannot be overfit and are maximally general. This might produce more robust algorithms and protect against black box attacks or other malicious practices applied to CNNs. This, however, will not be tested within the scope of this work but might be interesting follow-up.

2 Theory

This chapter provides an introduction to all relevant basis for this work. If you are already familiar with image processing and convolutional neural networks you can skip to 2.5.

2.1 2D Image Processing

Image processing describes the application of different algorithms on images with the purpose of gaining certain information about it or changing its representation. Most of the time images are given as two dimensional pixel arrays where each entry denotes the intensity of that pixel. In the case of grayscale images the value is between 0 and 255 representing black and white respectively. When handling color images an additional 3rd dimension is added with three channels representing a red, green, blue (rgb) encoding. Each entry, again, has values between 0 and 255 representing color intensity.
Instead of imagining an image as a flat 2D object, it can also be seen as a terrain with surface, where the height of each coordinate is determined by the intensity of its value. An example of this is shown in figure 2.1.

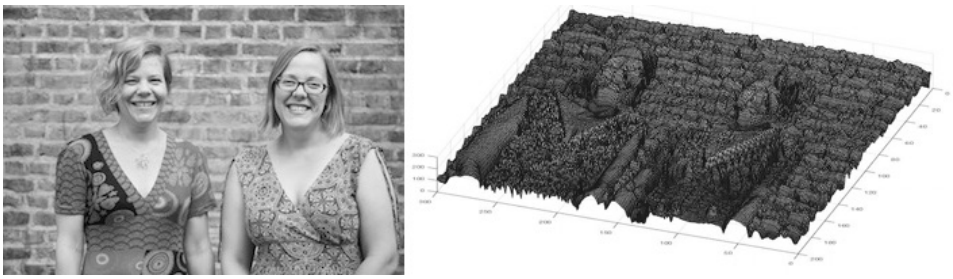


Figure 2.1: Left: image represented as 2D flat surface. Right: image as 3D terrain with uneven surface. [1]

[1] Figure taken from `https://plus.maths.org/content/fourier-transforms-images`

Like every other surface, these images can now be approximated as the sum of many different two dimensional sine waves. 2D sine waves are defined as in equation 2.1, where a is the amplitude and h, k are the frequencies in x and y direction respectively.

$$f = a\sin(h \cdot x + k \cdot y) \tag{2.1}$$

To give an example of how this approximation looks like, figure 2.2 shows examples of three different two dimensional sine waves. It can be observes that higher amplitudes dominate the resulting wave, i.e. determine the direction of the wave stronger than the smaller amplitudes.

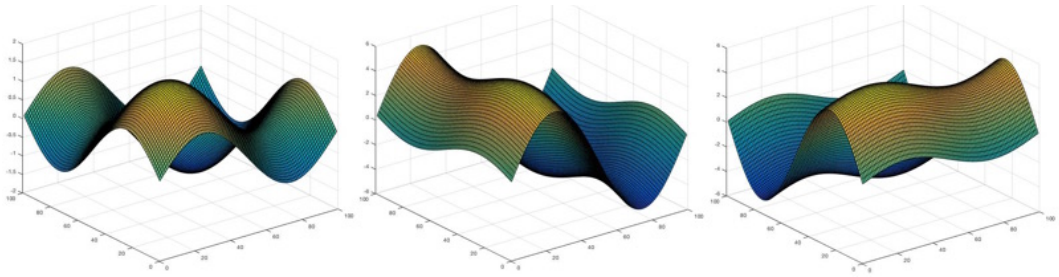


Figure 2.2: Left: $\sin(x)+\sin(y)$. Middle: $5\sin(x)+\sin(y)$. Right: $\sin(x)+5\sin(y)$. On the middle and right images the higher amplitudes of 5 dominate the resulting wave. [2]

2.2 Fourier Transformation

A Fourier Transform (FT) decomposes a signal into the frequencies that make it up.

2.2.1 One dimensional FT

In the case of one dimensional signals the decomposition are the coefficients of the sine waves representing the signal. A good example of this would be the decomposition of an audio signal. The FT is defined by equation 2.2 for any real number ω and any integrable function $f : \mathbb{R} \to \mathbb{C}$.

$$\tilde{f}(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \omega} dx \tag{2.2}$$

[2] Figure taken from `https://plus.maths.org/content/fourier-transforms-images`

To get back to the Fourier domain when given a frequency, the inverse Fourier transform defined in equation 2.3 is used.

$$f(x) = \int_{-\infty}^{\infty} \tilde{f}(\omega)\, e^{2\pi i x \omega} d\omega \tag{2.3}$$

When using discrete instead of continuous functions, the integrals in the definitions become sums. Then the definition of the forward FT is given in equation 2.4 and in equation 2.5 for the inverse FT.

$$\tilde{f}(\omega) = \sum_{x=1}^{n} f(x)\, e^{-2\pi i x \omega} \tag{2.4}$$

$$f(x) = \sum_{x=1}^{n} \tilde{f}(\omega)\, e^{2\pi i x \omega} \tag{2.5}$$

2.2.2 Two dimensional FT

Since images are two dimensional objects the Fourier transform needs to be extended. The Fourier transform then becomes a complex function of two or more real frequency variables ω_1, ω_2. Since images are finite objects the discrete version of the two dimensional Fourier transform is given in equation 2.6 for the forward case and in equation 2.7 for the inverse case.

$$\tilde{f}(\omega_1, \omega_2) = \sum_{x=1}^{n} \sum_{y=1}^{m} f(x, y)\, e^{-2\pi i (\omega_1 \cdot x + \omega_2 \cdot y)} \tag{2.6}$$

$$f(x, y) = \sum_{x=1}^{n} \sum_{y=1}^{m} \tilde{f}(\omega_1, \omega_2)\, e^{2\pi i (\omega_1 \cdot x + \omega_2 \cdot y)} \tag{2.7}$$

2.3 Object detection

Object detection is a task within image processing where objects on a given image are supposed to be detected. These objects can be anything from buildings over cars

to humans. The images are already annotated for training, i.e. a rectangle (or other representation) that approximates the object best is already placed over the picture with the associated class attached. An example of such an annotated image can be found in figure 2.3.

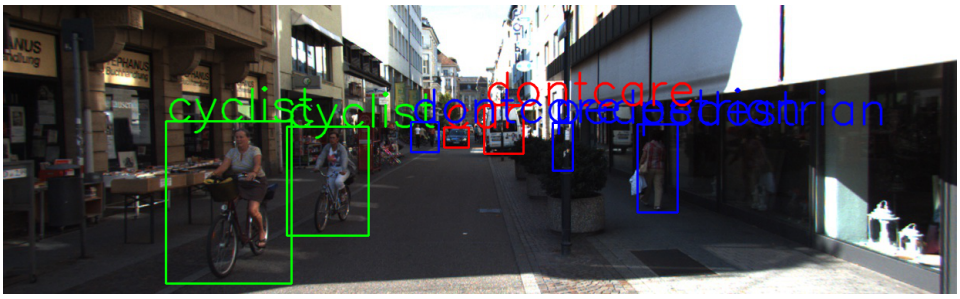


Figure 2.3: An example of a annotated image taken from the Kitti dataset. There are seven objects of four classes visible: cyclist, car, pedestrian and dontcare.

Many methods have been explored to increase the accuracy of object detection ranging from sliding window approaches over hand crafted feature extraction to artificial neural networks. All current state of the art results for object detection are achieved by using Convolutional Neural Networks (CNN) with minor or major adaptions (see [LAE+15], [Gir15], [RHGS15]). Therefore this work also uses a CNN as the backbone of the object detection.

2.4 Convolutional Neural Networks

For most image-related tasks, i.e. classification or object detection, a picture is used as a collection of pixels. However, not all pixels are equally important and subsets of the entire image form meaningfully connected subcollections. This might be a face in a photo of a family gathering. For humans the ability to detect these features and contextualize them comes naturally, for computers it does not. Therefore convolutional neural networks (CNNs) [LBD+89] are used. Convolutions are essentially just the application of filters on an image. The filter is applied at every possible location in the image, as described in figure 2.4. In formulas the application of a filter through convolution will be denoted with the convolution operator $\star$.

In CNNs there are multiple stages of filters in sequential order and multiple filters

[3]Figure taken from and animated version at `https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d`

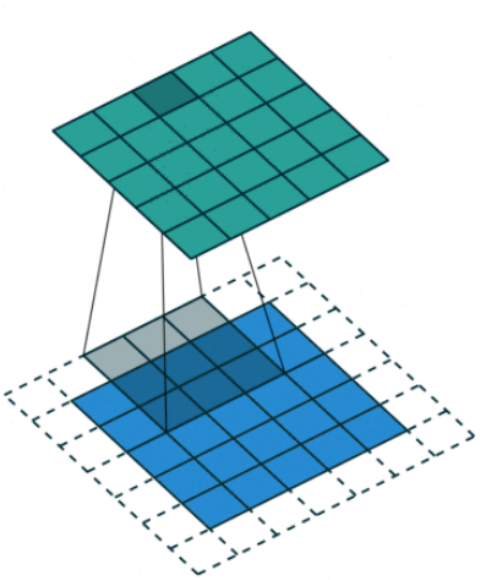


Figure 2.4: 3x3 convolution on 5x5 image. Resulting image is also 5x5 due to the padding of size 1 added to the original image. [3]

per layer. That means at every stage of the network different filters are applied on the outcome of an earlier step. The filters are assumed to learn different features of the images. The later the stage, the higher the level of complexity of the feature to be learned. That means, that an early filter might learn simple attributes such as edges or colors while a later filter might learn more complex features such as an eye or a nose and the very last filters even more complex semantic objects such as a face. The weights that are adapted during training determine what each individual filter does. This means that every filter that is a applied on a given layer learns its specific function such that overall the best accuracy can be achieved.

As just explained in conventional CNNs the filters are trained. However, there are some approaches that use static filters. The Scattering Transform is one of those approaches. Given its unique properties the results might be comparable to the conventional approach. If weights are not trained a lot of time can be saved.

2.5 Scattering Transform

A transformation from the time to the frequency domain cannot only be performed by using the sine but in principal with any given periodic function. Wavelets are

wave-like oscillation with an amplitude that begin and end at zero. In most use cases wavelets are specifically crafted to have certain properties. The Scattering Transform is based on a Morlet wavelet, which is defined in equation 2.8.

$$\psi(u) = C_1(e^{iu.\xi} - C_2)e^{\frac{-|u|^2}{2\sigma^2}} \tag{2.8}$$

where C_1 and C_2 are constants. C_2 is chosen such that $\int \psi(u)du = 0$, $u.\xi$ denotes the inner product of u and ξ, and $|u|^2$ is the norm in $\mathbb{R}^2$. Figure 2.5 shows the 2 dimensional Morlet wavelet with parameters $\sigma = 0.85$ and $\xi = \frac{3\pi}{4}$. These parameters are taken from [BM12]. No additional fine tuning w.r.t. to these parameters is done in this paper.

