Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Stuttgart, February 27, 2023

Masterarbeit Nr. 96

Filter Dictionaries for Optical Flow Prediction with RAFT

Peter Walter

Course of Study:	Simulation Technology
Examiner:	Prof. Dr.-Ing. Andrés Bruhn
2nd Examiner:	Prof. Dr. rer. nat. Thomas Ertl
Supervisor:	M. Sc. Jenny Schmalfuss

Abstract

In the field of optical flow estimation, a dense vector field must be generated describing the apparent two-dimensional displacement of objects in consecutive images of a sequence. Although state of the art predictions are currently produced by deep convolutional neural networks, one major issue is that they are strongly susceptible to adversarial attacks, such as the Perturbation Constrained Flow Attack, which create small, noisy perturbations pursuing maximal change in the optical flow estimate. To improve adversarial robustness, this thesis includes receptive field convolutional layers into the optical flow predicting neural network RAFT. These receptive field layers use filter dictionaries to impose specific (geometric) priors onto convolutional kernels and improve results in image classification and reconstruction tasks. Each kernel in these RFCNNs can be written as a weighted sum over a fixed subset of filters taken from the dictionary. Besides the existing Gaussian derivative and Parseval completed sparse directional dictionaries, a novel PCA dictionary is proposed which consists of the principal components of the previously trained network's kernels. All types of dictionaries are compared against each other at multiple positions in the network. Results show that receptive fields in individual layers mostly do not affect and in RAFT's feature encoder even degrade performance, while Parseval completed dictionaries do not benefit the neural network in this context of optical flow. However, filter dictionaries with geometric motivations in RAFT's update block, namely the Gaussian derivatives and sparse directional FDs, make the network up to 20% more robust against the PCFA in exchange for a worse fit in quality.

Keywords Filter dictionaries · Receptive fields · Neural networks · Optical flow · Adversarial robustness

Contents

1 Introduction

Optical flow refers to the apparent 2D displacement of correspondences on a per-pixel basis between a source and a target frame in an image sequence or video. In numerous applications, optical flow plays an essential part by providing such dense fields of correspondences. These fields can then be further used in downstream tasks, such as video inpainting [Kim et al., 2019; Xu et al., 2019], frame interpolation [Liu et al., 2020; Huang et al., 2022b], action recognition [Ullah et al., 2018; Zhao et al., 2020b; Ilic et al., 2022] and autonomous driving [Wang et al., 2021], or serve as a basis for human interpretation in medical applications [Yan et al., 2019; Li et al., 2022; Yu et al., 2020; Tehrani et al., 2020].

To estimate the optical flow, traditional optimization techniques, for example the method of [Horn and Schunck, 1981], have been widely replaced by deep convolutional neural networks with which more accurate results on demanding benchmarks like Sintel [Butler et al., 2012] or KITTI [Menze and Geiger, 2015] are achieved. These neural networks range from simple concatenated convolutional layers like FlowNet [Dosovitskiy et al., 2015; Ilg et al., 2017], to more complex structures such as PWC-Net, SpyNet, GMA and the Recurrent All-Pairs Field Transform (RAFT) [Sun et al., 2018; Ranjan and Black, 2017; Jiang et al., 2021; Teed and Deng, 2020], including 4D cost volumes, recurrent units and most recently also transformers in FlowFormer [Huang et al., 2022a].

One significant weakness of current state of the art neural networks, no matter the application, is their vanishing robustness against so called adversarial attacks [Cisse et al., 2017b; Moosavi-Dezfooli et al., 2017; Akhtar and Mian, 2018; Wu et al., 2020; Ranjan et al., 2019; Schmalfuss et al., 2022b]. These attacks seek a small, noisy perturbation of the input producing maximal change in the output, thus rendering the prediction futile. In the context of optical flow, the Perturbation-Constrained (Adversarial) Flow Attack (PCFA) of [Schmalfuss et al., 2022b] provides a potent attack. While closing the distance between the network's prediction and a pre-defined target flow, which is assumed to be sufficiently far away, the PCFA also constrains the perturbation below a certain threshold.

