



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: 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 16, 2019

Abstract

The Scattering Transform is a filter that can be applied on images with local invariance but global equivariance w.r.t. translation, scale, rotation and deformation of an object. It has already been shown that it is beneficial to use the scattering transform in combination with Deep Learning architectures in the case of classification. Specifically if there are low number of training data available or the data has specific geometric properties the Scattering Transform is very useful. This paper extends the use of the Scattering Transform from classification to object detection by combining it with classical object detection architectures such as an SSD in two ways. The first is analogous to the classification case, namely combining the architectures sequentially. The data is first transformed by the Scattering Transform and the Deep Learning architecture learns based on the resulting scattering coefficients. The second is a technique that is new to this paper. The Scattering Transform and the Deep Learning architecture are combined in parallel, that is the data is fed through both methods and then continuously merged at later stages of the network. The respective methods are tested and compared to the conventional approach on different datasets. Some of the datasets are created specifically to test the attributes that the theoretical guarantees of the Scattering Transform would predict, i.e. transformations such as scaling, deforming, rotating or translating an object. Lastly, it is tested on small datasets with low amounts of training and the timing of a forward pass is compared to other approaches. The experiments conclude that the Scattering Transform if combined sequentially is able to achieve similar results to the standard approach in most cases, underperforms in noisy environments such as real life images but overperforms when small amounts of data is available and the objects have specific geometric properties. This makes it a useful tool for specific applications.

Zusammenfassung

Die Scattering Transformation ist ein Filter, der auf Bilder angewendet werden kann. Sie hat die Eigenschaft lokal invariant aber global equivariant im Bezug auf Translation, Skalierung, Rotation und Deformation zu sein. Es wurde in früheren Arbeiten bereits gezeigt, dass diese Eigenschaft vorteilhaft für die Klassifizierung von Objekten auf Bildern ist wenn man die Scattering Transformation mit anderen Deep Learning Architekturen kombiniert. Besonders wenn wenig Datenpunkte oder bestimmte geometrische Eigenschaften der Objekte vorhanden sind, ist die Scattering Transformation erfolgreich. Diese Arbeit erweitert die Scattering Transformation vom Bereich der Klassifizierung auf die Objekterkennung in zwei Arten. Zunächst, analog zur Klassifizierung, werden die Scattering Transformation und die Deep Learning Architektur sequentiell kombiniert. Das heißt, die Daten werden erst von der Scattering Transformation verändert und deren Ergebnis dann in die konventionelle Architektur gefüttert. Die zweite Methode ist neu und wird in dieser Arbeit eingeführt. Dabei werden die beiden Architekturen parallel kombiniert. Die Daten werden sowohl durch die Scattering Transformation als auch durch die herkömmliche Deep Learning Architektur verarbeitet und dann kontinuierlich verschmolzen. Die Methoden werden auf verschiedenen Datensätzen getestet und mit den konventionellen Ansätzen verglichen. Dabei sind manche Datensätze spezifisch für das testen von den theoretisch vorhergesagten Eigenschaften der Scattering Transformation erschaffen worden. Zusätzlich werden auch sehr kleine Datensätze mit wenig Trainingszeit getestet und die Zeit die der jeweilige Ansatz für das verarbeiten eines Bilder benötigt gemessen und verglichen. Die Experimente zeigen, dass die Scattering Transformation, im Fall der sequentiellen Kombination, gleich gute Resultate wie die konventionellen Ansätze aufweisen. Im Fall von Daten mit viel Hintergrundrauschen ist sie etwas schlechter, im Fall von Translation der Objekte wesentlich besser. Wenn nur wenige Daten vorhanden sind und die Objekte spezifische geometrische Eigenschaften haben ist die Scattering Transformation ebenfalls überlegen. Damit ist sie ein sinnvolles Werkzeug für manche Anwendungen.

Contents

1	Introduction	11
1.1	Related Work	13
2	Theory	15
2.1	2D Image Processing	15
2.2	Fourier Transformation	16
2.2.1	One Dimensional FT	17
2.2.2	Two Dimensional FT	17
2.3	Scattering Transform	18
2.3.1	Filter Bank	19
2.3.2	Scattering Paths	20
2.3.3	Scattering Networks	21
2.3.4	Properties of the Scattering Transform	23
2.4	Convolutional Neural Networks	26
2.5	Hybrid Networks for Classification	28
2.6	Object Detection	28
2.6.1	Single Shot Multibox Detector (SSD)	29
2.7	Hybrid Networks for Object Detection	32
2.7.1	Sequential Hybrid Networks	32
2.7.2	Parallel Hybrid Networks	32
3	Experiments	35
3.1	Datasets	35
3.1.1	PASCAL VOC	35
3.1.2	KITTI	36
3.1.3	Toy Data	37
3.1.4	Test Invariances Toy Data	37
3.1.5	Classification Toy Data	39

3.2	Setup	40
3.2.1	Augmentations, Batchnorm, Multibox Loss, Optimizer	40
3.2.2	Hyperparameters	41
3.3	Classification	42
3.4	Object Detection	43
3.4.1	Baseline Performance of the SSD	43
3.4.2	Sequential Scattering	44
3.4.3	Parallel Scattering	44
3.4.4	Small Data Experiments	45
3.4.5	Timing experiments	46
4	Results	47
4.1	Classification	47
4.2	Object Detection	47
4.2.1	Baseline Performance of the SSD	48
4.2.2	Sequential Scattering	50
4.2.3	Parallel Scattering	50
4.2.4	Comparison of the Techniques	52
4.2.5	Small Data Experiments	52
4.2.6	Timing Evaluation	53
5	Conclusion	55
5.1	Future Outlook	56
6	Appendix	57
6.1	Negative Results	57
6.2	Explanations of the Scattering Transform	57
6.3	Results	57

1 Introduction

Many safety critical application require a visual understanding of the scene. For example, in autonomous driving we need to know what objects are on the road. In the task of object detection, objects are enclosed by rectangular bounding boxes and an object detection model is tasked with predicting these given an image. Even though the task is very easy for humans in most situations it is very hard for computers for two main reasons: Firstly, human visual systems have evolved over millions of years and have become very efficient at identifying important information quickly and processing them in an efficient manner. Secondly, a big part of object detection is incorporating prior knowledge, e.g. knowing what a car looks like in different circumstances and having abstract semantic knowledge of objects. It is unclear to what extent artificial systems are able to learn semantic concepts on this level of abstraction. However, in recent years computer vision algorithms have gotten significantly better for a number of reasons. Algorithms, especially in Deep Learning (DL), have evolved quickly and increased the accuracy for many complicated visual tasks drastically. Additionally, hardware has become faster and more algorithms can be partly or entirely parallelized, making it possible to run them on GPUs increasing their computation speed. Lastly, more and more big datasets have become publicly available. Since many DL algorithms need many samples to train on, having more and bigger datasets was an important pillar for their success.

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.

As outlined in the first paragraph many DL algorithms often rely on big datasets to train on. These might not always exist or are very expensive to annotate. Some diseases, for example, are rare and therefore only a few datapoints exist. Medical studies are often very expensive and can therefore only feature a couple of hundred patients. Additionally, at least for visual tasks, the data needs to be annotated. This is paid human labor that can cost a company much money. Over all it is therefore

important to find methods that reduce the number of training samples while keeping the high accuracies of current state of the art methods. One possible way to decrease the number of samples is to incorporate prior knowledge. This could be done by using theoretical properties, like invariances or equivariances w.r.t. specific transformations and combine them with DL. A possible way to realize this is shown by J. Bruna and S. Mallat [BM12] with the introduction of the Scattering Transform. The Scattering Transform is a new technique that uses wavelet operations on the image by using a static filter that requires no training. This is important because state of the art (SOTA) object detection algorithm use convolutional neural networks (CNNs). Filters used in those CNNs are all trained during the training period which costs time and energy. J. Bruna and S. Mallat [BM12] apply the Scattering Transform to images and perform classification tasks on their outputs. They also show that the technique is essentially equivalent to using CNNs with fixed weights for some or all filters. The reason why the Scattering Transform has proven so successful are the properties it provides. It is invariant to deformation and equivariant to translation. Equivariance is fulfilled for an algorithm if the output and the input are changed in a corresponding fashion when a transformation is applied. For object detection invariance with respect to translation is an undesirable property while equivariance is desired. Whether an object is on the left side of the image or the right one should be reflected in the outcome of the algorithm. In this work it is also argued that it has desirable properties w.r.t rotation and scaling, i.e. that the scattering transform is able to reproduce the size or rotation of an object in its output. 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 while also indicating those properties. A pedestrian that is close must be encoded differently from a pedestrian that is far away or otherwise the car runs her over or breaks without necessity.

The usage of static filters provides three specific advantages beyond their transformation properties. 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.

This work investigates the useful properties of the scattering transform and combines it with already established state of the art object detection algorithms. This is 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 extend 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 related work described in the following section.

The paper is separated in three main parts. A Theory chapter which presents an introduction to the techniques used in this paper and a more detailed analysis of the Scattering Transform. It is followed by an Experiment chapter which contains a detailed overview of the experimental setups for this work. Lastly, the Results chapter analyzes the experimental results and discusses them.

1.1 Related Work

The Scattering Transform has been applied successfully in many different classification tasks. [SM13a] 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.

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 2.3.

The first Section 2.1 gives a general overview of image processing for readers unfamiliar with the concept. Then the Fourier transform is described in 2.2 as an easier example of a filter that can be applied to images and changes the format of the their representation. In Section 2.3 the Scattering Transform is introduced and its theoretical properties explained. It is followed by a general introduction to convolutional neural networks (CNNs) in Section 2.4 after which hybrid networks in the context of classification are described in Section 2.5. Lastly, object detection is introduced in Section 2.6 and the hybrid networks in the context of object detection conclude the chapter with Section 2.7.

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.