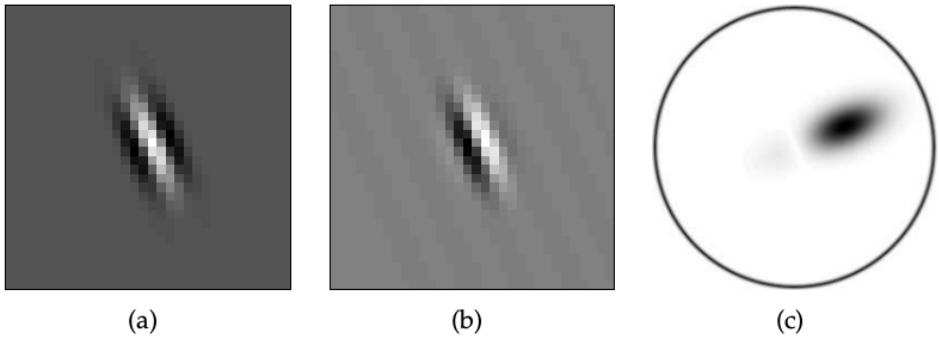


Figure 2.5: Complex morlet wavelet. a) Real part of ψ. b) Imaginary part of ψ. c) Fourier modulus $|\hat{\psi}|$. Image taken from [BM12].

2.5.1 Filter Bank

A wavelet transform filters x using a family of wavelets: $\{x \star \psi_\lambda(u)\}_\lambda$. It is computed with a filter bank of dilated and rotated wavelets having no orthogonality property. The filter bank has four parameters: M, N, J, L where M, N stand for the initial spatial size of the input, J is the scaling parameter and L is the number of angles used for the wavelet transform. J determines the size of the downsampling for the filters. The new output is downsampled by $2^{2 \cdot J}$, i.e. An input image of size $(32, 32)$ is downsampled by $J = 1$ to be of size $(16, 16)$ or by $J = 2$ to be of size $(8, 8)$. It is important to note that the filter bank is just an accumulation of filters and is independent of the data.
A visualization of the filter bank used in this work can be found in figure 2.6. The filters are shown for $J = 0, 1, 2$ and $L = 8$ different angles. The red blurry dot in the

bottom is the result of a Gabor filter which is a sinosoidal wave multiplied with a Gaussian function. The Gabor Filter is used as a low-pass filter.

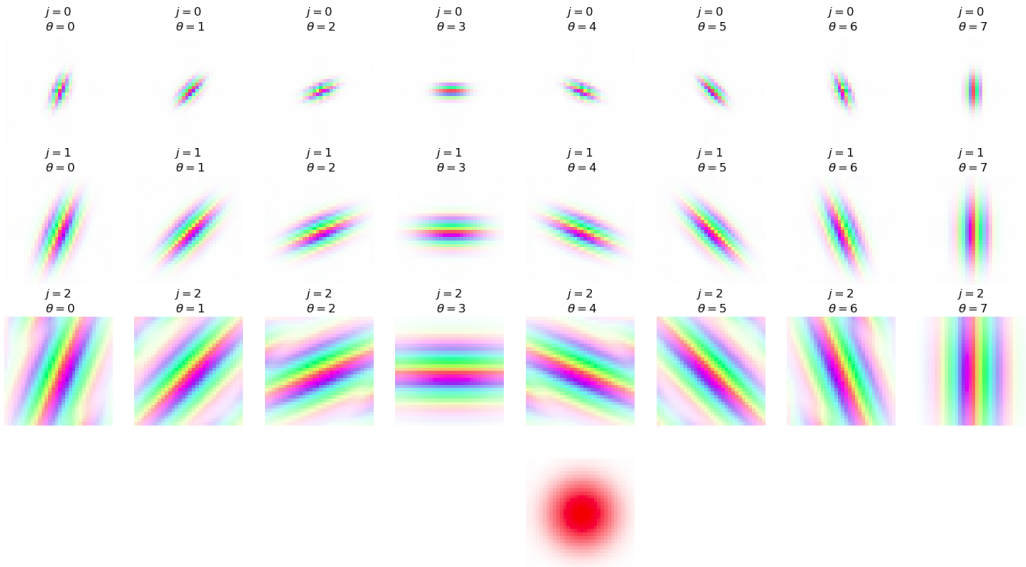


Figure 2.6: Visualization of the filter bank. The $j = 0,1,2$ describe the different downsample sizes. The $\theta = 0,...,7$ describe the different angles. For these images $N,M = 32, L = 8, J = 3$. The contrast corresponds to the amplitude and the color to the phase. The blurry red dot in the bottom is the corresponding low-pass filter.

2.5.2 Scattering Paths

To compute more and higher order scattering coefficients we iteratively apply the scattering transform. Let $U[\lambda]x = |x \star \psi_\lambda|$. A sequence $p = (\lambda_1, \lambda_2, ...\lambda_m)$ defines a *path*, which is the ordered product:

$$U[p]x = U[\lambda_m]...U[\lambda_2]U[\lambda_1] = |||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|...| \star \psi_{\lambda_m}|.$$

A scattering transform along the path p is defined as an integral, normalized by the response of a Dirac:

$$\bar{S}x(p) = \mu_p^{-1} \int U[p]x(u)du$$

with $\mu_p = \int U[p]\delta(u)du$. From this it follows that each scattering coefficient $\bar{S}x(p)$ is

invariant to a translation of x. The scattering is Lipschitz continuous to deformations as opposed to the Fourier transform modulus.

2.5.3 Scattering Networks

Scattering Networks are the result of all previously explained concepts. There are m layers in the network, where every m describes the length of the scattering paths in that layer. m is also called order because it describes the number of consecutive scattering applications in that path. A scattering network is a tree that starts with a single root node in layer $m = 0$ and branches out for every further layer with branching factor L. Similar to the filter bank, the scattering network is a collection of filters. However, the scattering network is the first time when data is actually input into those filters. In distinction to conventional CNNs the final output is not only taken from the last layer, but from every single node in that network. An example of a scattering network with $L = 4$ and $m = 2$ is shown in figure 2.7. The nodes describe the filters that are independent of the data, i.e. $U[\lambda_1]f$ for the first layer. The blue arrows indicate the outputs at every node, i.e. the scattering coefficients that result from applying this particular scattering path to data for example $S_J[\lambda_1]f$ for a node in the first layer. The root node $U_J[\theta]f = f \star \phi_J$ is the low-pass filter which is in this case a Gabor filter.
In [BM12] it is shown that using more than $m = 2$ produces a lot of unnecessary computation because most of the information from data is already captured in the second-order scattering coefficients. For practical purposes this paper from now on assumes that networks are at maximum $m = 2$ layers deep and in this paper for some applications $m = 1$ only. The total number of filters and therefore also the total number of outputs per datapoint are shown in 2.9 for $m = 1$ and in 2.10 for $m = 2$.

$$i \cdot (1 + JL) \tag{2.9}$$

$$i \cdot (1 + JL + \frac{1}{2}J(J-1)L^2) \tag{2.10}$$

i denotes the number of input channels of the input data which is 3 for most applications since RGB images are used. In the case of RGB images the scattering transformation is applied for every channel separately. Figure 2.7 is only showing a network for one abstract J. In a real network this J is an actual integer. Therefore it also has to be factored in the equations 2.9 and 2.10. Lastly, it should be noted that the

output of the scattering network all have the same downsampled size determined by J even if the filters have different sizes. This is achieved by subsampling the current scattering coefficients in the Fourier domain such that the output size is the desired one. To make this more clear an example is provided: A scattering network with $N,M = 32; J = 2; L = 8$ is initialized. The network is applied on an RGB image. Therefore there are $3 \cdot (1 + 2 \cdot 8) = 51$ outputs of size $(8,8)$ because of the downsampling factor $J = 2$.

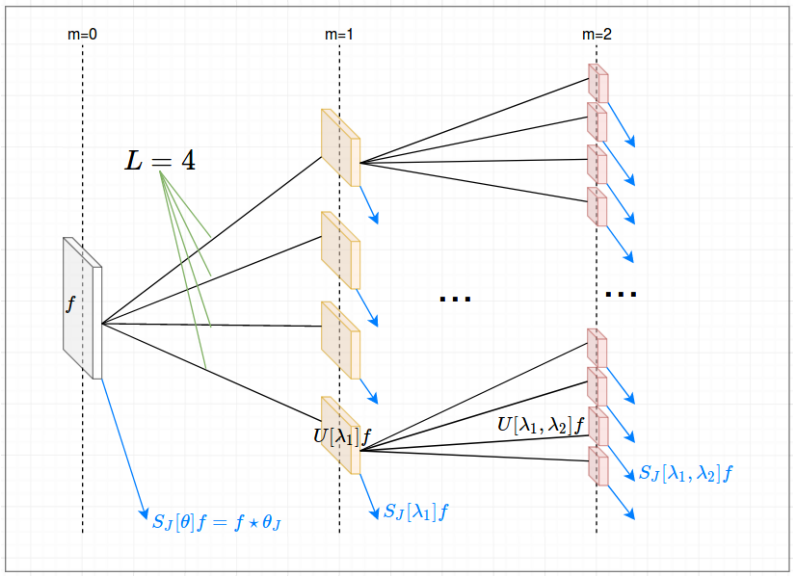


Figure 2.7: Representation of a scattering network. Each filter is described by its path, e.g. $U[\lambda_1]f$. Every blue arrow describes an output at the particular filter. The $m = 0,1,2$ describe the number of iterative applications of the scattering transform. $L = 4$ is the branching factor in the network also known as the number of angles for the filter.

The filters are applied in a convolutional manner, i.e. every point in the feature map is the result of the filter applied on a particular patch in the previous image or feature map. In comparison to the way convolution is described in section 2.4 the convolution is implemented through the Fourier Transform. By applying the inverse Fourier transform $\mathcal{F}^{-1}$, we can write:

$$f * g = \mathcal{F}^{-1}\Big\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\Big\}$$

Instead of sliding a filter over the image, both the filter and the image are transformed in Fourier space and multiplied point-wise. The result of the point-wise multiplication

is transformed back through the inverse Fourier Transform. The feature maps have semantic meaning and can be analyzed visually. For two different images the 0th order, first order and second order scattering coefficients have been visualized in figure 2.8 and 2.9. The first subfigure shows the original image. The second, third and fourth subfigures show the 0th to second order scattering coefficients respectively. The number of filters grows as described in 2.9 and 2.10. Through this example it is very clear that the filters primarily encode geometric information, i.e. edges and corners of the object in the image. To get an impression of the scattering coefficients in a more realistic setting, an additional image is analyzed in figure 2.9. As one can see in the cat image, the scattering coefficients are also interpretable in terms of geometric properties of the cat. Additional examples are shown in the appendix. A comparison of the images in the appendix and in this section suggests that the scattering transform is equivariant w.r.t transformation instead of invariant. This means that the translation of an object in the image corresponds to a similar or at least proportional translation in the feature map. This is a desirable property for object detection but stands in direct conflict to the characterization of invariance w.r.t translation taken from the original paper and described in 2.5.4.