The distance thus reached is called the adversarial robustness and provides a lower bound to the Lipschitz constant of the network. Whereas computing the exact Lipschitz constant is NP-hard [Virmaux and Scaman, 2018], its upper bounds are known to be linked to robustness and generalization performance [Goodfellow et al., 2014; Szegedy et al., 2014; Yoshida and Miyato, 2017; Gouk et al., 2021; Cisse et al., 2017a].

To improve the adversarial robustness of the optical flow predicting neural network RAFT, this work adopts structured Receptive Field Convolutional Neural Networks (RFCNN) to enforce specific (geometric) priors into convolutional layers shown to enhance performance in image classification and inpainting [Jacobsen et al., 2016; Schmalfuss et al., 2022a]. This strategy uses clever spanning sets of filters called Filter Dictionaries (FD) which can contain low-pass, derivative and other motivated filters. Weighting these filters with trainable parameters retains the network's ability to learn from training data. By taking a linear combination of a random but fixed subset of a FD with only 3 filters, as proposed by [Schmalfuss et al., 2022a], the resulting kernels are interpretable filters with a motivated meaning, e.g. a mixture of derivative filters in different directions and orders. With several kernels to learn, it is assumed that the network is able to cover all possible kernels, since all filters are chosen sufficiently often. On the other hand, trimming the learnable parameters for each kernel to 3 also reduces the total number parameters significantly.

A first trial of this approach is tested in [Walter, 2022], which uses Parseval completed sparse directional FDs and places them in single convolutional layers of RAFT. In summary, both the quality on the test data and robustness suffers when placed in the first layers of the image encoders, but improves robustness in the motion encoder of the recurrent block.

Research Questions

Since sparse directional FDs are not the only choice for receptive fields, this raises the following questions. 1) Do other FDs, such as Gaussian derivative FDs, which more closely represent the smoother kernels of the trained baseline yield better robustness? 2) Thinking one step further, can a FD fitted to the freely trained filters of the applied layer improve robustness? 3) Can FDs without Parseval completion yield a better robustness? 4) In which positions is a FD most effectively placed?

To be precise, this work covers three main research questions:

Q.1 How much does the FD-choice matter in improving optical flow robustness?

- **Q.1.1** Next to Parseval completed sparse directional FDs, how do smooth Gaussian derivative FDs perform?
- **Q.1.2** Next to Parseval completed sparse directional FDs, how does a novel FD, which is tailored to freely trained filters, perform?

Q.2 Is the Parseval condition of FDs necessary for improved robustness or do plain dictionaries suffice?

Q.3 Which layers of the RAFT network yield the best robustness improvement when replaced with receptive fields?

Thesis Structure

This thesis starts in Chapter 2 with a more thorough introduction into the optical flow task, its prediction via variational methods and recent neural networks, including a detailed description of RAFT, and takes a closer look on the PCFA and possible defenses against adversarial attacks. The chapter then finishes with the principles of RFCNNs leading up to the author's project work [Walter, 2022]. To answer the first two Questions **Q.1** and **Q.2**, Chapter 3 explains how the mentioned FDs can be created and clarifies their individual motivations. It also provides an overview of the FDs' basic properties and their usage in this thesis for kernels with miscellaneous kernel sizes.

Their performance including the placement in the RAFT network, Question **Q.3**, is then examined in Chapter 4. The experiments are split into five parts, each concerning a different placement of the FDs. Starting with the baseline in Experiment 4.1, the FD placements into single layers, the complete feature encoder and update block are discussed in Experiments 4.2 to 4.4 respectively. Experiment 4.5 then attempts to replicate achieved improvements by regularizing the Lipschitz constant of specific layers using the method of [Gouk et al., 2021].

All combinations of FD choice and layer position (configurations) are evaluated by using the following three aspects: 1) quality on the trained Sintel train and test data, 2) generalization performance on the KITTI training split, without further finetuning, and 3) adversarial robustness against the PCFA on the Sintel test set. In addition to the lower Lipschitz bound of the whole network through the robustness, upper bounds of the convolutional layers' Lipschitz constants are investigated as well. Lastly, a short summary of this thesis and possible future topics are given in Chapter 5.