¹Figure taken from <https://plus.maths.org/content/fourier-transforms-images>

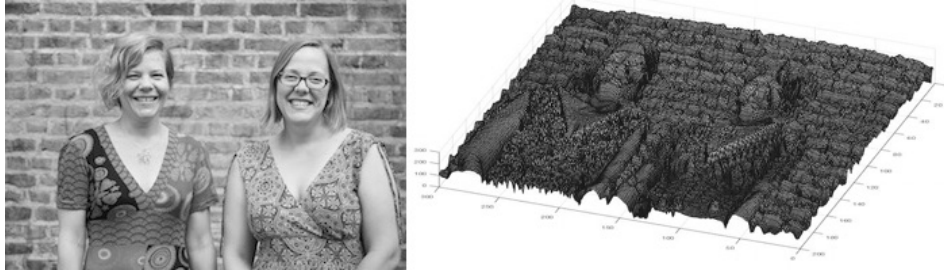


Figure 2.1: Left: image represented as 2D flat surface. Right: image as 3D terrain with uneven surface. ¹

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.

$$f = a \sin(h \cdot x + k \cdot y) \quad (2.1)$$

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 observed that higher amplitudes dominate the resulting wave, i.e. determine the direction of the wave stronger than the smaller amplitudes.

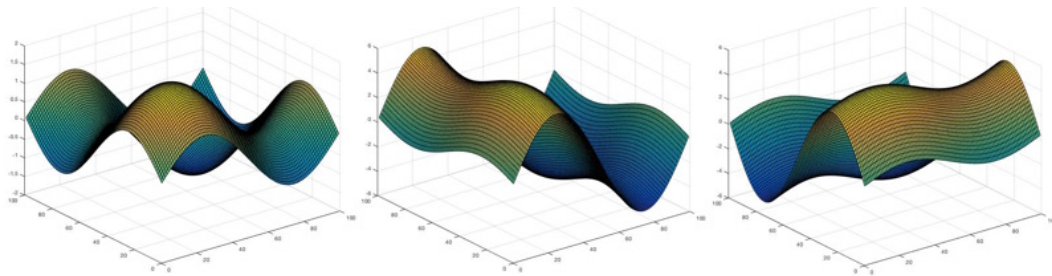


Figure 2.2: Left: $\sin(x) + \sin(y)$. Middle: $5 \sin(x) + \sin(y)$. Right: $\sin(x) + 5 \sin(y)$. On the middle and right images the higher amplitudes of 5 dominate the resulting wave. ²

2.2 Fourier Transformation

To get a better understanding of the Scattering Transform an explanation of a simpler and more commonly used transform is briefly discussed first in form of the Fourier

²Figure taken from <https://plus.maths.org/content/fourier-transforms-images>

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

2.2.1 One Dimensional FT

In the case of one dimensional signals, such as an audio signal, 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 $f : \mathbb{R} \rightarrow \mathbb{C}$.

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

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

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

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.

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

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

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.

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

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

2.3 Scattering Transform

This section explains the details of the Scattering Transform. It starts by explaining the basic functionality and adds functionality in every subsection. Subsection 2.3.1 describes a combination of multiple wavelet filters called filter bank. Subsection 2.3.2 shows how wavelets can be combined sequentially. The scattering networks described in Subsection 2.3.3 are a parallel and sequential combination of the wavelet filters that is also later used in the experiments. The section is concluded by a presentation and discussion of the properties of the Scattering Transform in Subsection 2.3.4.

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.

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

where C_1 and C_2 are constants. C_2 is chosen such that $\int \psi(u) du = 0$, $u \cdot \xi$ denotes the inner product of u and ξ , and $|u|^2$ is the norm in \mathbb{R}^2 . Figure 2.3 shows the real and complex part of a Morlet wavelet.

Similar to the Fourier transform the Morlet wavelet can also be extended to multiple dimensions. Figure 2.4 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.

³Figure taken from <https://pdfs.semanticscholar.org/c354/c467d126e05f63c43b5ab2af9d0c652dfe3e.pdf>

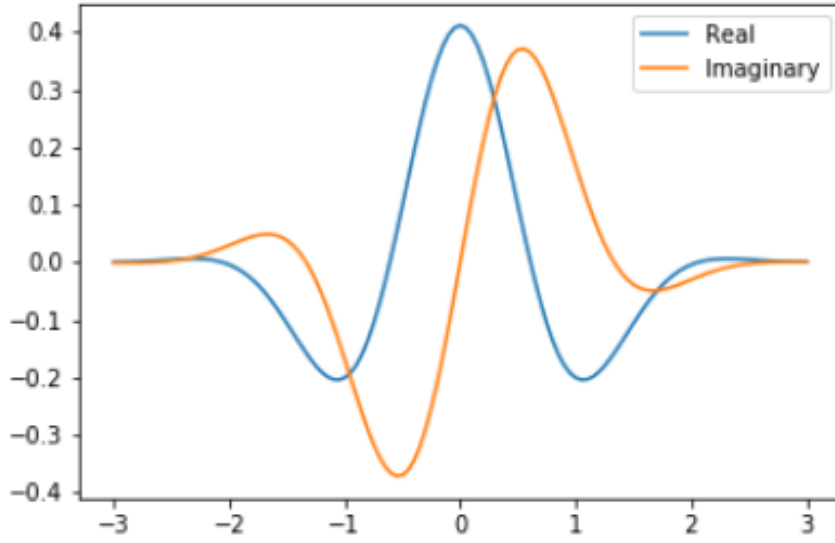


Figure 2.3: Real and Complex part of Morlet wavelet in 1D. ³

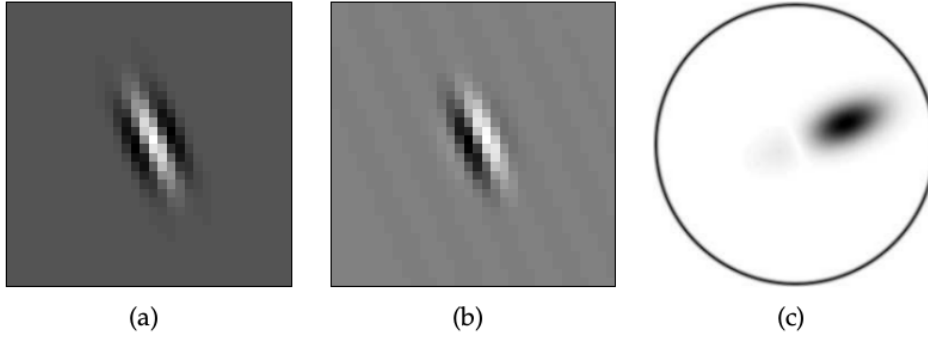


Figure 2.4: Complex morlet wavelet. a) Real part of ψ . b) Imaginary part of ψ . c) Fourier modulus $|\psi|$. Image taken from [BM12].

2.3.1 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^{2J} , 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 2.5. 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 sinusoidal wave multiplied with a Gaussian function. The Gabor Filter is used as a low-pass filter.

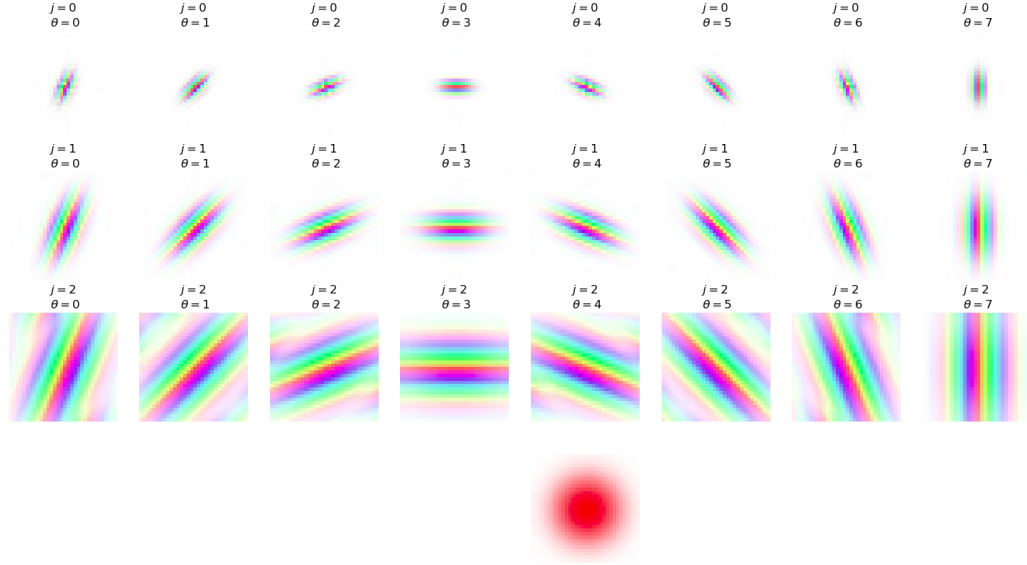


Figure 2.5: Visualization of the filter bank. The $j = 0, 1, 2$ describe the different downsample sizes. The $\theta = 0, \dots, 7$ describe the different angles. For these images $N, M = 32, L = 8, J = 3$. The contrast corresponds to the amplitude and the color to the phase. The blurry red dot in the bottom is the corresponding low-pass filter.

2.3.2 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 $p = (\lambda_1, \lambda_2, \dots, \lambda_m)$ defines a *path*, which is the ordered product:

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

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 $\tilde{S}x(p)$ is invariant to a translation of x . The scattering is Lipschitz continuous to deformations as opposed to the Fourier transform modulus.

2.3.3 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 . The scattering network is a collection of filters through which data can be forwarded as in any other computational graph. In contrast to most conventional CNN architectures (some use skip connections) the 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 2.6. 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 k_1, k_2 and therefore also the total number of outputs per datapoint are shown in Equation 2.9 for $m = 1$ and in Equation 2.10 for $m = 2$.

$$k_1 = i \cdot (1 + JL) \quad (2.9)$$

$$k_2 = i \cdot (1 + JL + \frac{1}{2}J(J-1)L^2) \quad (2.10)$$

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 2.6 is only showing a network for one abstract J . Given that J describes the factor by which the outputs are downsampled 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 $J = 2$.