2.5.4 Properties of the Scattering Transform

In this subsection the properties of the scattering transform as used in this work are layed out and the reasons for the properties are pointed out but not proven. For a more detailed explanation and references to proofs the original paper [BM12] must be consulted.
A wavelet is a localized waveform and therefore stable to deformations. This is an upgrade to the sinusoidal waves of the Fourier transform which do not have this property. A wavelet transform computes convolutions with wavelets. It is thus translation covariant not invariant. To achieve invariance a non-linearity must be added. One can then show that the only non-linearity that fulfills the $\mathbf{L}^2(\mathbb{R}^2)$ stability, is differentiable and commutes with translations is the $\mathbf{L}^1(\mathbb{R}^2)$ norm. Therefore it is chosen as the non-linearity.
As already stated in its respective subsection 2.5.2 a scattering transform along any scattering path is translation invariant. Compared to the Fourier transform modulus, which is also invariant to deformations, the scattering transform is Lipschitz continuous to deformations, i.e. the change in the scattering coefficients is bounded and determined by the change in the deformed object.

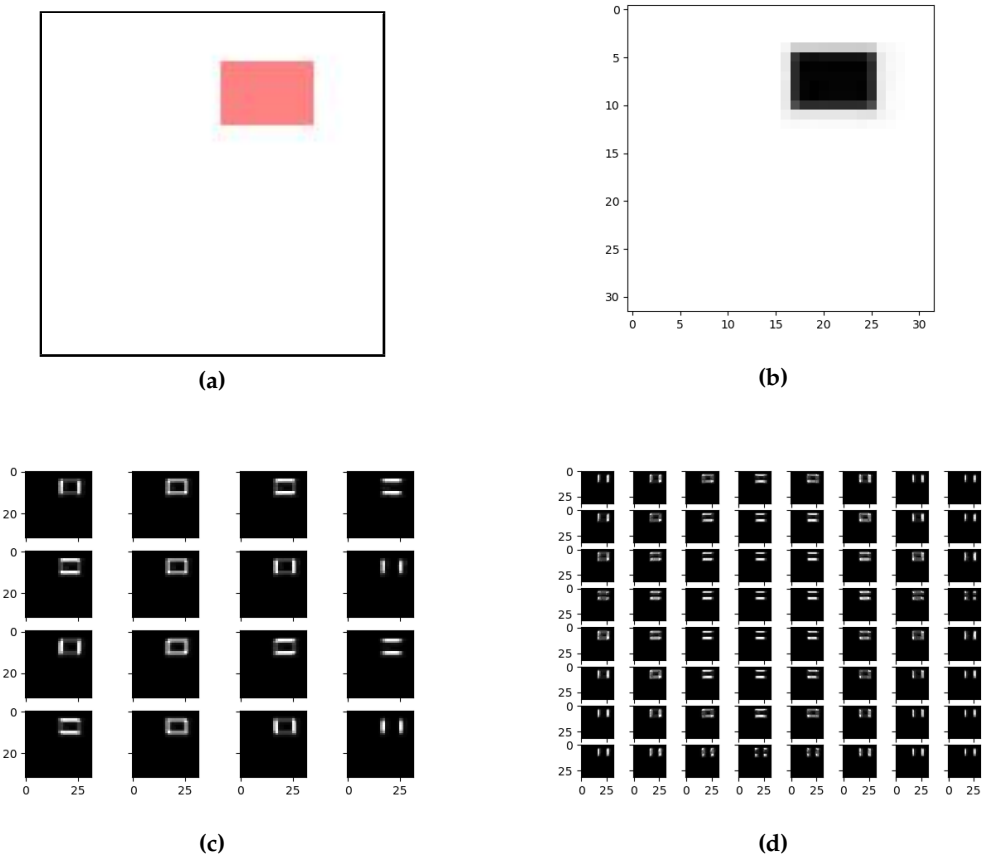


Figure 2.8: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

There are no theoretical bounds on the behavior of the scattering transform when confronted with scaled or rotated objects. However, there are some predictions one can make that will be layed out in the following. Given that scattering networks use rotated and dilated filters with L different angles with equidistant spacing it seems plausible that rotated objects should be captured in some of the filters. Scaled objects should also be captured by the scattering network, because scaling can be viewed as a specific subset of deformation. The Lipschitz continuity w.r.t. deformation should also benefit the detection of scaled objects.

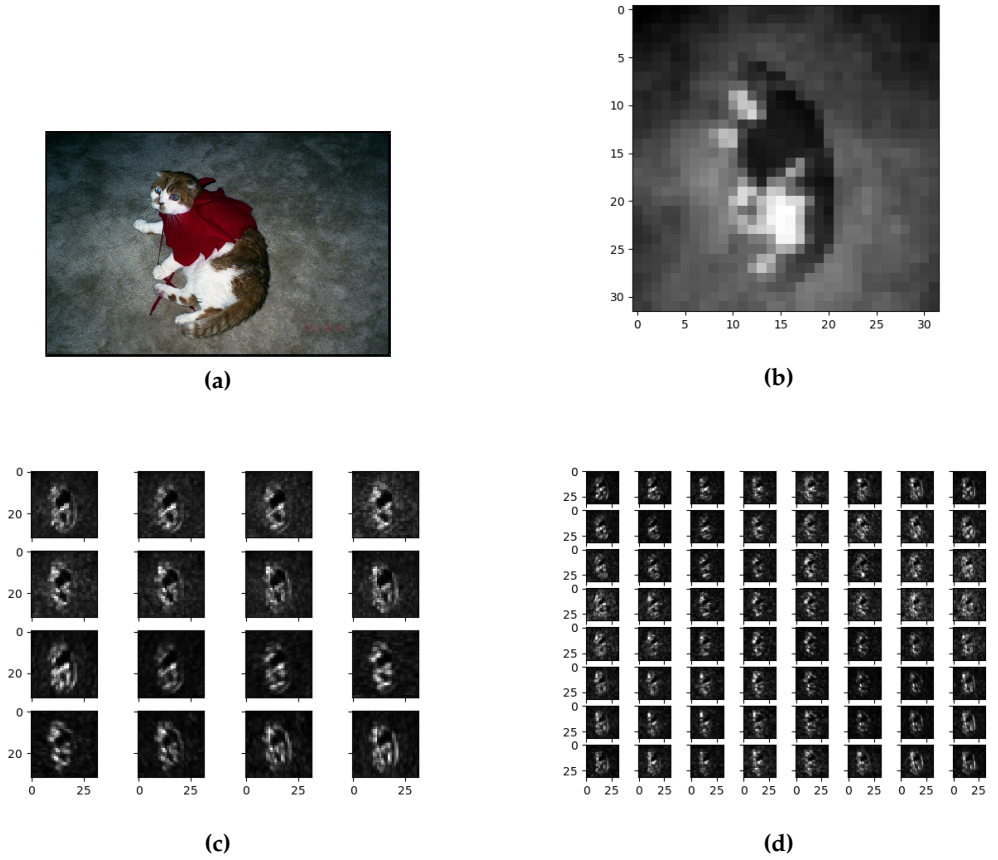


Figure 2.9: Image taken from the VOC dataset. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

2.5.5 Discussion of the properties

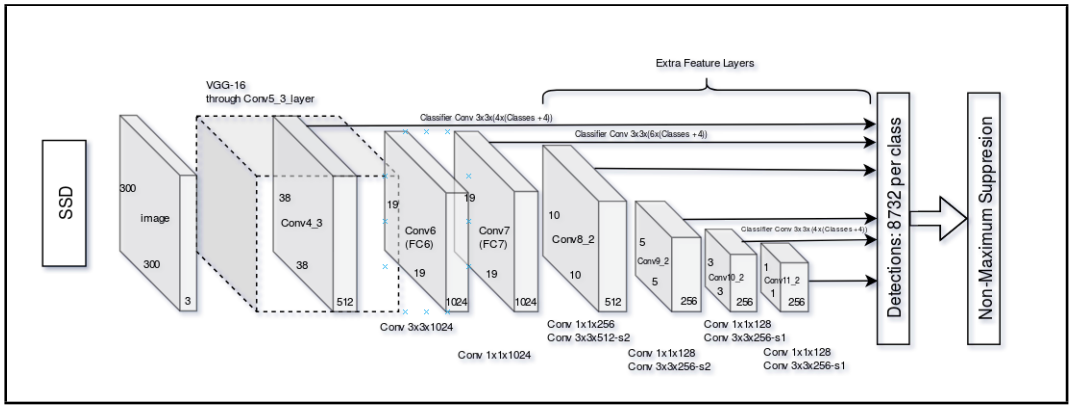
In this work the scattering transform is applied to object detection. In previous works the scattering transform has mainly been applied to image classification. In the following a short distinction between the two tasks is presented. For image classification invariance/Lipschitz continuity w.r.t. rotation, translation, scale and deformation are all positive attributes because every image corresponds to exactly one category. In the context of object detection invariance/Lipschitz continuity w.r.t. rotation, scale and deformation are positive attributes. However, the invariance w.r.t. translation is potentially very problematic. When two objects at two different locations in the image lead to exactly the same scattering coefficients it is impossible

to detect these objects correctly. However, the empirical results, i.e. figures 2.8, 2.9 and appendix suggest translation equivariance instead of invariance. Equivariance is a desirable property for object detection and therefore not problematic.
Given this theoretical insight this work proposes a specific kind of hybrid network which includes information from both a conventional CNN and the scattering networks.