2 Related Work

Given all the previous studies which prompted this thesis, this chapter will give a short summary of all parts necessary for further understanding. It will start by introducing the concept of optical flow and its predictions methods. From plain variational methods to recent artificial neural networks these methods have become more complex and more accurate. However, neural networks lack stability against so called adversarial attacks and struggle to generalize properly [Sun et al., 2022; Schmalfuss et al., 2022b]. One such state of the art neural network, RAFT, can be interpreted as a self-learned variational method. RAFT forms the baseline onto which the Filter Dictionaries of Chapter 3 are applied.

The second part will explore how FDs help to improve neural networks by following the works of Jacobsen et al. [2016] and Schmalfuss et al. [2022a], while an in depth analysis of FDs and their creation is located in Chapter 3. Finally, this chapter will conclude with a recapitulation of the author's project work [Walter, 2022], a first trial testing the viability of FDs in the above mentioned neural network RAFT.

2.1 Predicting Optical Flow

In the field of computer vision, optical flow is the apparent 2D displacement of an object from one image to the next in a video or image sequence. An object located at pixel (x, y) in the first image $\mathcal{I}_1$ could be located at a potentially different pixel $(\tilde{x}, \tilde{y})$ in the second image $\mathcal{I}_2$, This could be a consequence of the object's and the camera's movement, thus, and to avoid unnecessary confusion, this thesis will always consider motion with respect to the camera's frame of reference.

The goal of optical flow estimation is to find a dense vector field $(u, v)^\top : \mathcal{I}_1 \to \mathbb{R}^2$, i.e. find a vector $(u(x, y), v(x, y))^\top$ for every $(x, y) \in \mathcal{I}_1$ which describes the object's movement direction and velocity to the second image. Measured in pixels, adding the optical flow to the original position should ideally result in the pixel

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} x + u(x, y) \\ y + v(x, y) \end{pmatrix} \tag{2.1}$$

of the object in $\mathcal{I}_2$. It is easy to see that this cannot hold everywhere. If, for example, the object moves out of frame or becomes occluded by another object, $(\tilde{x}, \tilde{y})$ does not exist. However

optical flow can still be defined to be the motion to the place where the object would be, if it was visible.

Between two images of the Sintel data set [Butler et al., 2012] the optical flow could look similar to Figure 2.1. Here, the resulting, scaled vector field is shown for a sub-sampled set of pixels. To visualize the optical flow at all pixels, the vectors are color encoded with the hue showing the direction and the saturation denoting the magnitude of the flow.

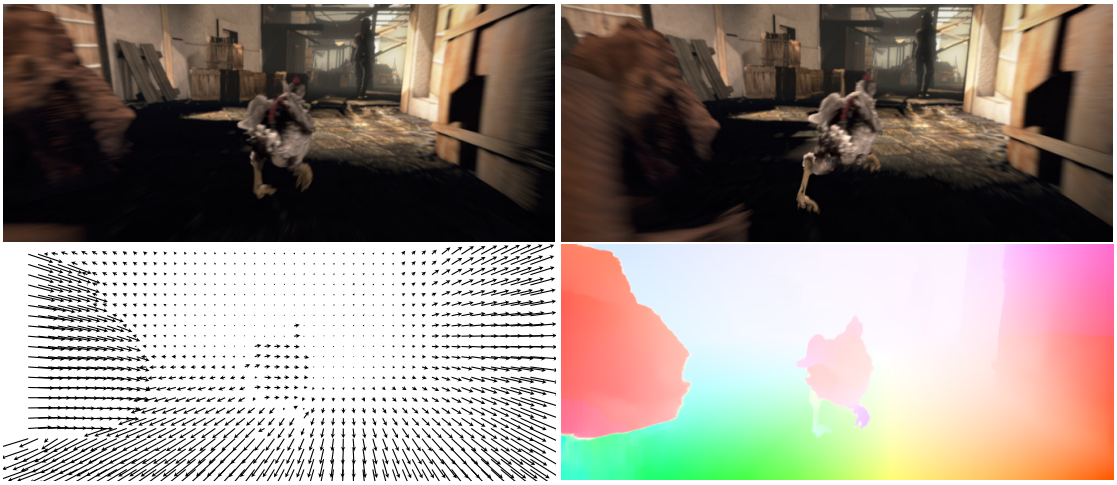


Figure 2.1: Two example images from the Sintel data set [Butler et al., 2012] and the respective optical flow as a vector field and color encoded image.