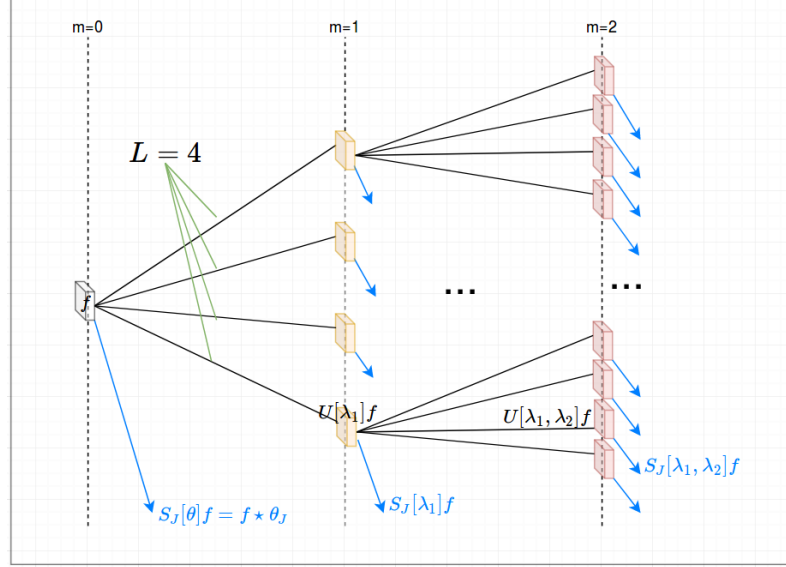


Figure 2.6: Representation of a scattering network. Each filter is described by its path, e.g. $U[\lambda_1]f$. Every blue arrow describes an output at the particular filter. The $m = 0, 1, 2$ describe the number of iterative applications of the scattering transform. $L = 4$ is the branching factor in the network also known as the number of angles for the filter. The filters

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:

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

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 a meaning that is understandable in human terms and can be analyzed visually. For two different images the 0th order, first order and second order scattering coefficients have been visualized in Figure 2.7 and 2.8. Subfigure a) shows the original image. Subfigures b),c) and d) show the 0th to second order scattering coefficients respectively. The number of filters grows as described in Equation 2.9 and Equation 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 2.8. 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. 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 and is also in alignment with the description of the following Subsection 2.3.4 when the distinction between local invariance and global equivariance is made.

2.3.4 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 we refer the reader to the original paper [BM12].

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. Before the properties can be explained the concepts of invariance and equivariance are defined. A function f is invariant w.r.t to a transformation T if the transformation does not change the outcome of the function when applied to the input x as described in 2.11.

$$f(Tx) = f(x) \quad (2.11)$$

A function f is equivariant w.r.t to a transformation T if the outcome of the function when applied to the input is transformed in a similar fashion as the input as shown in 2.12. Equivariance with respect to translation, for example, means the feature map

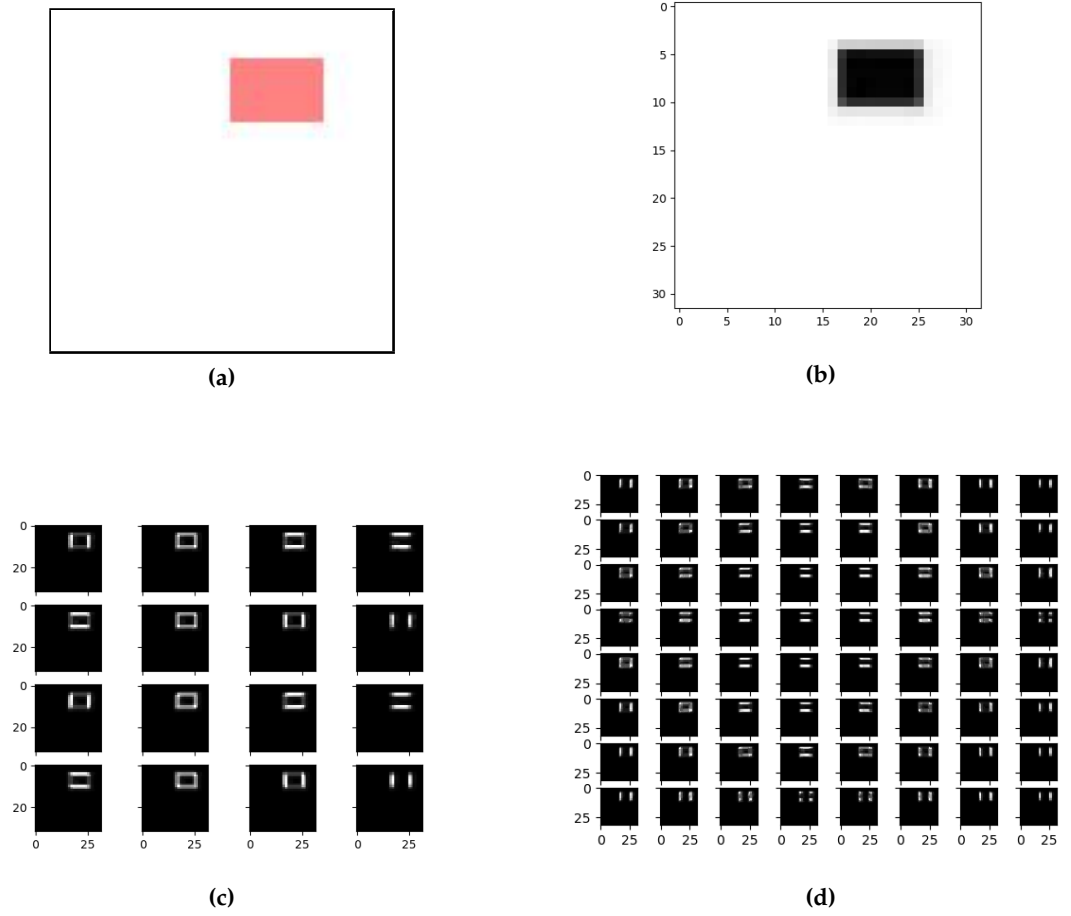


Figure 2.7: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

of the filter reflects that translation in a proportional manner.

$$f(Tx) = Tf(x) \quad (2.12)$$

A wavelet transform computes convolutions with wavelets. It is thus translation equivariant not invariant. To achieve local invariance a non-linearity must be added. The $L^1(\mathbb{R}^2)$ norm is chosen as the non-linearity. The invariance also only holds locally for displacements c with $|c| \ll 2^J$. This means that during the reduction to smaller feature maps the scattering transform does not lose any information on the object but still keeps global equivariance.

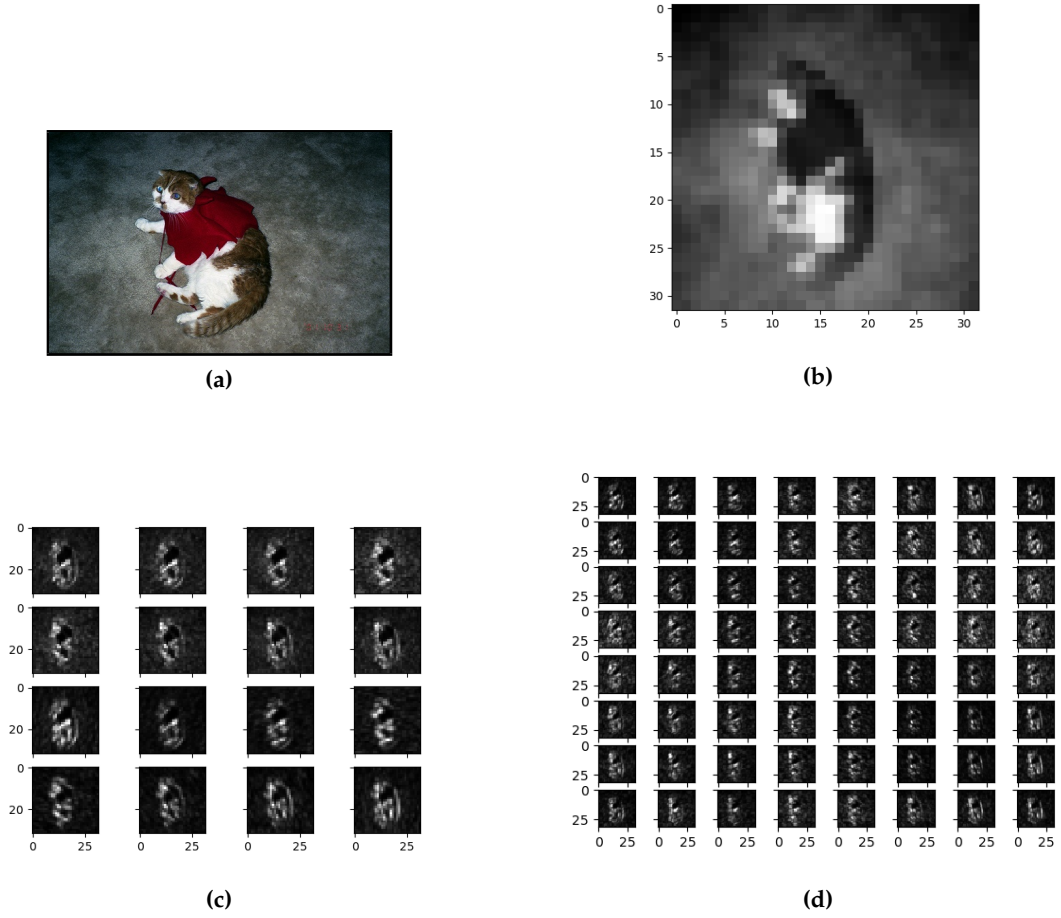


Figure 2.8: Image taken from the VOC dataset. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

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. Deformation stability of the Scattering Transform is obtained with localized wavelet filters which separate the image variations at multiple scales and orientations.

L. Sifre and S. Mallat [SM13b] show that the Scattering Transform also naturally fulfills global equivariance while having local invariance w.r.t. scale and rotation because of the Morlet wavelet.

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 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 invariances can be problematic. If a transform is applied to an object the object detection algorithm should still be able to identify the object correctly thereby showing invariance but also produce a new and adapted bounding box thereby showing equivariance. Since the Scattering Transform shows local invariance and global equivariance the problems of invariance are not given as long as J is sufficiently small, i.e. the image is not reduced too much. The empirical results, i.e. figures 2.7, 2.8 and appendix suggest translation equivariance instead of invariance. Equivariance is a desirable property for object detection and therefore not problematic. Similar results are also seen for scale as shown in the appendix. 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 which is described in Subsection 2.7.2.