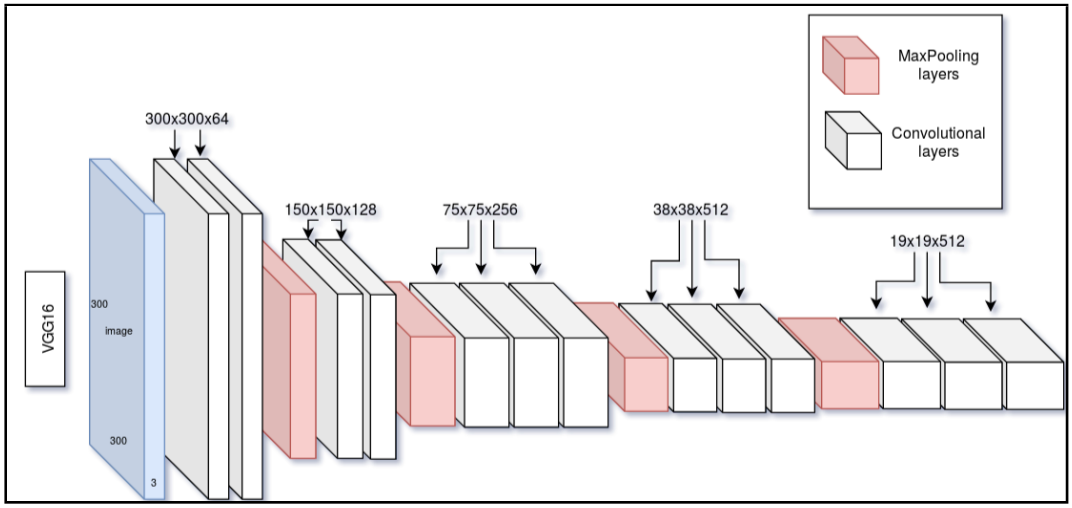
2.6 Single Shot Detector (SSD) - Object Detection Network

The Single Shot Detector (SSD) is a fully convolutional object detection network [LAE+15]. The SSD consists of two main components: an adapted version of a standard CNN used in classification tasks, i.e. VGG or ResNet, and extra feature layers that transform the features from the first part to meaningful information w.r.t. the final classes and their location. The final detections are then pushed through a Non-Maximum Suppression operation to put a higher weight on probably correct detections and a lower weight on probably incorrect detections. A detailed graphical description of the SSD and VGG16 can be found in figure 2.10. In the upper picture an overview of the SSD architecture can be seen. The data is piped through the adapted classification architecture, in this case VGG16. However, instead of connecting the output with a fully connected layer to the output layer it is now piped through the extra feature layers. Those are responsible for anchor boxes on the image with different feature sizes and aspect ratios. Some of the combinations of feature sizes and aspect ratios will match an object on the image. The probability that an anchor box represents an object is evaluated by the confidence layers. There are a total of 8732 detections per image and class . All detections that are bigger than a certain threshold are the output of the network. The VGG is just represented as a big box in the first image but a more detailed view is represented in the second. The main idea behind the VGG is to pipe the data through multiple convolutional layers that have decreasing feature map size but increasing number of channels. As already described in 2.4 the representations are increasingly large with increasing layer number. The reduction of the feature map size is done by a 2x2 max pooling operation which is a method to reduce the size of representations while keeping the most important information. For a theoretical introduction to max pooling or a general overview of CNNs see Schmidhubers overview paper [Sch15].

In its original paper [LAE+15] the SSD performed very well on the PASCAL VOC2007



(a)



(b)

Figure 2.10: Figure of the SSD network. a) Entire SSD with focus on extra feature layers and Non-Maximum Suppression. b) VGG16 in more detail. Explanations for the figures are in the respective section. The tensors NxNxC stand for the feature map size and number of channels, i.e. conv_2_1 has a feature map of 150x150 with 128 parallel channels. The red boxes describe $2x2$ max-pooling operations.

benchmark set, however, on most other object detection tasks, i.e. Kitti [GLSU13], it is outperformed by fast R-CNN [Gir15] or faster R-CNN [RHGS15] two other architectures for object detection. In this work SSD is used instead of other architectures because it is comparably simple and easier to set up. Given that the task of this work is only to test whether the scattering transform can be used effectively to perform object detection, relative measures to the standard architecture are sufficient.

However, there is no principal reason not to combine the scattering transform in the same way it is done in this paper with other object detection architectures if those are using convolutional layers (which is the current state of the art). This might be a suggestion for future work but is not part of this paper.

2.7 Hybrid Networks

Hybrid networks in the context of this paper describe a combination of conventional object detection network architectures and the scattering transform. Two possible ways to combine the two techniques are described in the following.

2.7.1 Sequential Hybrid Networks

Sequential hybrid networks describe an architecture in which the first couple of filters in a conventional object detection network have been replaced by the scattering transform. A general overview of the sequential architecture can be found in figure 2.11. The details of the implementation are found in figure 3.10 in the following chapter.

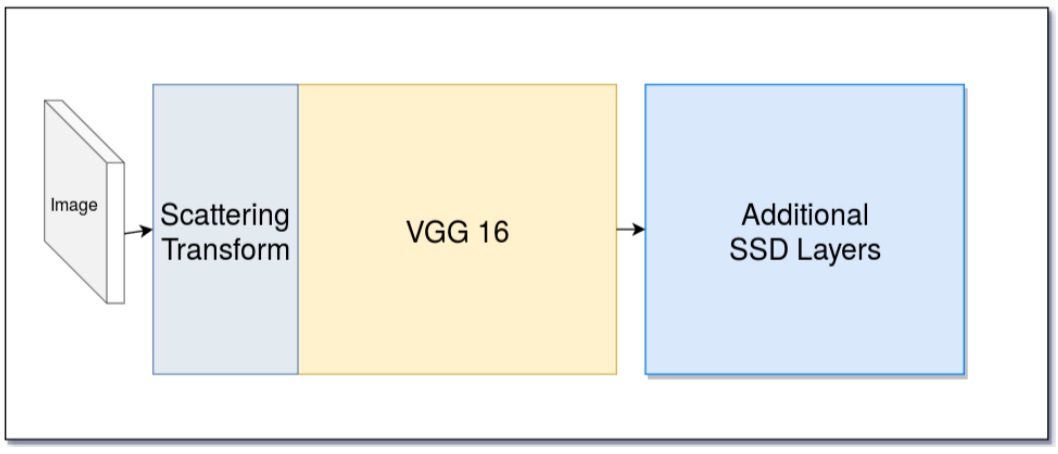


Figure 2.11: General overview of the sequential scattering architecture.

2.7.2 Parallel Hybrid Networks

Another possibility to combine the two networks is through concatenation during the forward pipeline. The intention here is to combine the strength of both techniques: the flexibility of the CNN with the robustness of the scattering transform. A general overview of the parallel architecture can be found in figure 2.12 The details of the implementation are found in figure 3.11 in the following chapter.

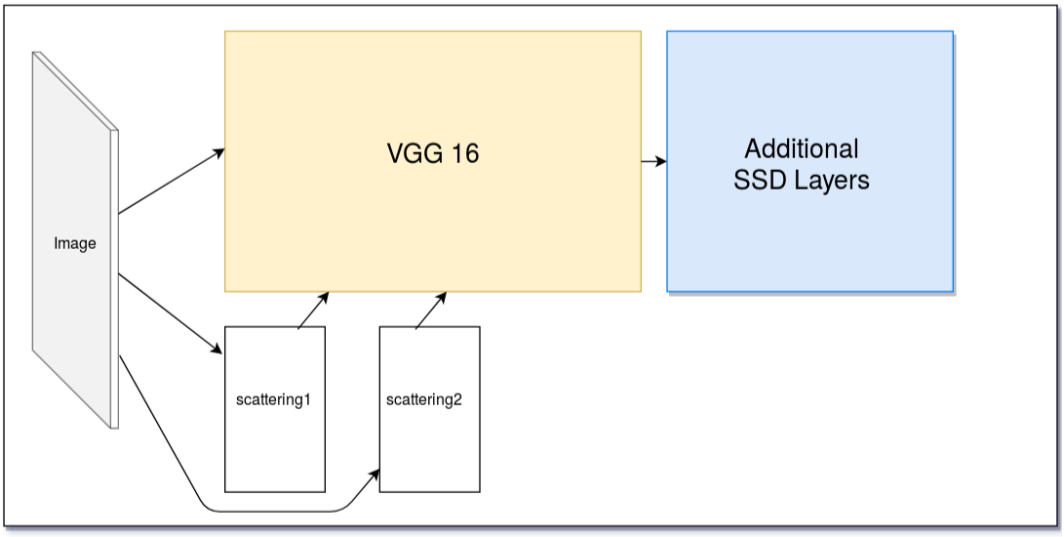


Figure 2.12: General overview of the parallel scattering architecture.

3 Experiments

Multiple experiments will be conducted in this paper. They test the performance of different scattering architectures, i.e. sequential and parallel, and compare them to the standard architecture on different datasets.

3.1 Datasets

The experiments in this work were conducted on multiple datasets. Two of them are known and openly available datasets for benchmarking purposes, namely VOC and KITTI. The other five datasets used were specifically created for this work to test the geometric properties of the scattering transform for object detection tasks.

3.1.1 PASCAL VOC

The PASCAL Visual Object Classes dataset is a common benchmark for object detection tasks [EEVG+15]. This work uses a combination of VOC images from 2007 and 2012 totaling 27088 images. They are split in test and training set such that the training set contains around 16k images. There are 20 different classes in the dataset that are frequently seen in a person's daily life ranging from aeroplane over bicycle to TV monitor. Most images are taken such that the object is centered and only one object is seen. In some rare instances two or more objects are seen in the same image, i.e. when a person is riding a horse or two cats are playing. To get a feeling for the dataset figure 3.1 shows three sample images from PASCAL VOC.

3.1.2 KITTI

The KITTI Vision dataset is specifically created for object detection of traffic scenes [GLSU13]. It has four classes: cars, cyclists, pedestrians and miscellaneous. The

(a) (b) (c)

Figure 3.1: Three samples from the PASCAL VOC dataset showing a dog, bus and TV monitor from left to right.

entire dataset has around 8k images and is split in around 6k training and 2k test images for this work. The scenes visible in the KITTI dataset contain multiple objects, i.e. multiple cars, cyclists and pedestrians in the same image and are therefore significantly harder to detect compared to PASCAL VOC. The original images are taken with aspect ratio 3:1. This might complicate the object detection process for the SSD which resizes the input to quadratic size. Four traffic scenes are depicted in 3.2.

(a) (b)

(c) (d)

Figure 3.2: Four samples from the KITTI dataset.

3.1.3 Toy Data

To test the quality of the networks on easier datasets with simple geometric properties a toy data set was constructed. Every image contains three objects that can also be overlayed. Every object has one of three randomly chosen colors. The objects are either a triangle, a rectangle or an ellipse. In total 6k such images are contained in the training set and 2k in the test set. Four samples can be found in 3.3

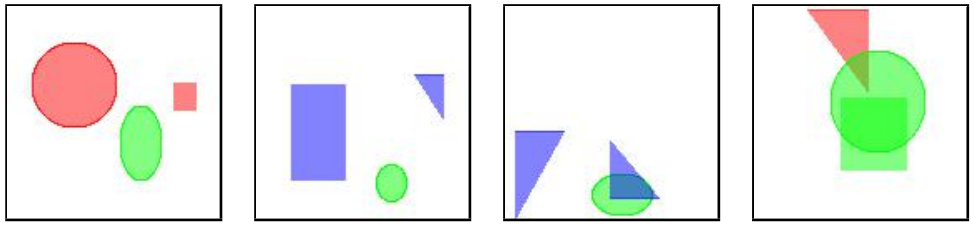


Figure 3.3: Four samples from the toy data set.