The challenge of optical flow prediction lies in accurately matching pixels from $\mathcal{I}_1$ to $\mathcal{I}_2$. A simple toy example are two images of a flat, white wall. Without further information it is impossible to determine whether the wall has or has not moved. Any non-white object like a picture frame or painting on the wall would change the situation considerably. With this 'feature' located in $\mathcal{I}_1$ one could look for the same constellation of pixels in $\mathcal{I}_2$. By extrapolating this motion onto the whole image one can infer that the wall has moved just the same. The definitions of such 'features' have changed over the years and led to several approaches discussed in the following section.

2.1.1 Variational Methods

Variational methods for optical flow predictions rely on the so called "brightness constancy assumption", "gradient constancy assumption" or others [Horn and Schunck, 1981; Nagel and Enkelmann, 1986; Barron et al., 1994; Brox et al., 2004; Zach et al., 2007; Zimmer et al., 2011; Fortun et al., 2015; Maurer et al., 2017]. In the case of the brightness constancy assumption, the same object is assumed to have equal brightness in both $\mathcal{I}_1$ and $\mathcal{I}_2$, i.e. for all $x, y \in \mathcal{I}_1$

$$\hat{\mathcal{I}}_1(x, y) = \hat{\mathcal{I}}_2(x + u(x, y), y + v(x, y)), \tag{2.2}$$

with $\hat{\mathcal{I}}$ being the gray value representation of $\mathcal{I}$. In other words, the flow should minimize the following data term

$$\mathcal{D}(\hat{\mathcal{I}}_1, \hat{\mathcal{I}}_2, u(x, y), v(x, y)) \coloneqq (\hat{\mathcal{I}}_2(x + u(x, y), y + v(x, y)) - \hat{\mathcal{I}}_1(x, y))^2. \tag{2.3}$$

Since Equation (2.2) has two unknowns, a unique solution cannot be determined. To solve this so called "aperture problem" [Bertero et al., 1988], one must add regularizing constraints, e.g. enforced spatial smoothness of the flow [Horn and Schunck, 1981]. These constraints are added to $\mathcal{D}(\cdot)$ as an additional regularizing term $\mathcal{R}(\cdot)$.

The optical flow problem can then be stated as a general minimization problem of the following energy

$$\begin{aligned} E(u(x, y), v(x, y); \alpha) = &\iint_{(x,y)\in\mathcal{I}_1} \mathcal{D}(\hat{\mathcal{I}}_1, \hat{\mathcal{I}}_2, u(x, y), v(x, y)) \\ &+ \alpha^2 \, \mathcal{R}(x, y, u(x, y), v(x, y), \nabla u(x, y), \nabla v(x, y), \ldots) \, dx \, dy, \end{aligned} \tag{2.4}$$

with regularization weight α.

The Method of Horn and Schunck

One such variational method is the method of [Horn and Schunck, 1981]. Assuming the brightness constancy assumption of Equation (2.3), a first order Taylor expansion of $\mathcal{I}_1(x, y)$ in both space and time and enforcing no sudden changes in the resulting vector field yields:

$$\begin{aligned} E_{HS}(u(x, y), v(x, y); \alpha) = &\iint_{(x,y)\in\mathcal{I}_1} (u(x, y)\partial_x \hat{\mathcal{I}}_1(x, y) + v(x, y)\partial_y \hat{\mathcal{I}}_1(x, y) + \partial_t \hat{\mathcal{I}}_1(x, y))^2 \\ &+ \alpha^2 (\|\nabla u(x, y)\|^2 + \|\nabla v(x, y)\|^2) \, dx \, dy, \end{aligned} \tag{2.5}$$

where $\partial_t \hat{\mathcal{I}}_1(x, y) \coloneqq \hat{\mathcal{I}}_2(x, y) - \hat{\mathcal{I}}_1(x, y)$. Here, the time difference is assumed to be normalized to 1.