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 2.9. 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 per layer. That means at every stage of the network different filters are applied on

⁴Figure taken from and animated version at <https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d>

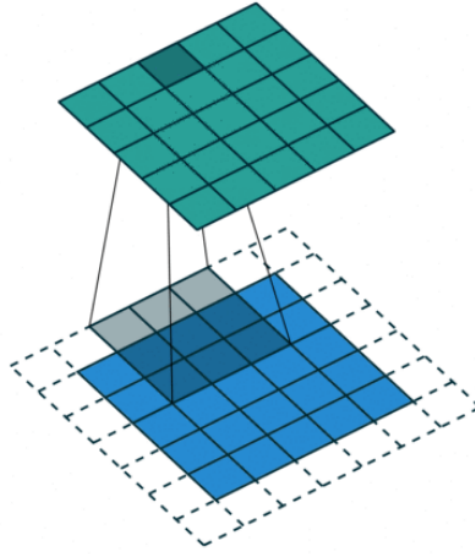


Figure 2.9: 3x3 convolution on 5x5 image. Resulting image is also 5x5 due to the padding of size 1 added to the original image. ⁴

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 are trained by backpropagation using stochastic gradient descent. They determine what each individual filter does. This means that every filter that is 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. Static filters essentially encode prior knowledge which is important for two key reasons. First, it lessens the dependency on big labeled datasets since the filters already contain some information and do not need to adapt. Second, specific theoretical guarantees can be made for some filters. A filter can be invariant or equivariant w.r.t. a specific transformation. Equivariance is a desirable property for object detection while invariance is undesirable. The bounding boxes for an image with one object and another image with the same but transformed object should not be similar but also translated accordingly. The Scattering Transform is one approach that ensures equivariance/invariance w.r.t.

specific transformations as explained in Subsection 2.3.

2.5 Hybrid Networks for Classification

In [OBZ17] E. Oyallon et al construct a hybrid of the Scattering Transform and classic classification architectures. They remove the first couple of layers of a ResNet [HZRS15] architecture and substitute it with a scattering network such that the image is first transformed by the Scattering Transform and the ResNet is then trained on the scattering coefficients. Their results show two important insights. First, the accuracies on most classification tasks such as MNIST are as good as when just training a ResNet architecture. Second, when using less datapoints or images with distortions the hybrid network outperforms the standard architecture. This paper attempts to extend these results on object detection which is explained in the following section.

2.6 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 2.10.

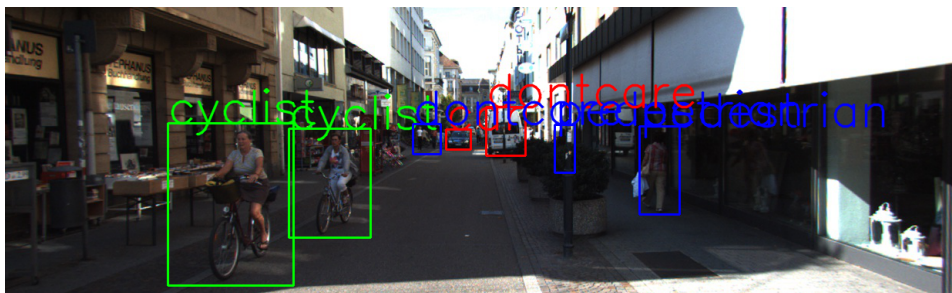


Figure 2.10: An example of a annotated image taken from the Kitti dataset. There are seven objects of four classes visible: cyclist, car, pedestrian and dontcare.

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) in different ways. Broadly speaking object detection networks can be distinguished in one-stage and two-stage detectors. One-stage detectors are fully convolutional networks, e.g. YOLO [RF18] or SSD [LAE⁺15]. Two stage object detection networks have a region proposal method as their first part. This method can either be selective search as in the case of RCNN [GDDM13] or a region proposal network as in FastRCNN [Gir15] or FasterRCNN [RHGS15]. The second stage then consists of classifying the objects in the proposed regions and finding the most fitting bounding boxes. In this stage all previously named network types use a CNN for the classification. Therefore this work also uses a CNN as the backbone of the object detection. The object detection network that is used for this paper is discussed in the following subsection.

2.6.1 Single Shot Multibox Detector (SSD)

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 (NMS) operation to put a higher weight on probably correct detections and a lower weight on probably incorrect detections. Additionally NMS also merges all detections belonging to the same class and only keeps the bounding boxes with highest confidence for a given region thereby removing redundant encodings of the same object. A detailed graphical description of the SSD and VGG16 can be found in figure 2.12. 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. Anchor boxes are predefined boxes offered to the network such that it only has to choose the best fitting one instead of suggesting its own box proposals. The concept of anchor boxes and feature maps is visualized in Figure 2.11.

Some of the combinations of feature sizes and aspect ratios will match an object

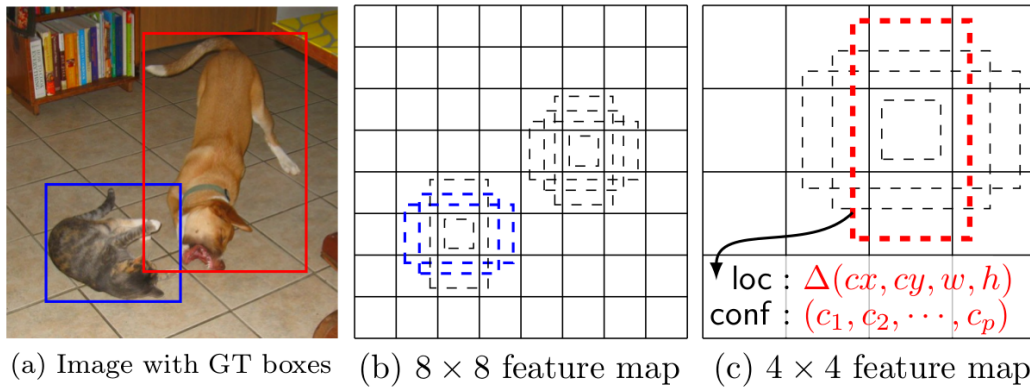
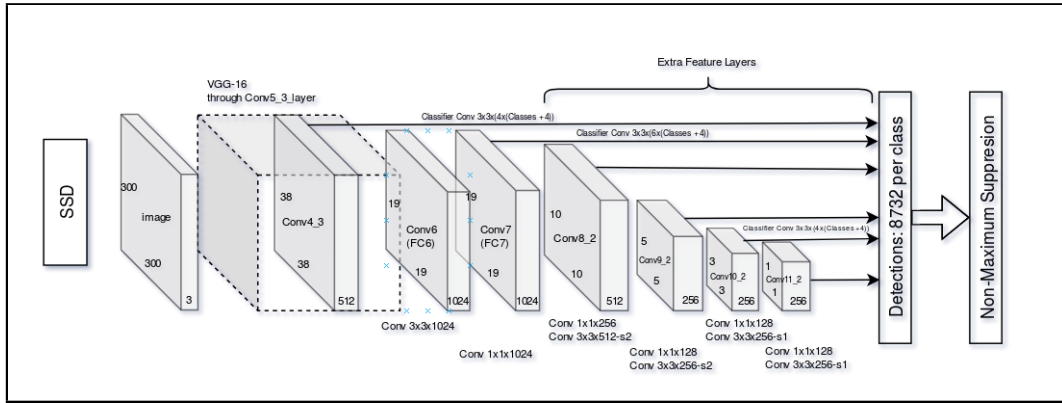


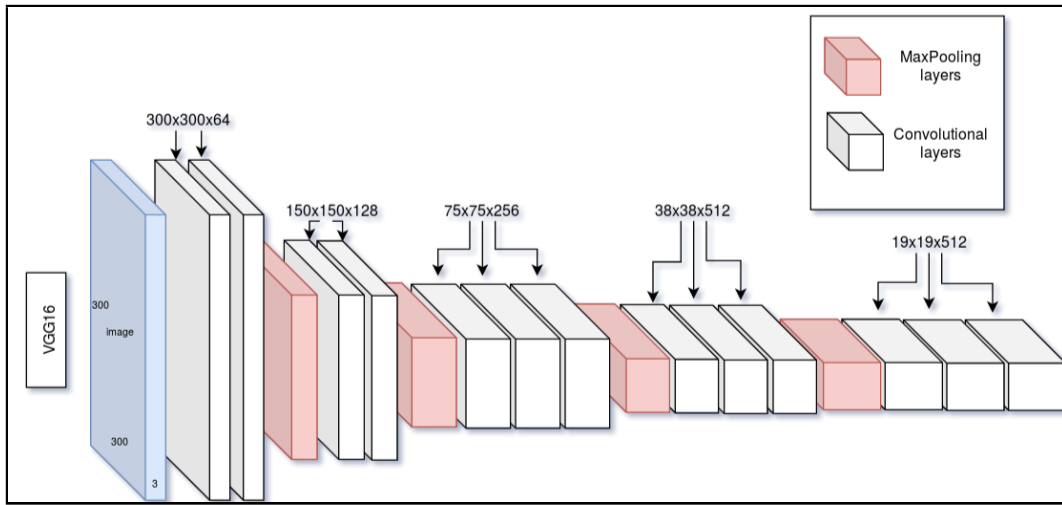
Figure 2.11: A graphical explanation of the feature maps and anchor boxes. The grid shows the feature map. The boxes with dotted contours describe different possible anchor boxes for one entry on the grid. Image from the original SSD paper [LAE⁺15].

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 as a consequence of all convolutions described in Figure 2.12. 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 2×2 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 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



(a)



(b)

Figure 2.12: 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 $N \times N \times C$ stand for the feature map size and number of channels, i.e. conv_2_1 has a feature map of 150×150 with 128 parallel channels. The red boxes describe 2×2 max-pooling operations.

suggestion for future work but is not part of this paper.