3.1.4 Test Invariances Toy Data

To test the possible invariances individually four additional datasets have been created. All of them contain only one object per image. They are all created in the same procedure: A base image with a randomly chosen object is created and put in the training set. Ten further images are derived from the base image w.r.t. the particular invariance of which 4 are put in the validation set and 6 in the test set. However, to test the invariances the validation set is not used such that the network is not training on manipulated data. This is done 1000 times per dataset resulting in 1k training and 6k test images per invariance. All objects have one of three randomly chosen colors such that the network does not learn to predict based on the color. The specific procedures are described in the following and samples are provided.

Scale

A base image is created as explained above. Ten further images are created by scaling the image with factors from 0.5 to 1.5 of the original size. The objects are triangles, rectangles and ellipses. Samples can be seen in 3.4.

Rotation

A base image is created as explained above. Ten further images are created by rotating the object around its own center with equal angle sizes. The objects are triangles, rectangles and heptagons. Samples can be seen in 3.5.

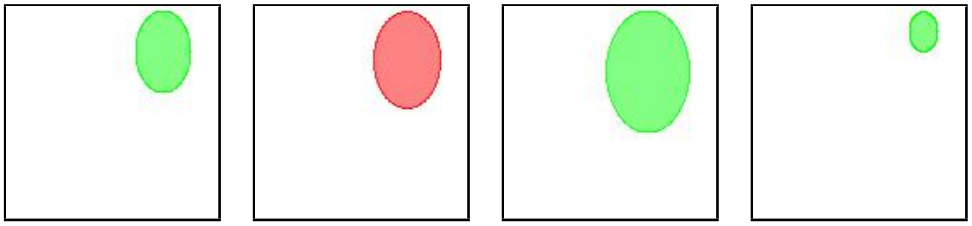


Figure 3.4: Four samples from the scale toy data set. a) is the base image; b) -d) are the scaled versions

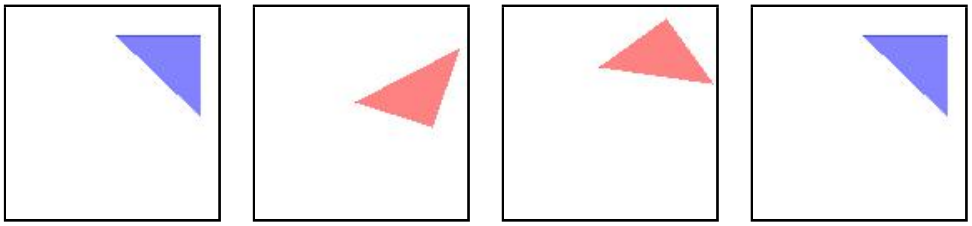


Figure 3.5: Four samples from the rotation toy data set. a) is the base image; b) -d) are the rotated versions

Deformation

A base image is created as explained above. Ten further images are created by taking one point of the object and adding noise to it. The noise is uniformly distributed and capped such that it can maximally add or subtract 20% of the original object size to the point. The objects are triangles, rectangles and heptagons. Samples can be seen in 3.6.

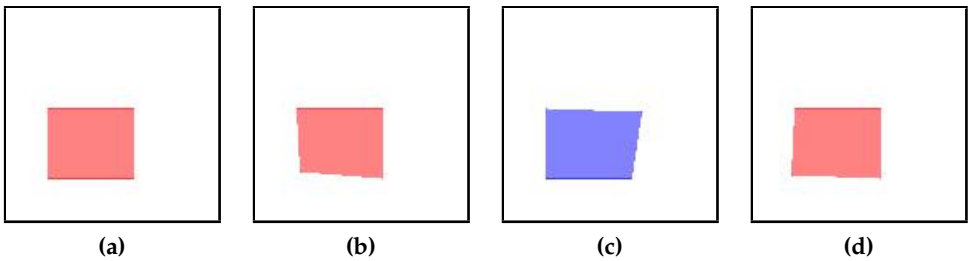


Figure 3.6: Four samples from the deformation toy data set. a) is the base image; b) -d) are the deformed versions

Translation

A base image is created as explained above. Ten further images are created by changing the location in the image randomly. The objects are triangles, rectangles and ellipses. Samples can be seen in 3.7.

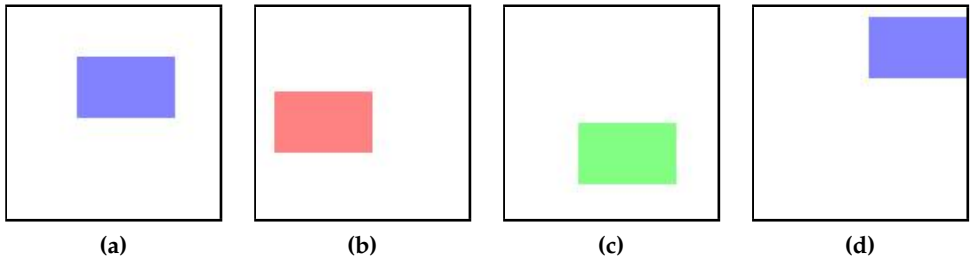


Figure 3.7: Four samples from the translation toy data set. a) is the base image; b) -d) are the translated versions

3.1.5 Classification Toy Data

To reproduce the results of [OBZ17] regarding classification a network is trained on a toy dataset created for classification. Every image contains exactly one triangle, ellipse or rectangle in different sizes and colors. Samples of the images are shown in figure 3.8.

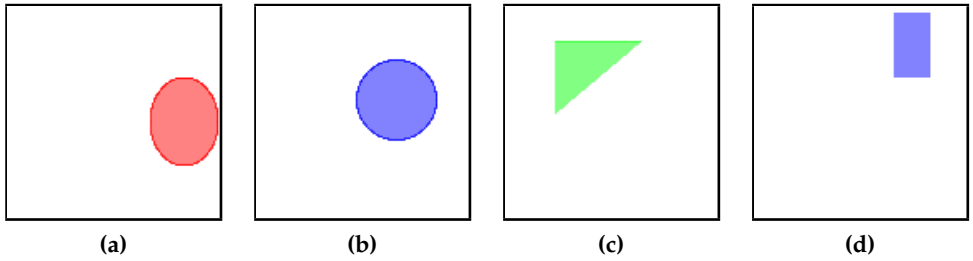


Figure 3.8: Four samples from the classification toy data set.

3.2 Setup

In the following section technical specificities are explained to further understand and potentially reproduce the findings of this work. The code for the experiments

has been implemented in Python3, and PyTorch [PGC+17] was used as a framework for the neural networks. The code can also be accessed at `https://github.com/mariushobbhahn/CompSci_Bachelor_201819`.

3.2.1 Technical Details

All experiments are run on the TCML-Cluster of the University of Tübingen. It has multiple nodes of 4 times GeForce GTX 1080 Ti GPUs. All algorithms use CUDA and as many computational steps as possible are parallelized.

3.2.2 Augmentations, Batchnorm, Multibox Loss, Optimizer

Since neural networks have become so popular many pre-processing steps and techniques applied during the forward pass have become successful and standard. In this subsection is a description of the different approaches that were chosen for the experiments. In all cases the images need to be reshaped to fit the SSD setup to have a size of 300x300 pixels with three color channels. A preprocessing step which will later be called Augmentations consists of photometric distortions, i.e. random contrast and saturation or changing the color representation of the image, random sample cropping and random mirroring of the data. The effectiveness of these augmentations is tested in the baseline experiments.
When the data has entered the network some other techniques can be applied during the forward pass. Using batch norm [IS15] is one popular tool. It consists of standardizing every batch to be distributed according to a Gaussian with mean 0 and standard deviation 1. This is done to speed up the training by preventing internal covariance shifts in the forward pass. The effectiveness of this technique is also tested in the baseline experiments.
To evaluate the quality of the proposals a metric is needed. The SSD works with the Multibox Loss. It is defined as

$$L(x,c,l,g) = \frac{1}{N}(L_{\text{conf}}(x,c) + \alpha L_{\text{loc}}(x,l,g)) \tag{3.1}$$

where $x_{ij}^p = \{1,0\}$ is an indicator for matching the i-th default box to the j-th ground truth box of category p, c are the class confidences, l are the predicted box and g the ground truth box parameters. Overall the multibox loss is a weighted sum of the localization loss L_{loc} which is a smooth L1-loss between the predicted and ground

truth box, and the confidence loss L_{conf} which is a softmax loss over multiple classes. The technical details can be found in the original paper [LAE+15]. The multibox loss combines the regression task of finding the correct bounding boxes and classification task of finding the correct objects for those boxes.
Lastly, the training procedure needs to be optimized. This is done by Stochastic Gradient Descent (SGD). Other optimizers such as Adam were tried but did not change the result in any way.

3.2.3 Hyperparameters

As any complicated neural network the SSD has many hyperparameters over which can be optimized. A full list can be found in table 3.1.

Table 3.1: Hyperparameters for the SSD. Explanations for some Parameters are given in the accompanying text.

Parameter	respective value(s)
Momentum	0.9
Weight decay	5e-4
Gamma (SGD)	0.1
learning rate (lr)	1e-3
lr batch norm	3e-3
lr steps (VOC)	(80000, 100000, 120000)
max iter (VOC)	125000
lr steps (Kitti)	(150000, 175000, 185000)
max iter (Kitti)	200000
lr steps (Toy Data)	(60000, 65000, 70000)
max iter (Toy Data)	75000
feature maps	[38, 19, 10, 5, 3, 1]
steps	[8, 16, 32, 64, 100, 300]
s_sizes	[30, 60, 111, 162, 213, 264, 315]
aspect ratios	[0.33, 0.5, 1, 2, 3]
multibox	[4, 6, 6, 6, 4, 4]

For some hyperparameters context and explanation is given in the following.
The first hyperparameters: momentum, weight decay gamma, learning rate, etc. are related to the optimizer itself. Gamma describes the amount by which the learning rate is adapted at every learning rate step. The learning rate is different when the batch norm is used as well, since the standardization allows for higher learning rates due to faster convergence. The learning rate steps describe at which epochs during

the training the learning rate should be reduced further. The datasets have different number of iterations due to their complexity. A network just needs more time to find patterns in street scenes than in triangles in front of a white background. The second part describes hyperparameters that are related to the object detection part of the network. The list called feature maps describes the sizes of the feature maps in the extra feature layers of the SSD network. They can also be seen in 2.10. The feature maps parameter is a list of feature map sizes used for the object detection. The bigger the size of the feature map the smaller the objects that can be detected are. A graphical explanation for the functionality of the feature map can be found in figure 3.9.