In this case, a linear system of equations is obtained from the energy through optimization and discretization. While solving this system, the data term drives the solution towards a flow that

respects the brightness constancy term and is most influential where the image derivatives are large, i.e. the edges, corners of that picture frame from the example above. But in places on the images where all derivatives are close to 0, i.e. the white wall, the optical flow could, in theory, be chosen arbitrarily. In these areas the regularizing term takes over through enforcing few to no changes in the flow field. By extension, the optical flow in smooth areas of the image gets filled with the values coming from the informative 'features'.

More advanced variational approaches build upon the method of Horn and Schunck and derive more accurate and detailed forms of the data term as well as more complex regularizers [Nagel and Enkelmann, 1986; Brox et al., 2004; Zach et al., 2007; Werlberger et al., 2009; Zimmer et al., 2011; Maurer et al., 2017]. For example, Brox et al. [2004] add the gradient constancy assumption to the minimizing energy and Werlberger et al. [2009] explore anisotropic regularization techniques. These methods however yield improvements in accuracy, but also lead to large computation times [Bruhn et al., 2003].

2.1.2 Neural Networks and the RAFT Architecture

More recently, optical flow predictions using neural networks, like FlowNet [Dosovitskiy et al., 2015; Ilg et al., 2017], MaskFlownet [Zhao et al., 2020a], SpyNet [Ranjan and Black, 2017], PWC-Net [Sun et al., 2018], RAFT [Teed and Deng, 2020], GMA [Jiang et al., 2021] and Flowformer [Huang et al., 2022a] have emerged. Not only are they more accurate than variational methods but also have fast inferences on demanding benchmarks such as KITTI [Menze and Geiger, 2015] and Sintel [Butler et al., 2012].

It is important to note that for neural networks no assumptions and regularizing terms have to be constructed, they learn to recognize and interpret image 'features' just from training data. Being both a blessing and a curse, this leads to the risks of overfitting making generalization behavior a crucial part of neural network training [Dietterich, 1995; Sun et al., 2022].

Besides network architecture design, training procedures cannot be neglected [Sun et al., 2022] with data augmentation being an essential part. Another possibility for improving generalization power is to apply unsupervised learning procedures to given neural networks. Hence is is possible to extend the networks skills into the wild by training also on unlabeled data [Yu et al., 2016; Jonschkowski et al., 2020; Kong and Yang, 2022; Scheurer, 2022]. On labeled benchmarks however, unsupervised learning gives mixed results and generally lags behind supervised counterparts. As unsupervised learning applications are largely independent of the network in question it is practicable to compare the different approaches solely on their supervised performance.

Neural networks for optical flow predictions often draw inspiration from their theoretical counterparts, extend upon existing networks, or borrow successful techniques from other fields and are therefore carefully orchestrated architectures. The rest of this section is dedicated to paraphrase how networks can originate from theoretic principles, explain the Recurrent

All-Pairs Field Transform in detail which serves as the plain-vanilla network used in this thesis and finally touch how, in the recent past, natural language processing has influenced optical flow.

PCW-Net

PWC-Net for example is based upon a coarse-to-fine warping technique known from variational methods [Sun et al., 2018; Brox et al., 2004]. A sequence of sub-sampled images becoming more coarse at every level make up the so called spatial pyramid. Starting at the coarsest level, a flow estimate is computed which is then upsampled to be the initial guess of the finer prediction. Warping is the concept of incorporating this initial guess into the fixed part of the minimization equation, thus, only solving for the missing modification to this flow. Finally, the sum of the flow of all layers is the final prediction. PCWNet differs in only two aspects. It uses learned features in different resolutions to build the pyramid, instead of the images themselves, and replaces the numerical solver with a neural network at every level.

RAFT

One state of the art convolutional neural network (CNN) for optical flow prediction is the Recurrent All-Pairs Field Transform [Teed and Deng, 2020]. With the Networks architecture, seen in Figure 2.2, the authors imitate a variational method's iterative prediction process. The idea behind this neural network formulation is to allow for complex energies without needing to derive any equations. All necessary terms are automatically learned from the data, while uninformative terms are ignored. Due to the inherent Black-Box nature of neural networks, none of these terms can easily be verified to be the exact counterparts of an energy minimization [Shrikumar et al., 2017; Olden and Jackson, 2002; Adadi and Berrada, 2018; Gunning et al., 2019]. Nevertheless, the RAFT optical flow calculation can be split into three main parts: feature extraction, visual similarity computation and iterative updates.