2.7 Hybrid Networks for Object Detection

Hybrid networks in the context of this paper describe a combination of conventional object detection network architectures and the scattering transform. Note that this is similar to the hybrid networks as originally proposed by [OBZ17] but extended to object detection. Two possible ways to combine the two techniques are described in the following. Subsection 2.7.1 describes the extension of the hybrid networks to object detection. Subsection 2.7.2 shows a new technique combining the two methods in a different way.

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 2.13. The details of the implementation are found in figure 3.9 in the following chapter.

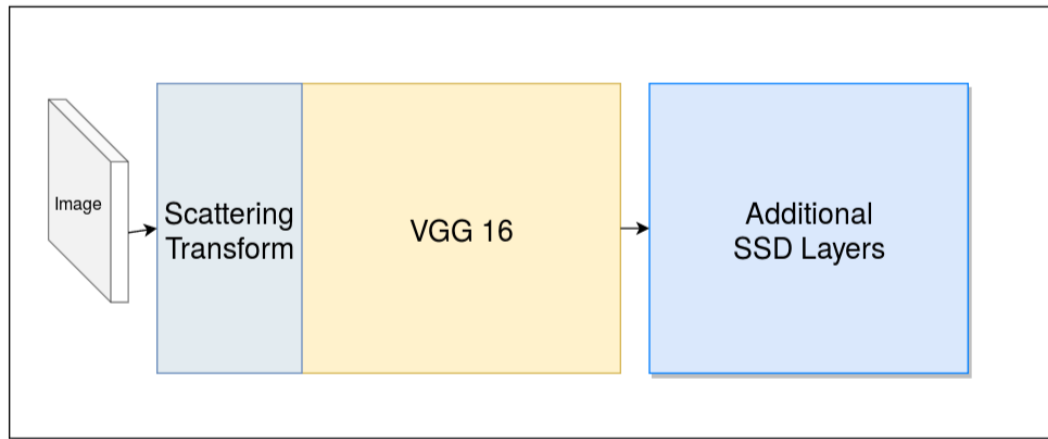


Figure 2.13: General overview of the sequential scattering architecture.

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 2.14 The details of the implementation are found in figure 3.10 in the following chapter.

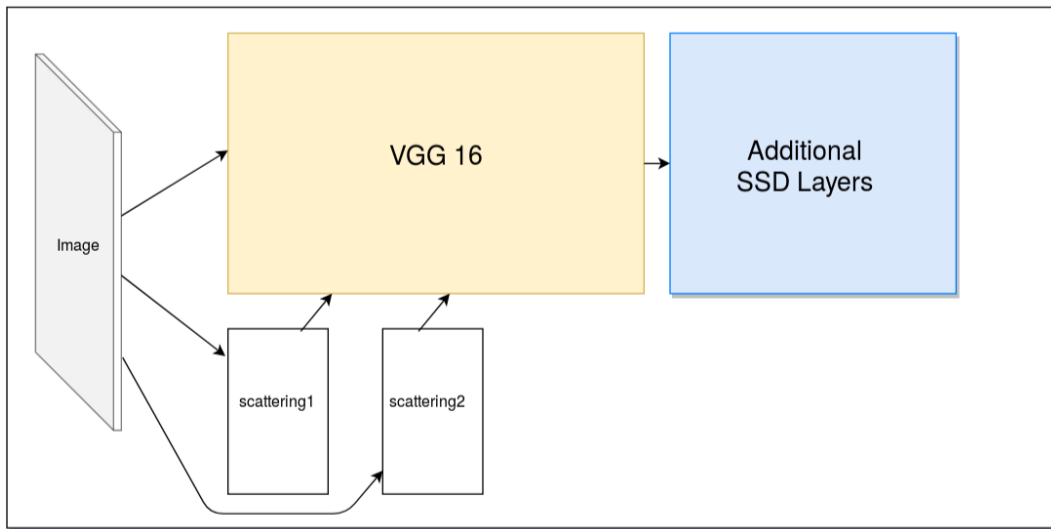


Figure 2.14: General overview of the parallel scattering architecture.

3 Experiments

Multiple experiments are 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.

Section 3.1 gives a general overview of the datasets used in this paper. Section 3.2 shows what kind of setup was used for the architectures to conduct the experiments. The reproduction experiments of the classification results on different datasets is set up in Section 3.3 where the main experiments for object detection are described in Section 3.4.

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.

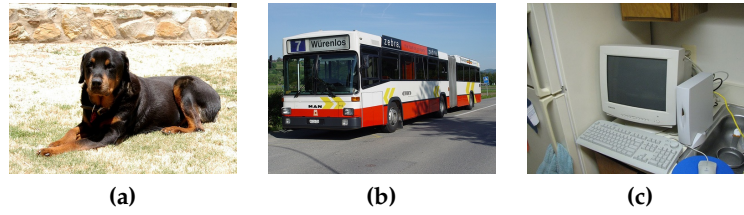


Figure 3.1: Three samples from the PASCAL VOC dataset showing a dog, bus and TV monitor from left to right.

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 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. The SSD setup used in this paper uses an input size of 300 by 300 thereby reshaping the image and most objects in it which could pose problems. A SSD with input size of 1000 by 300 was setup but did not yield better results (see 6.1 for the description of a negativ result). Four traffic scenes are depicted in 3.2.



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 overlaid. 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

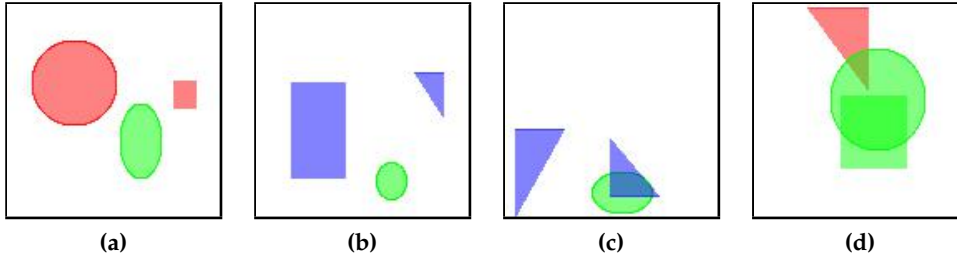


Figure 3.3: Four samples from the toy data set.

3.1.4 Test Invariances Toy Data

To test the 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. Six further images are derived from the base image w.r.t. the particular invariance and put into the test set. 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. Six further images are created by scaling the object 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.

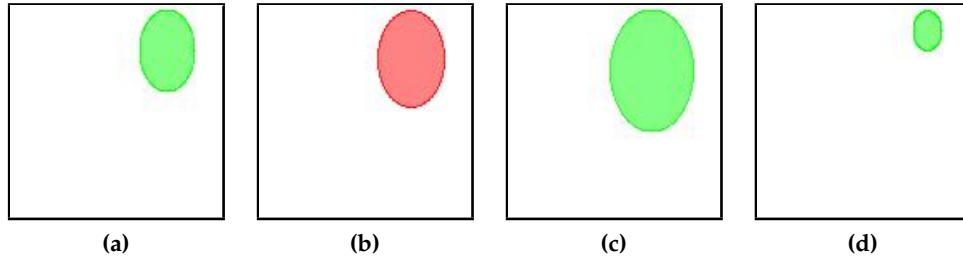


Figure 3.4: Four samples from the scale toy data set. a) is the base image; b) -d) are the scaled versions

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.

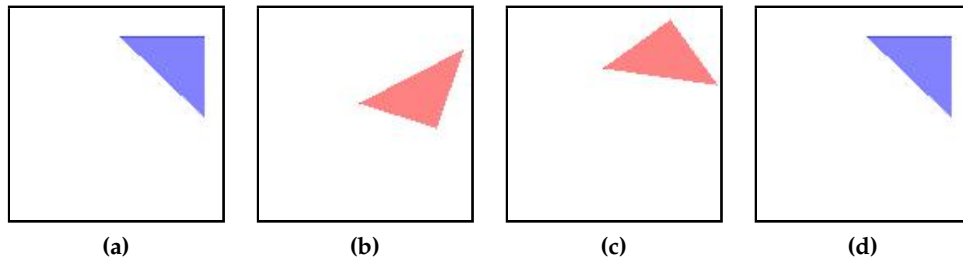


Figure 3.5: Four samples from the rotation toy data set. a) is the base image; b) -d) are the rotated versions

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.

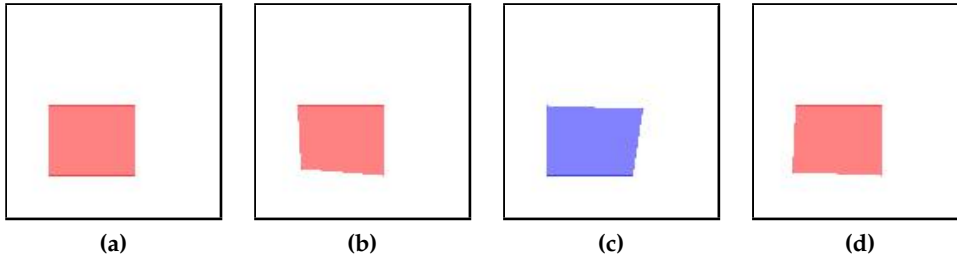


Figure 3.6: Four samples from the deformation toy data set. a) is the base image; b) -d) are the deformed versions

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.

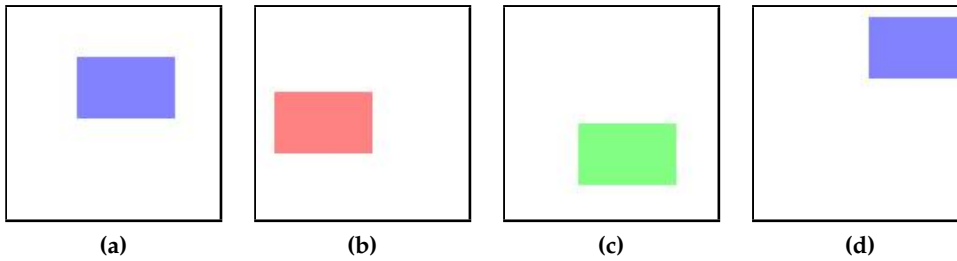


Figure 3.7: Four samples from the translation toy data set. a) is the base image; b) -d) are the translated versions

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. The data from the object detection toy data cannot be reused because they contain three objects per image. Samples of the images are shown in figure 3.8.