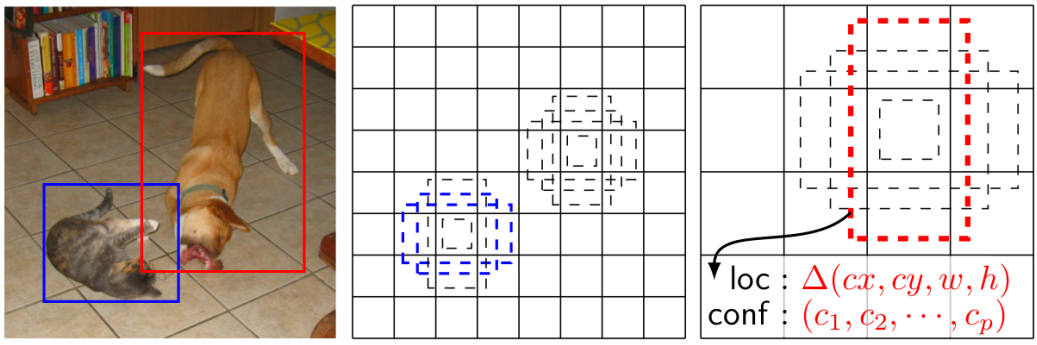


Figure 3.9: A graphical explanation of the feature maps. Image from the original SSD paper [LAE+15].

The steps parameter is used to determine the center points for the anchors on a given feature map cell. To make sure all cells have an anchor for each detection the following equation is approximately fulfilled for all extra layers: feature_maps[i] · steps [i] ≈ 300. The 300 is chosen because it is the size of an image in x- and y-direction. S_sizes is a scaling factor for the respective boxes at different stages of the detection forward pass. Aspect ratios describes the relative size in x- and y-direction of a given box, i.e. a box with aspect ratio 0.5 is twice as high as wide. Multibox describes the number of boxes with different aspect ratios per layer. A more detailed explanation can be found in the original SSD paper [LAE+15]. All values of those parameters are also taken from the original implementation. In the appendix in section 6.1 is a list of parameters and other methods that were tried during the making of this work but did not yield positive results such that future related research does not have to be a waste of valuable time.

3.3 Classification

Classification is a necessary precondition of object detection. To reproduce the results of [OBZ17] for the classification toy data set a small network is set up. The architecture is a two step procedure: first the data is channeled through a scattering transform with $N, M = 128, J = 2, m = 1$. In the second step the results of the scattering transform are piped through a ResNet [HZRS15]. Other architectures such as the VGG could also be used for the classification but the ResNet showed faster convergence for this experiment.

3.4 Baseline Performance of the SSD

In this experiment the Performance of the SSD is tested with different hyperparameters on different datasets. The hyperparameters are: Augmentations, Batchnorm and Pretrained where Pretrained describes a weight initialization of the VGG16 that has been pretrained on ImageNet [**?**]. The datasets are KITTI, PASCAL VOC and the Toy data set. During the experiments it was realized that the variance outcomes with similar inputs was greater than expected. Therefore every experiment was run multiple times and the mean and standard deviation is reported. There were two main goals of these experiments: a) set a baseline for the three main datasets to which the later experiments can be compared to. b) Test the influence of the hyperparameters to establish which parametrization of them was useful to reduce the number of networks to train for the later experiments. To test the effectiveness of different hyperparameters a generalized linear model (GLM) with binomial model family and logit link function was chosen. The model has all hyperparameters as independent variables and accuracy as the dependent variable. The effectiveness of a hyperparameter was evaluated along two metrics: a) whether the coefficient of the GLM describing that particular parameter was positive, i.e. whether the parameter had a positive influence on the accuracy and b) the p-Value of the parameter in the GLM. Even though there are many flaws with p-Values, especially with small numbers of repetitions such as in this case, a p-Value smaller than 0.05 was used to say that a variable has significant positive impact on the accuracy of the model. However the p-Value will merely be reported as an indication for the readers intuition. The parametrization of later experiments was decided according to the sign of the coefficient because of all the problems with p-Values.

3.5 Sequential Scattering

The VGG part of the network was adapted by removing the first two filters and max-pooling operations and replacing them by the scattering transform. The second max-pooling operation needed to be removed since the output of the scattering transform is of size 75x75 since $J = 2$ was chosen. The adapted VGG can be seen in figure 3.10. All other parts of the network, i.e. the feature layers stay equivalent to the basic SSD.
The experiments consisted of training a sequential scattering networks on all available datasets, i.e. KITTI, PASCAL VOC, Toy data, scale toy data, rotation toy data, deformation toy data and translation toy data with the hyperparametrization given by the baseline experiments. Additionally, *pretrained* was tested for all combinations. The results of *pretrained* might contain interesting information about the way in which the scattering networks process their data in the context of a different representation of the data.

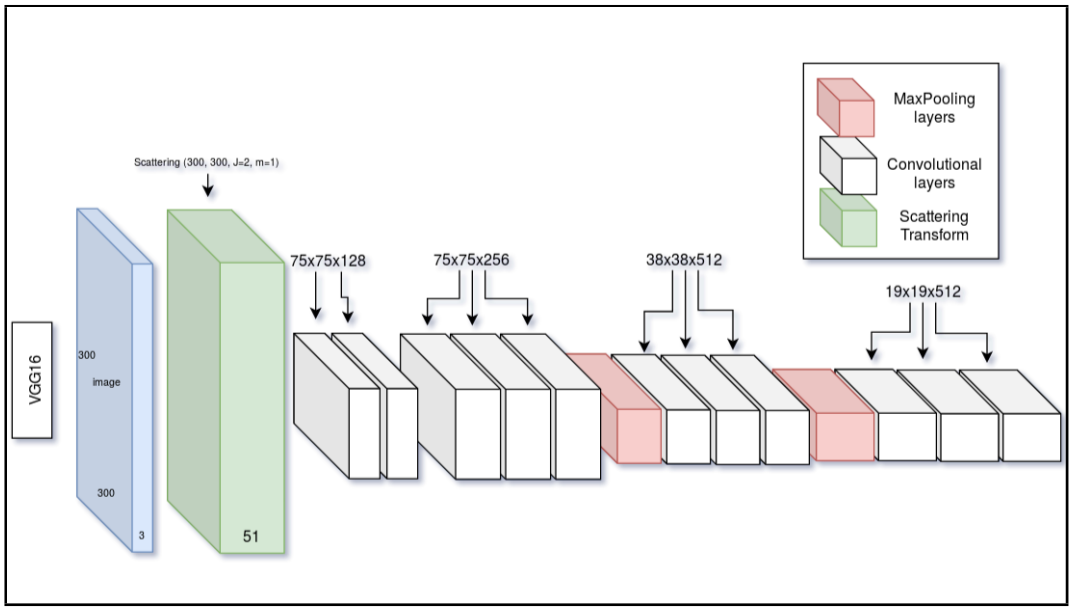


Figure 3.10: Sequantial scattering SSD. The first two filters are replaced by a scattering transform with $J = 2$ and $m = 1$. This means the output has $3 \cdot 17 = 51$ output channel.

3.6 Parallel Scattering

The parallel scattering network was realized by forwarding every image through the conventional pipe and two scattering networks. The first one had $J = m = 1$ and the second had $J = m = 2$. They were merged with the normal forward pass after the first and second max pooling operation respectively. The number of channels was calculated as described in 2.9 and 2.10 and results in 27 and 243 channels for the first and second scattering transformation respectively. A graphical interpretation of the parallel hybrid network can be seen in figure 3.11.
The experiments consisted of training a sequential scattering networks on all available datasets, i.e. Kitti, PASCAL VOC, Toy data, scale toy data, rotation toy data, deformation toy data and translation toy data with the hyperparametrization given by the baseline experiments. Additionally, *pretrained* was tested for all combinations. The results of *pretrained* might contain interesting information about the way in which the scattering networks process their data when confronted with another representation of it. The *batchnorm* makes all results of the parallel scattering very bad and is therefore turned off.

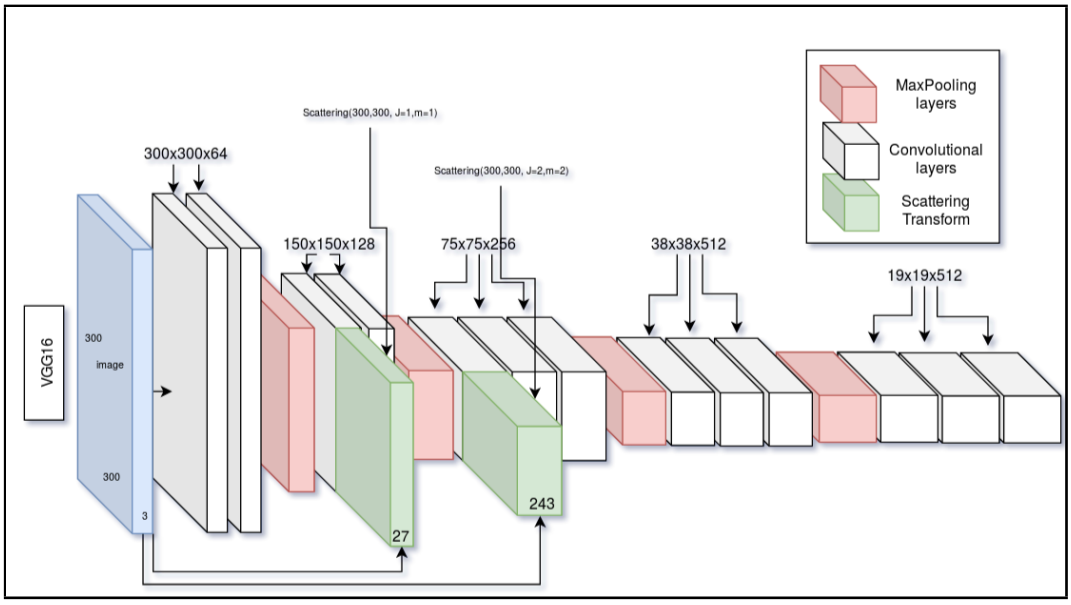


Figure 3.11: Parallel scattering SSD. The image is piped through the normal VGG path and additionally through a scattering transform with $J = 1$ and $m = 1$ and a second one with $J = 2$ and $m = 2$. The outputs of the scattering transforms are concatenated with the other pipe as soon as the dimensions are compatible.

3.7 Small Toy Data Experiments

One of the possible advantages of filters that need no new training is that their models converge faster and that they generalize better from small sample sizes. Therefore an experiment was run for the toy dataset with only 600 samples for training and 200 for testing. The networks only train 25k epochs instead of 100k. The experiment was conducted for the standard SSD, the sequential scattering and the parallel scattering network respectively. Each network was trained multiple times to prevent outliers from determining the results too much. To test the convergence behavior on small datasets even further a second experiment with only 5k epochs was run for all three network types. To compare the results in a more realistic setting all three network types are also trained on the PASCAL VOC dataset with 25k and 5k epochs respectively.