Feature extraction is done through two convolutional networks. The feature encoder learns data-specific feature vectors for every pixel to be calculated for both images, while the context encoder is applied only to the first image. Both of these learned features could in theory encode anything ranging from pixel brightness, edges, corners, or more abstract elements like the aforementioned picture frame on the wall.

Visual similarity. Secondly, the visual similarity between both images is calculated as a 4D Correlation Volume of the two feature encoder outputs. Each entry holds the information of similarity between pixels $(x_1, y_1) \in \mathcal{I}_1$ and $(x_2, y_2) \in \mathcal{I}_2$ as the dot product of their feature vectors. This setup allows the network to extract the correlation between each pixel (x, y) and its corresponding point $(\tilde{x}, \tilde{y}) \coloneqq (x + u(x, y), y + v(x, y))$ with $(u, v)^\top$ being the current flow estimate, a quantity analogous to the data term $\mathcal{D}(\cdot)$ seen in variational approaches. The

difference being that the features are automatically fitted to the data instead of predefined, fixed properties such as the brightness constancy assumption.

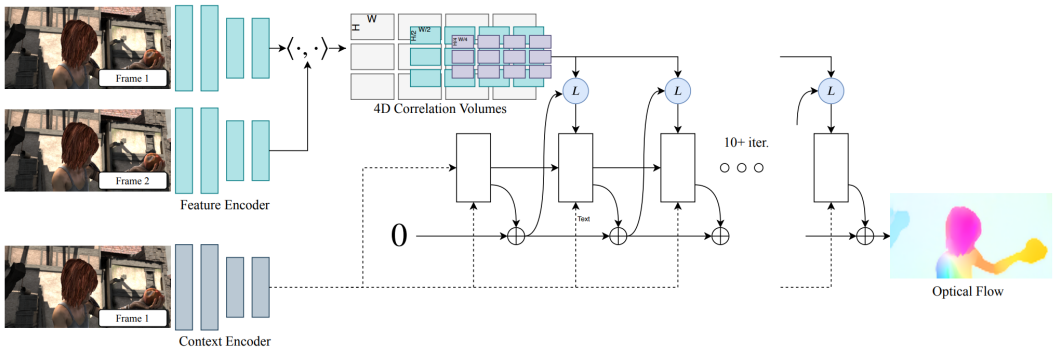
To later find flows with better similarity and to be computationally efficient, the correlation is read out only in a small neighborhood of $(\tilde{x}, \tilde{y})$. This procedure omits information about a possible improvement for large changes of the current flow estimate, e.g. bigger than the range of the neighborhood. Therefore, *Teed and Deng* construct a 4-layer correlation pyramid by pooling the volume in the latter two dimensions with kernel sizes $k = 1, 2, 4, 8$. Now, with the same number of look-ups in each pyramid layer a bigger, yet coarser, neighborhood can also be sampled without drastically increasing the computational effort.

Iterative updates. Finally, a recurrent neural network performs ten or more iterative updates to simulate a first-order optimization algorithm. Starting with an initial guess, e.g. zero-flow, the update block repeatedly computes flow updates ΔF. The quantity L can be interpreted as the discrete optimization objective $E_{\mathrm{RAFT}} = L(\theta; \mathcal{I}_1, \mathcal{I}_2, u(x, y), v(x, y))$. One part of L is the current flow estimate itself, while the other part is computed from a combined convolutional encoding of correlation and flow estimate and relates to the energy analogous data and regularization terms. The main driving force is then a Gated Recurrent Unit (GRU) feeding externally on the objective $L(\cdot)$ and the context encoder's features. Its hidden layer output is then passed through two more convolutions before becoming the current flow update $F_{t+1} = F_t + \Delta F_t, t \geq 0$. Note, the GRU in RAFT is based on convolutions instead of fully connected layers.