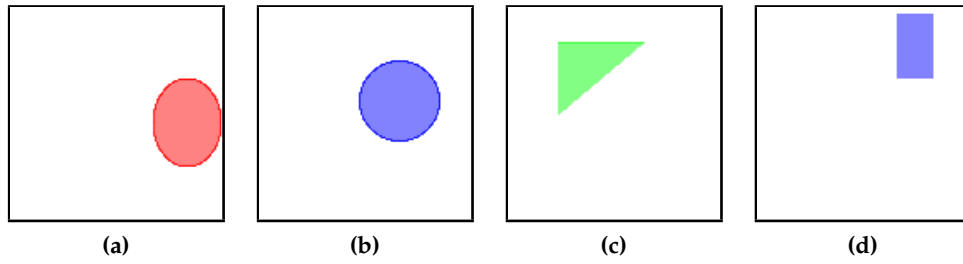


Figure 3.8: Four samples from the classification toy data set.

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 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 are reshaped to fit the SSD setup to have a size of 300x300 pixels with three color channels. A data augmentation preprocessing step 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

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

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.

The training objective is optimized using Stochastic Gradient Descent (SGD). Other optimizers such as Adam were tried but did not change the result in any way.

3.2.2 Hyperparameters

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 2.12. 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 and anchor boxes can be found in Figure 2.11 shown in the SSD Subsection. 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: $\text{feature_maps}[i] \cdot \text{steps}[i] \approx 300$. 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 Object Detection

In the following section the main experiments of the paper are described. Subsection 3.4.1 describes the experiments to determine the baseline results and hyperparametrization for the standard SSD approach. Subsection 3.4.2 and 3.4.3 give a detailed explanation of the setup for the sequential and parallel scattering experiments. Lastly, Subsection 3.4.4 explain the experimental setup for small datasets and short training times and Subsection 3.4.5 sets up an experiment where the time consumption for all forward passes is measured.

3.4.1 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 [DDS⁺09]. The datasets are KITTI, PASCAL VOC and the Toy data set. 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. It must be pointed out that the same experiments but in smaller scale have been conducted for scattering methods to prevent unfair comparisons. Whenever the parametrization differs it is pointed out in the respective Section of the Results chapter. 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.

3.4.2 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 75×75 since $J = 2$ was chosen. The adapted VGG can be seen in figure 3.9. 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.

3.4.3 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 3.10.

The experiments consisted of training a parallel 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

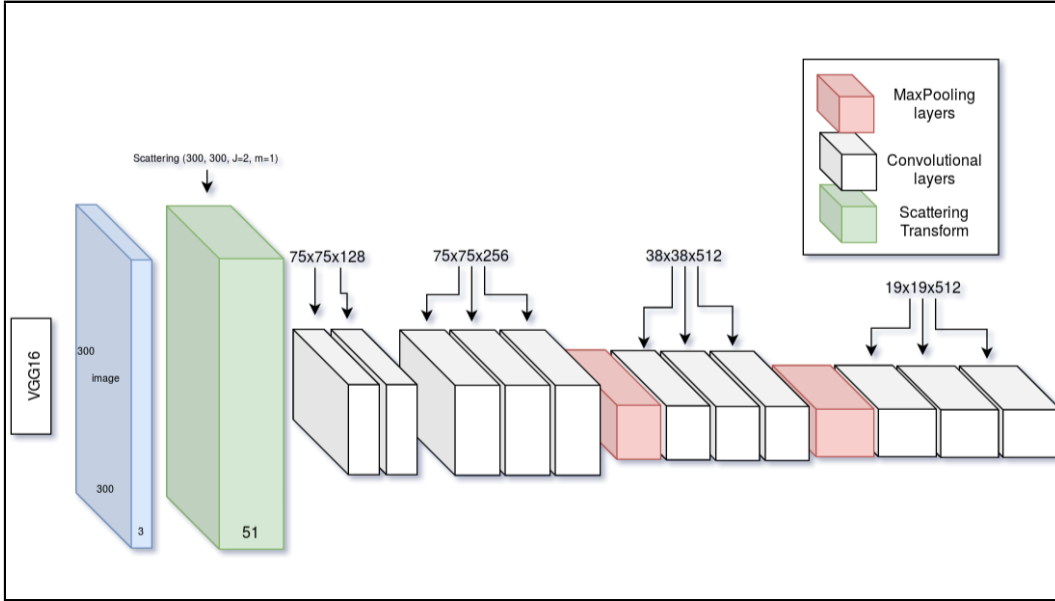


Figure 3.9: Sequential scattering SSD. The first two filters are replaced by a scattering transform with $J = 2$ and $m = 1$. This means the output has $3 \cdot 17 = 51$ output channel.

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 for the experiments.

3.4.4 Small 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

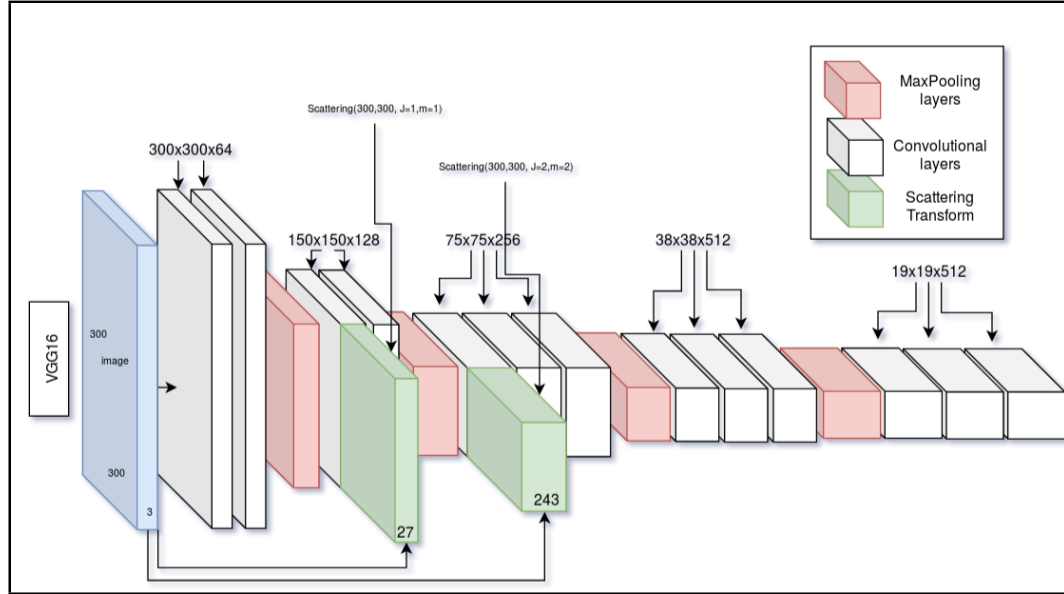


Figure 3.10: Parallel scattering SSD. The image is piped through the normal VGG path and additionally through a scattering transform with $J = 1$ and $m = 1$ and a second one with $J = 2$ and $m = 2$. The outputs of the scattering transforms are concatenated with the other pipe as soon as the dimensions are compatible.

5k epochs respectively.

3.4.5 Timing experiments

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. The experiment is conducted on a GeForce GTX 1080 Ti.

4 Results

In this chapter the results of all experiments are presented. Every result is put into perspective and discussed.

Section 4.1 shows the result of the classification experiments while Section 4.2 shows the results of the object detection experiments with all subexperiments such as small data experiments and timing experiments.

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 Object Detection

The results of the object detection experiments are presented in the following. First the baselines of the standard approach are established in Subsection 4.2.1. Then the experimental results of the sequential and parallel scattering approach are shown and discussed in Subsection 4.2.2 and 4.2.3 respectively. Lastly, Subsections 4.2.5 and 4.2.6 show the results of the small data and timing experiments respectively.

4.2.1 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.

Hyperparameterization and Baselines

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.

All results of the GLM can be found in the appendix in figure 6.4.

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

Table 4.1: Results of the baseline experiments. Mean and standard deviation are denoted for every combination of features that were measured. The results show that the combination of Augmentations and Pretrained true and Batchnorm false achieve the highest accuracy in all cases.

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

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 possibility to generalize is rather limited for a standard SSD network trained on a low number of training data points. All results of the GLM can be found in the appendix in Figure 6.5.

Table 4.2: Results of the invariant toy data experiments. Mean and standard deviation are denoted for every combination of features that were measured. Pretraining on ImageNet decreases the average accuracy slightly.

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

4.2.2 Sequential Scattering

The results of the sequential scattering experiments with augmentations and batch-norm set true can be found in Table 4.3. Pretraining on ImageNet does not make a significant difference for the resulting accuracy. The GLM in the appendix 6.6 even shows a negative sign of the responding coefficient. Hypothetically the pretrained weights could not only have learned the reoccurring contents of the images but also meaning that is independent of the representation of the data. Given the results of the experiments w.r.t. pretraining this hypotheses can not be supported.

The explicit comparison to both other techniques will be done in the Comparison Subsection ?? but two important differences stand out. The first is the major drop in accuracy of the PASCAL VOC dataset. While having 63 % accuracy was reached with the standard SSD, only 43.2 % was reached with sequential scattering. However, the standard setup was unable to identify objects after translation while the sequential scattering setup is able to achieve 85.7 % accuracy thereby providing evidence for the equivariance given by the Scattering Transform.

4.2.3 Parallel Scattering

The results for the parallel scattering experiments can be found in Table 4.4. It can be seen that pretraining on Image net has a positive impact on the results. This is also confirmed by the positive coefficient of the GLM that is reported in the appendix 6.7. Patterns similar to the sequential scattering approach arise, i.e. the accuracy on

Table 4.3: Results of the sequential scattering experiments. Mean and standard deviation of the accuracy are denoted for every combination of features that were measured. Pretraining on ImageNet does not add any benefit.