3.8 Timeconstraints

Especially in object detection online usage of the network is an important goal. In the context of self driving cars, for example, a network that is not able to process data within a given period of time makes the network completely useless since the car is unable to drive efficiently. The experiment will measure the average time of $n = 100$ forward passes for the SSD, sequential and parallel scattering on a given dataset. For this experiment every timing run will be on one GPU only since this is the most likely use case for applications.

4 Results

In this chapter the results of all experiments are presented. Every result is put into perspective and discussed making an extra discussion chapter unnecessary.

4.1 Classification

The Network in combination with the scattering transform is able to classify all test data correctly with 100 percent accuracy after two episodes of training before it starts overfitting to the training data. In comparison, the network without the scattering transform achieves 99.65 percent accuracy after four episodes before it starts overfitting. This shows that the scattering transform is able to provide useful information, at least when presented with simple datasets as shown in figure 3.8. Further implications like a slightly better accuracy or a slightly faster convergence are potentially true, but cannot be concluded with high confidence from this single observations. This experiment gets its relevance for this paper by showing classification, a necessary condition for object detection, is fulfilled.

4.2 Baseline Performance of the SSD

In this section the results of all the main experiments are presented. First the baselines that establish reasonable values for the use of batch norm and augmentations. Additionally, the baselines for comparison with the hybrid networks are determined. In the second subsection the results of the sequential scattering experiments are presented while the third subsection shows those of the parallel approach.

4.2.1 Hyperparameterization and Baselines for PASCAL VOC, KITTI and Toy data

The average and standard deviation (std) of a given combination of variables can be found in table 4.1. The coefficient of the GLM for augmentations are positive and it is therefore used as a standard for future experiments. The p-value is $0.044 < 0.05$ and we can therefore say it has a significant positive effect on the training. The coefficient for batch norm is also positive and is used in later experiments. Its p-value, however, is $0.79 > 0.05$ and therefore we cannot assume that it has a significant positive effect. Pretraining on ImageNet has a positive coefficient as well but no significant p-value with 0.41. Future experiments will still consider both possible values of *pretrained* since interesting information about the network structure might be analyzed. On average the combination of Augmentations, no batchnorm and Pretrained has the highest mean accuracy for all datasets. For PASCAL VOC an accuracy of 0.630 is achieved, for Kitti it is 0.125 and for the Toy data it is 0.792. These are the baselines to be compared with the results of the scattering experiments. The standard deviations show that the networks converge to somewhat similar functions even if their results deviate by some noise. Given that outliers were removed prior to the GLM fitting the standard deviations are only minor.

A screenshot of all results of the GLM can be found in the appendix in figure 6.4.

4.2.2 Invariant toy data

The results of the invariant toy data experiments with the standard architecture with augmentations and batchnorm set to true can be found in table 4.2. All experiments have on average a slightly higher accuracy for the model without pretraining. This is confirmed by the negative coefficient of the GLM that can be found in the appendix in figure 6.5. All other results of the GLM are in the same figure. This seems plausible given that the invariances of geometric object have nothing to do with the patterns learned on real life objects from ImageNet. The networks are able to recognize objects are deformation with accuracy of 0.928. However the deformations are not that big which might be the primary reason for that result. Rotation and scale have and accuracy of 0.635 and 0.644 respectively. Translation has an accuracy of 0.002. This could be explained either by overfitting on the training set or a impossibility to generalize from the small number of training data. Overall it shows that the

Table 4.1: Results of the baseline experiments. Mean and standard deviation are denoted for every combination of features that were measured

Dataset	Augmentations	Batchnorm	Pretrained	Mean	Std_dev
VOC	0	0	0	0.108	0.008
VOC	0	0	1	0.363	0.055
VOC	0	1	0	0.329	0.041
VOC	0	1	1	0.341	0.017
VOC	1	0	0	0.364	0.025
VOC	1	0	1	0.630	0.003
VOC	1	1	0	0.568	0.002
VOC	1	1	1	0.619	0.007
Kitti	0	0	0	0.027	0.024
Kitti	0	0	1	0.032	0.009
Kitti	0	1	0	0.032	0.010
Kitti	0	1	1	0.024	0.002
Kitti	1	0	0	0.050	0.010
Kitti	1	0	1	0.125	0.011
Kitti	1	1	0	0.116	0.012
Kitti	1	1	1	0.113	0.010
Toy_data	0	0	0	0.487	0.060
Toy_data	0	0	1	0.505	0.027
Toy_data	0	1	0	0.511	0.123
Toy_data	0	1	1	0.474	0.053
Toy_data	1	0	0	0.773	0.017
Toy_data	1	0	1	0.792	0.049
Toy_data	1	1	0	0.613	0.128
Toy_data	1	1	1	0.710	0.058

possibility to generalize is rather limited for a standard SSD network trained on a low number of training data points.

Table 4.2: Results of the invariant toy data experiments. Mean and standard deviation are denoted for every combination of features that were measured

Dataset	Pretrained	Mean	Std_dev
Deformation_data	0	0.928	0.003
Deformation_data	1	0.896	0.026
Rotation_data	0	0.635	0.010
Rotation_data	1	0.622	0.026
Translation_data	0	0.001	0.001
Translation_data	1	0.002	0.001
Scale_data	0	0.644	0.006
Scale_data	1	0.637	0.004

4.3 Sequential Scattering

4.4 Parallel Scattering

4.5 Small Toy Data Experiments

The results of the small data experiments can be found in table 4.3. On the small toy dataset the sequential scattering outperforms both the standard and the parallel scattering network significantly with 0.759 to 0.630 and 0.411 for 25k epochs and 0.121 to 0.043 and 0.003 for 5k epochs. In the case of the PASCAL VOC dataset the standard SSD outperforms both others with 0.317 to 0.053 and 0.013 for 25k epochs and 0.025 to 0.011 and 0.004 for 5k epochs. The conclusions from this are twofold: a) the

4.6 Timing Evaluation

The results of the timing evaluation can be found in table 4.4. The sequential scattering SSD setup is the fastest with an average of 0.178 seconds per forward pass followed by the normal SSD setup with 0.236 seconds. Both are far ahead of the parallel scattering setup with 1.499 seconds per forward pass. The standard deviations are very small in all cases as a result of the deterministic nature of all three methods. The timing is not compared to other network setups, i.e. a ResNet instead of a VGG, since the scattering approach is easily transferable to other networks and

Table 4.3: Results of the small data experiments. Mean and standard deviation of the accuracy are denoted for every combination of features that were measured.

Dataset	epochs	network type	Mean	Std_dev
Toy_data_small	25k	standard	0.630	0.008
Toy_data_small	25k	sequential_scattering	0.759	0.004
Toy_data_small	25k	parallel_scattering	0.411	0.012
Toy_data_small	5k	standard	0.043	0.007
Toy_data_small	5k	sequential_scattering	0.121	0.027
Toy_data_small	5k	parallel_scattering	0.003	0.001
VOC	25k	standard	0.317	0.011
VOC	25k	sequential_scattering	0.053	0.006
VOC	25k	parallel_scattering	0.013	0.001
VOC	5k	standard	0.025	0.001
VOC	5k	sequential_scattering	0.011	0.007
VOC	5k	parallel_scattering	0.004	0.000

relative results are therefore the only relevant ones.

There are two conclusions to be drawn from these results. First, the sequential scattering is slightly faster than the normal SSD and therefore could replace it in specific niche tasks when both have the same accuracy. Second, the parallel scattering approach is slower by a factor of 6-9 and is therefore only justified either when time is no constrain (e.g. offline applications) or when the accuracy of the parallel approach far outperforms the other two. The reason for the parallel approach taking so much longer than the other two is the calculation of second order coefficients. If this is left out. This approach should only be marginally slower.

Table 4.4: Mean and standard deviation (std.) of 100 runs of the timing evaluation for the normal SSD, the sequential and the parallel scattering SSD are shown. Means are reported in seconds.

network type	mean	std.
normal SSD	0.236	0.004
sequential scattering	0.178	0.004
parallel scattering	1.499	0.002

5 Conclusion

This chapter gives a very small summary and an overall conclusion of the techniques and experiments in this paper. Additionally it features an outlook for future work related to or building up on this paper.

5.1 Future Outlook

Given the just presented conclusion there are two main avenues for future work. First, additional experiments which support the strength of the scattering transform should be conducted. These could entail training on very small dataset from real applications instead of just creating toy data such as in this paper or finding other use cases that fulfill the specific advantages of the scattering transform, i.e. low noise environments and clear geometric properties of the shapes. The second avenue is an investigation of other ways to set up the parallel scattering network. The setup chosen in this paper is too time consuming and does not achieve a justifiable accuracy compared to the other presented methods. A first step could be to not use the second order scattering coefficients but only first order coefficients instead.

6 Appendix

Additional material for explanations and results can be found here. There is also a section of parameters and methods that were tried but did not yield positive results. This is done such that other scientist that want to research this or related problems do not need to waste time.

6.1 Negative Results

1.

6.2 Explanations of the Scattering Transform

Figure 6.1 shows a translated version of the image found in section 2.5.3.
Figure 6.2 shows the scattering coefficients of another object, in this case an ellipse, to see the properties of rounded edges.
Figure 6.3 shows a scaled version of 6.2.

6.3 Results

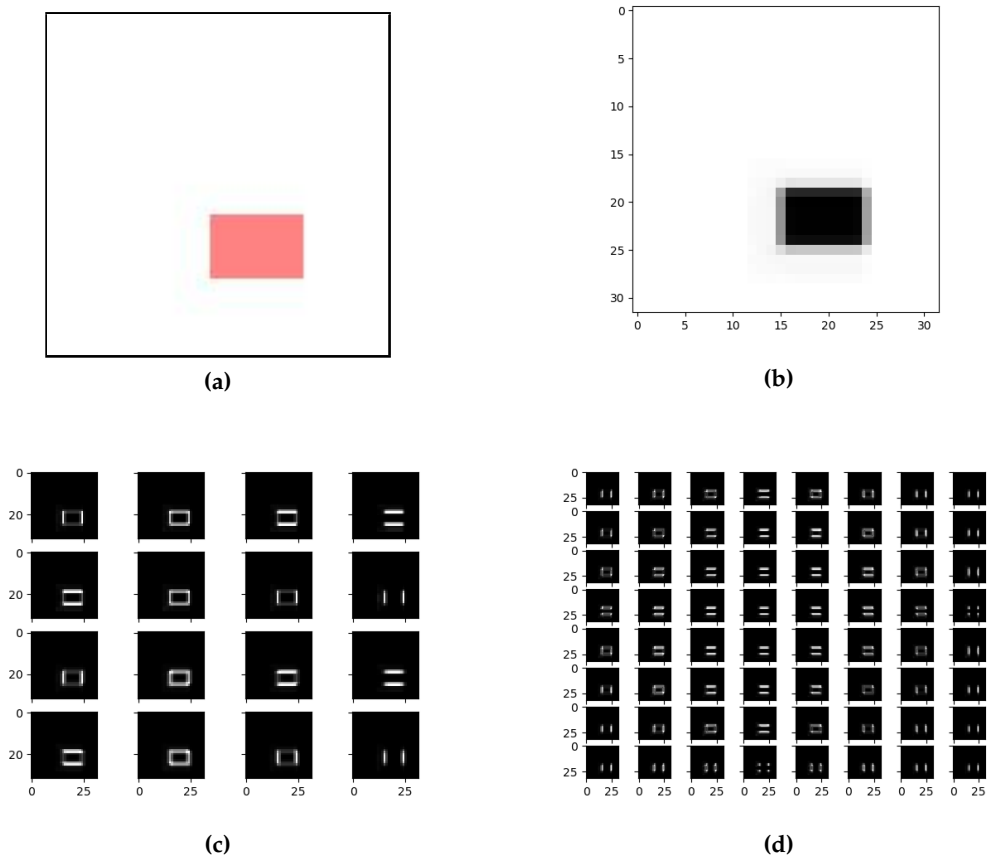


Figure 6.1: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

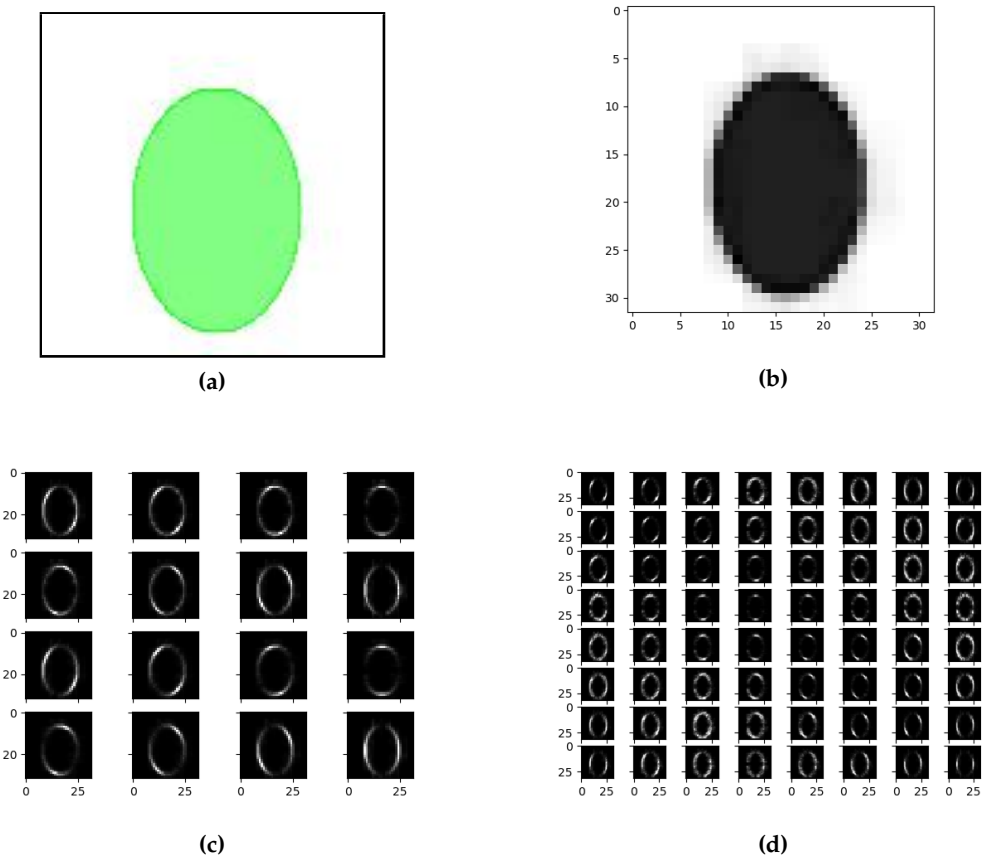


Figure 6.2: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

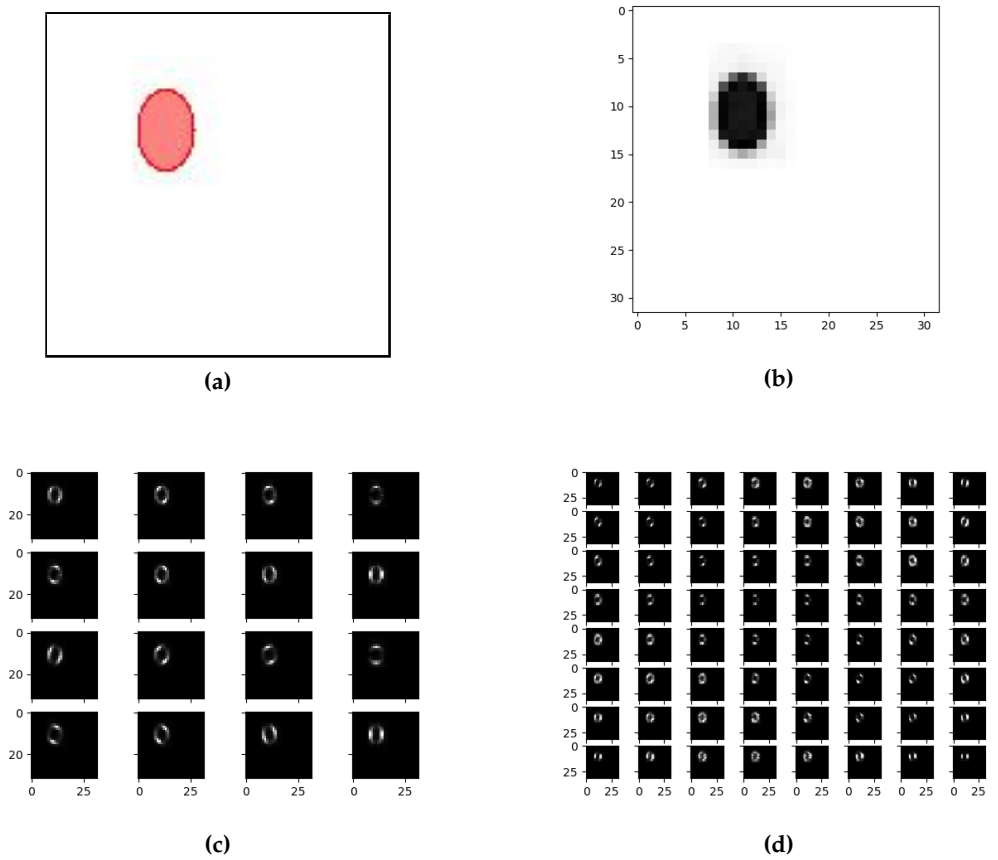


Figure 6.3: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

Dep. Variable:	Accuracy	**No. Observations:**	88
Model:	GLM	**Df Residuals:**	82
Model Family:	Binomial	**Df Model:**	5
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-29.873
Date:	Wed, 12 Jun 2019	**Deviance:**	2.4687
Time:	16:47:27	**Pearson chi2:**	2.45

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-1.2916	0.447	-2.892	0.004	-2.167	-0.416
VOC	0.1038	0.369	0.282	0.778	-0.619	0.826
Toy_data	0.9542	0.394	2.420	0.016	0.182	1.727
Kitti	-2.3497	0.617	-3.811	0.000	-3.558	-1.141
Augmentations	1.0703	0.530	2.018	0.044	0.031	2.110
Batchnorm	0.1368	0.525	0.261	0.794	-0.892	1.166
Pretrained	0.4302	0.527	0.817	0.414	-0.602	1.463

Figure 6.4: GLM for the baseline experiments. Positive Coefficients imply a better accuracy. P-values below 0.05 are significant.

Dep. Variable:	Accuracy	**No. Observations:**	42
Model:	GLM	**Df Residuals:**	37
Model Family:	Binomial	**Df Model:**	4
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-11.790
Date:	Wed, 12 Jun 2019	**Deviance:**	0.086250
Time:	16:46:44	**Pearson chi2:**	0.0930

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-0.5830	1.724	-0.338	0.735	-3.962	2.796
Deformation_data	2.9682	1.903	1.560	0.119	-0.761	6.697
Rotation_data	1.1550	1.756	0.658	0.511	-2.287	4.597
Scale_data	1.2074	1.770	0.682	0.495	-2.261	4.676
Translation_data	-5.9137	6.678	-0.886	0.376	-19.002	7.175
Pretrained	-0.0920	0.822	-0.112	0.911	-1.704	1.520

Figure 6.5: GLM for the baseline invariant experiments. Positive Coefficients imply a better accuracy. P-values below 0.05 are significant.

Dep. Variable:	Accuracy	**No. Observations:**	36
Model:	GLM	**Df Residuals:**	31
Model Family:	Binomial	**Df Model:**	4
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-8.0310
Date:	Wed, 12 Jun 2019	**Deviance:**	1.3209
Time:	16:43:30	**Pearson chi2:**	1.27

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-1.0483	0.356	-2.947	0.003	-1.745	-0.351
Toy_data_small	0.6860	0.549	1.250	0.211	-0.390	1.762
VOC	-1.7343	0.706	-2.457	0.014	-3.118	-0.351
twentyfivek	1.1156	0.637	1.751	0.080	-0.133	2.364
fivek	-2.1638	0.881	-2.457	0.014	-3.890	-0.437
standard	0.2188	0.735	0.298	0.766	-1.222	1.660
sequential_scattering	0.0583	0.740	0.079	0.937	-1.392	1.509
parallel_scattering	-1.3253	0.887	-1.494	0.135	-3.064	0.413

Figure 6.6: GLM for the small data experiments. Positive Coefficients imply a better accuracy. P-values below 0.05 are significant.

Bibliography

[ACC+17] Tameem Adel, Taco Cohen, Matthan Caan, Max Welling, On behalf of the AGEhIV study group Initiative, and the Alzheimer's Disease Neuroimaging. 3d scattering transforms for disease classification in neuroimaging. *NeuroImage: Clinical*, 14:506–517, 2017. Exported from https://app.dimensions.ai on 2018/10/21.

[BM12] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *CoRR*, abs/1203.1513, 2012.

[EEVG+15] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[Gir15] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[LAE+15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[LBD+89] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Back-

propagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[OBZ17] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. *CoRR*, abs/1703.08961, 2017.

[OM14] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. *CoRR*, abs/1412.8659, 2014.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[RHGS15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[Sch15] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[SM13] Laurent Sifre and Stephane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1233–1240, Washington, DC, USA, 2013. IEEE Computer Society.