Dataset	Pretrained	Mean	Std_dev
VOC	0	0.399	0.004
VOC	1	0.432	0.038
Kitti	0	0.110	0.001
Kitti	1	0.077	0.003
Toy_data	0	0.831	0.000
Toy_data	1	0.818	0.001
Deformation_data	0	0.933	0.001
Deformation_data	1	0.929	0.001
Rotation_data	0	0.590	0.014
Rotation_data	1	0.659	0.059
Translation_data	0	0.857	0.000
Translation_data	1	0.767	0.127
Scale_data	0	0.646	0.000
Scale_data	1	0.641	0.003

PASCAL VOC is significantly below the standard SSD but the drop of the translation data in the standard approach is not existent in the parallel scattering setup.

Table 4.4: Results of the parallel scattering experiments. Mean and standard deviation of the accuracy are denoted for every combination of features that were measured. Pretraining on ImageNet does improve accuracy.

Dataset	Pretrained	Mean	Std_dev
VOC	0	0.146	0.0
VOC	1	0.476	0.0
Kitti	0	0.042	0.0
Kitti	1	0.105	0.0
Toy_data	0	0.745	0.0
Toy_data	1	0.804	0.0
Deformation_data	0	0.900	0.0
Deformation_data	1	0.912	0.0
Rotation_data	0	0.677	0.0
Rotation_data	1	0.613	0.0
Translation_data	0	0.841	0.0
Translation_data	1	0.846	0.0
Scale_data	0	0.642	0.0
Scale_data	1	0.641	0.0

4.2.4 Comparison of the Techniques

A final comparison between the best mean accuracies for all datasets and approaches can be seen numerically in Table 4.5 and visually in Figure 4.1. There are three key observations in the comparison. First, the results of the respective approaches differ in only two of the seven datasets. This is a positive result because it means that the scattering transform is a good alternative to the standard algorithms in object detection in most circumstances. Second, they differ in the results for PASCAL VOC where the standard approach outperforms both scattering approaches and they differ in the exact inverse direction for translation data. The first result is potentially due to the high background noise of natural images. The Scattering Transform is essentially an edge detector and the noise might produce a lot of edges in the scattering coefficients where none should be. In the case of translation data the standard approach is not general enough to learn the concept of object translation. The Scattering Transform, in contrast, already naturally contains prior knowledge in form of equivariance w.r.t. translation achieving significantly better results than the standard approach. The third observation is the comparison of the scattering techniques between each other. The picture shows that the respective lines of sequential and parallel scattering are nearly identical. This either means that the concatenation of the scattering results introduces an upper bound to the performance of the standard path or that the concatenation is done in a way in which the scattering results dominate the normal pathway. The last possibility is more likely in the context of this paper since the second concatenation of the scattering transform merges 243 scattering filters with 128 feature maps from the standard CNN path. At this point the information from the standard path ways might be ignored by later layers given their numerical minority. Future experiments could contain a concatenation with less scattering filters.

4.2.5 Small Data Experiments

The results of the small data experiments can be found in table 4.6. 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

Table 4.5: Comparison between the standard SSD, sequential scattering and parallel scattering. For each method the best mean accuracy for all datasets is reported and compared. The standard SSD outperforms both others on VOC. The inverse happens in translation data.

	Standard SSD	Sequential Scattering	Parallel Scattering
Kitti	0.125	0.110	0.1048
VOC	0.630	0.432	0.4760
Toy_data	0.792	0.831	0.8037
Deformation_data	0.928	0.933	0.9117
Rotation_data	0.635	0.659	0.6766
Translation_data	0.002	0.857	0.8459
Scale_data	0.644	0.646	0.6415

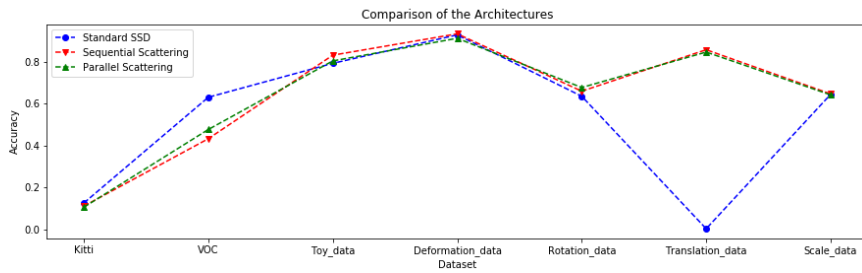


Figure 4.1: Comparison between the standard SSD, sequential scattering and parallel scattering. For each method the best mean accuracy for all datasets is reported and compared. The standard SSD outperforms both others on VOC. The inverse happens in translation data.

twofold: a) The sequential scattering setup is useful in specific use cases while having problems with very noisy and unstructured datasets such as VOC and b) The parallel scattering approach (at least with the setup used in this paper) does not really yield the expected results. Instead of providing the best of both techniques the parallel scattering gets outperformed on both datasets. All results of the GLM can be found in the appendix in figure 6.8.

4.2.6 Timing Evaluation

The results of the timing evaluation can be found in table 4.7. 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

Table 4.6: 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

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 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 4.7: 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. The sequential scattering is faster than the standard approach. Both are significantly faster than the parallel scattering.

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.

This paper extended the Scattering Transform from classification to object detection. To achieve this two techniques were used. The first is an extension of the hybrid scattering networks used in [OBZ17] where a scattering network and a CNN are combined sequentially. The second is a new technique introduced in this work where the results from a scattering network are merged with a CNN architecture at different stages. The evaluation of the new techniques was done on seven different datasets. Two of which, PASCAL VOC and KITTI show real world image scenarios. The other five, Toy_data, Deformation_data, Rotation_data, Translation_data and Scale_data are created just for this work to test performance on images who contain objects with specific properties. Besides the experiments on these datasets the techniques are also tested on very small datasets with very low number of training epochs. Lastly their forward pass is timed and compared to see whether the static nature of the filter yields significant decrease in training time.

The results of all those experiments yield many conclusions. The first conclusions can be seen when looking at Figure 4.1 in the comparison Subsection ?? . It shows that both scattering approaches match the performance of the standard approach in five out of seven cases, underperform in one and overperform in the other. This is likely due to the scattering transform handling geometric properties or transformations of objects better than the standard approach but having difficulties with objects in high noise environments such as real world datasets outside of controlled environments. When looking at the results of the small data experiments one can see that the sequential scattering approach outperforms the standard and parallel scattering approach for the toy data but both scattering architectures get beaten by the standard architecture for the PASCAL VOC datasets. This again indicates that the sequential scattering approach has an advantage for small datasets and short training times

when using data that contain objects with some geometric properties.

When comparing the timing, the sequential scattering is faster than the standard approach by taking only 75% as long per forward pass. The parallel scattering approach is significantly slower while not adding any additional accuracy.

Overall this leads to two clear conclusions. First, the parallel scattering approach as described in this paper is in no way better than any of the other two and therefore needs to be part of future research before being used for applications. Second, the sequential scattering approach is neither strictly better than the standard approach nor should it be applied in all cases. However, in some applications it might provide significantly more accurate and more robust results. When the data does not contain much natural noise and has certain geometric patterns the sequential scattering is likely better than the standard approach. Especially when only few data are available sequential scattering is more optimal. This is further amplified in scenarios in which time is a key bottleneck because the time for a forward pass is lower for the sequential scattering approaches. Scenarios where these properties are given could include images of rare diseases since few images exist or object recognition on text because of the clear shapes of characters.

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. Given that the aspect ratio of the images from the KITTI dataset are 3:1 we set up an SSD with input size 1000x300 and adapted the feature sizes accordingly. However, the results were comparable to or worse than the 300x300 version but took longer to compute and was therefore discarded.
2. The values called `s_sizes` in the hyperparameters Subsection 3.2.2 are calculated according to a minimum value of 0.2 and maximum value of 0.9 with a linear interpolation in between. We changed these values to 0.1 and 0.95 and other combinations. Non of them were more successful than the default ones.

6.2 Explanations of the Scattering Transform

Figure 6.1 shows a translated version of the image found in section 2.3.3.

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

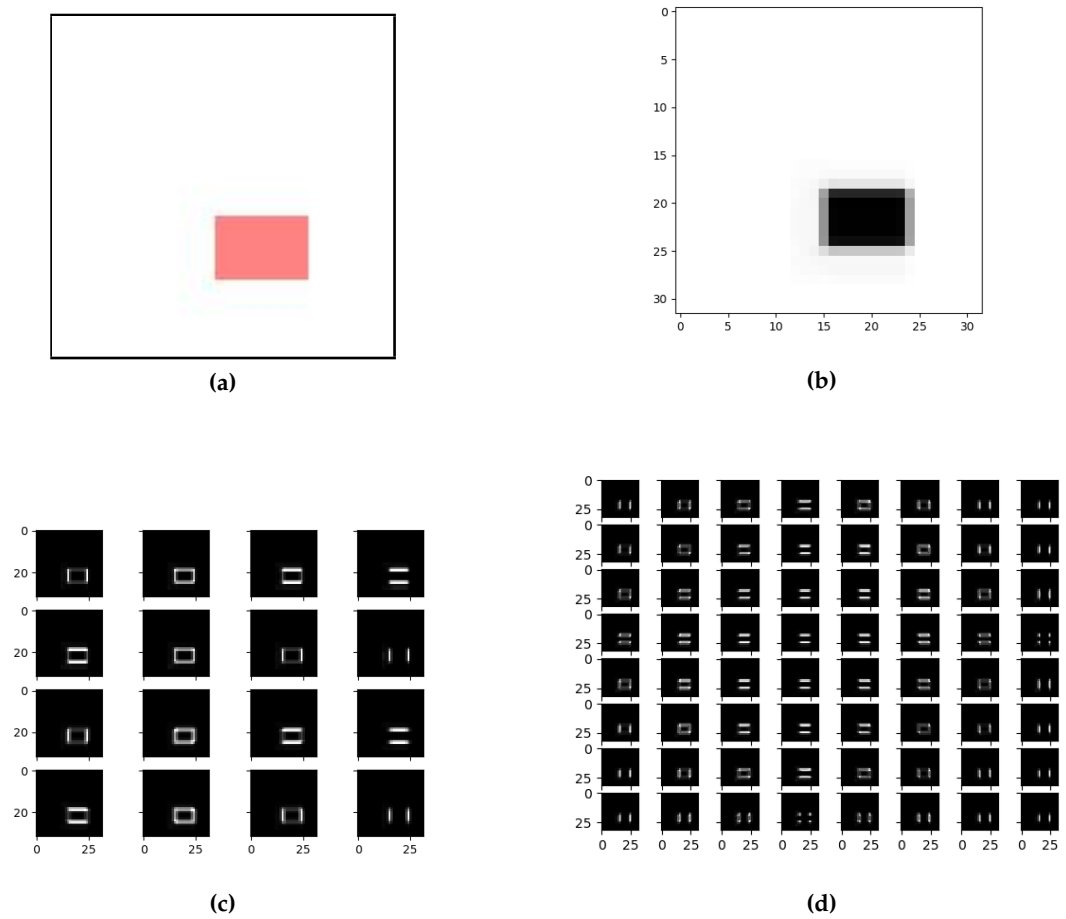


Figure 6.1: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

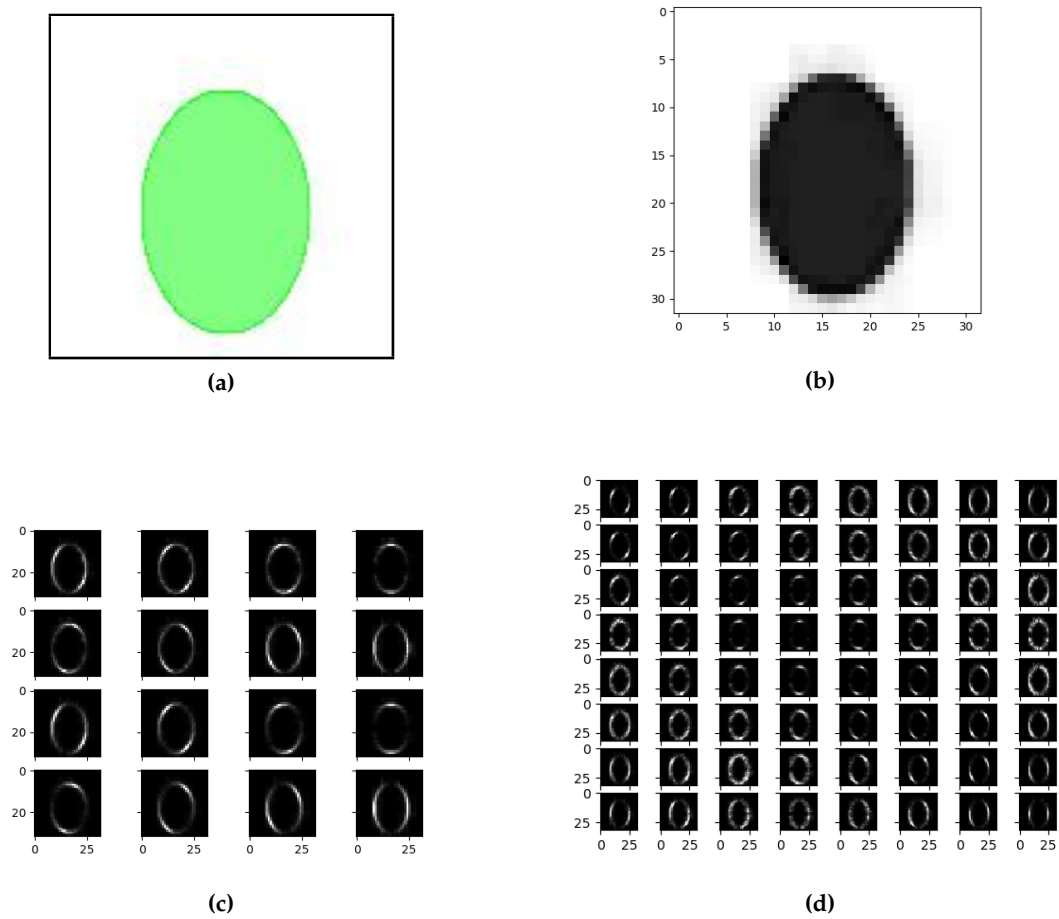


Figure 6.2: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

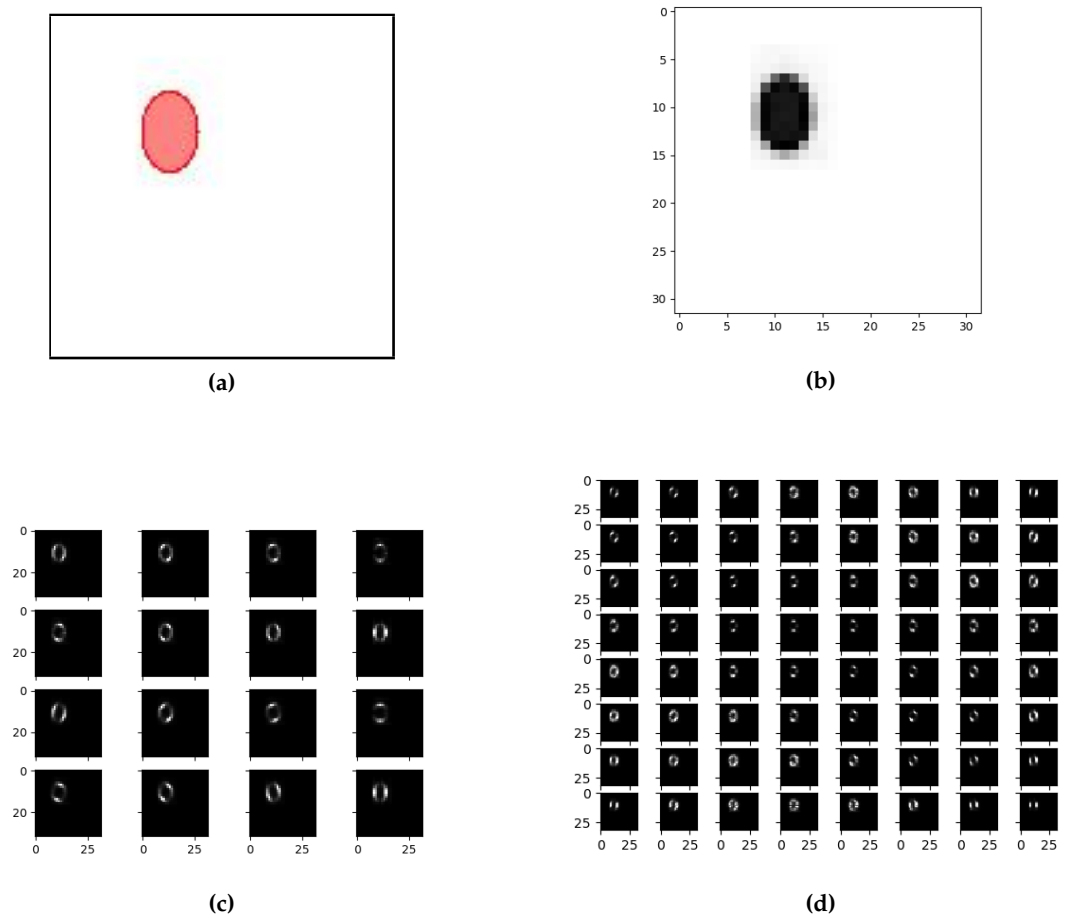


Figure 6.3: Image taken from a toy dataset created for this work. a) Original image; b) 0th order scattering coefficients, i.e. a Gaussian low-pass filter; c) First order scattering coefficients; d) Second order scattering coefficients

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:	40
Model:	GLM	Df Residuals:	32
Model Family:	Binomial	Df Model:	7
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-13.468
Date:	Fri, 14 Jun 2019	Deviance:	0.45398
Time:	16:16:06	Pearson chi2:	0.504

	coef	std err	z	P> z	[0.025	0.975]
Intercept	0.5339	0.515	1.037	0.300	-0.475	1.543
VOC	-0.8504	0.959	-0.887	0.375	-2.729	1.029
Kitti	-2.7801	1.273	-2.184	0.029	-5.275	-0.285
Toy_data	1.0364	1.005	1.031	0.303	-0.934	3.007
Deformation_data	2.0923	1.446	1.447	0.148	-0.742	4.927
Rotation_data	0.0003	0.825	0.000	1.000	-1.616	1.617
Scale_data	0.0821	0.832	0.099	0.921	-1.549	1.713
Translation_data	0.9533	0.983	0.970	0.332	-0.973	2.880
Pretrained	-0.0502	0.789	-0.064	0.949	-1.596	1.495

Figure 6.6: GLM for the sequential scattering experiments. Positive coefficients imply a better accuracy. P-values below 0.05 are significant.

Dep. Variable:	Accuracy	No. Observations:	14
Model:	GLM	Df Residuals:	6
Model Family:	Binomial	Df Model:	7
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-4.7429
Date:	Sun, 16 Jun 2019	Deviance:	0.24234
Time:	12:27:21	Pearson chi2:	0.234

	coef	std err	z	P> z	[0.025	0.975]
Intercept	0.2234	0.858	0.260	0.794	-1.458	1.904
VOC	-1.2057	1.484	-0.812	0.417	-4.114	1.703
Kitti	-2.9562	2.450	-1.207	0.228	-7.758	1.846
Toy_data	0.8385	1.605	0.522	0.601	-2.307	3.984
Deformation_data	1.8721	2.195	0.853	0.394	-2.431	6.175
Rotation_data	0.1984	1.439	0.138	0.890	-2.622	3.019
Scale_data	0.1826	1.437	0.127	0.899	-2.633	2.999
Translation_data	1.2936	1.809	0.715	0.475	-2.252	4.839
Pretrained	0.3595	1.338	0.269	0.788	-2.263	2.982

Figure 6.7: GLM for the parallel scattering 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 6.8: 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.
- [DDS⁺09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [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.
- [GDDM13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.
- [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.
- [RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.
- [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].
- [SM13a] Laurent Sifre and Stéphane 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.
- [SM13b] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1233–1240, 2013.