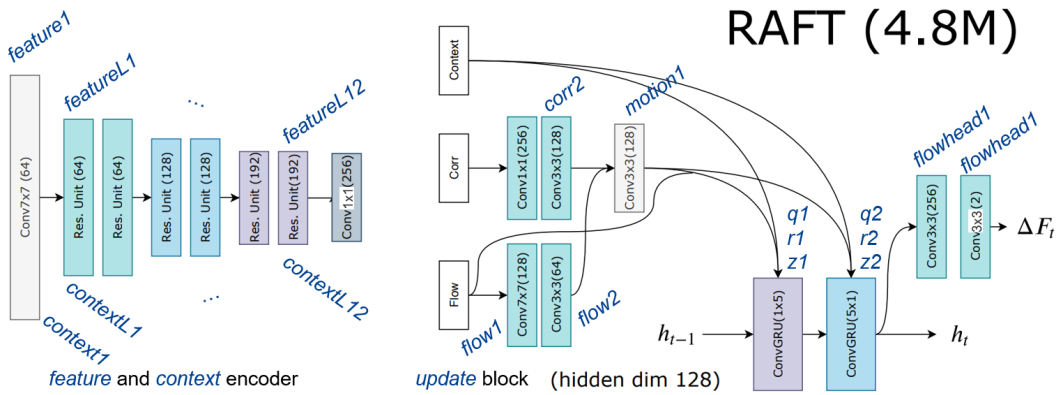
Layer-names. Further, it is crucial to this thesis to assign layer-names to all important convolutional layers, c.f. the detailed RAFT architecture in Figure 2.2b. Convolutions with kernel size of 1×1 are not explicitly named. The feature and *context* encoder consist of a 7×7 convolution *feature1* (*context1*) followed by twelve 3×3 Residual Units *featureL1-12* (*contextL1-12*). The update block starts with the motion encoder of the 7×7 *flow1* and 3×3 *flow2* flow estimate encoders, the 3×3 *corr2* correlation encoder and the 3×3 convolution *motion1*. The GRU block is split up into a 1×5 and a 5×1 GRU. Its three activations being *z1*, *r1*, *q1* and *z2*, *r2*, *q2* respectively. The final flow update layer consists of the two 3×3 convolutions *flowhead1* and *flowhead2*.

Flowformer

At the time of writing this thesis the forefront of current research has advanced one step further. A significant drawback of CNNs is their fixed and rather small kernel size, because only elements within that range can be connected to learn features. Attention based transformers are used in natural language processing [Vaswani et al., 2017] and just now have been applied to computer vision [Dosovitskiy et al., 2020; Huang et al., 2022a]. Due to their ability of modeling long-range relations they can be very effective for optical flow prediction. By exploiting this property Huang et al. [2022a] have developed the RAFT-based architecture FlowFormer which,



(a) The RAFT network



(b) All RAFT layers and its kernel sizes with indicated nomenclature (blue).

Figure 2.2: The principle of RAFT and its architecture. Both images originally taken from [Teed and Deng, 2020]. The kernel sizes of the last layers in Figure 2.2b of the encoders and the update block are adjusted to fit the implementation

broadly speaking, improves the motion encoder and the recurrent unit to directly extract and interpret information about the cost volume across the whole image. With this network they currently rank as one of the best published results on Sintel and also show strong cross-data set generalization.

2.1.3 PCFA and Adversarial Robustness

However, even state of the art neural networks are notorious for being vulnerable against adversarial attacks as various authors show. They intend to corrupt the models predictions as much as possible by changing the input, ideally only by a small amount [Szegedy et al., 2014; Moosavi-Dezfooli et al., 2017; Akhtar and Mian, 2018; Bhambri et al., 2019; Akhtar et al., 2021; Cisse et al., 2017b; Ranjan et al., 2019; Schrodi et al., 2022; Schmalfuss et al., 2022b].

Successfully applied in classification tasks, *targeted* attacks are a common form of attack. They search for a perturbation which is able to change the predicted label l to the target label $\check{l} \neq l$. This perturbation is usually minimized or its norm *constrained* below some threshold, as a small change constitutes a more effective attack. In addition, attacks can be performed on each image separately (*image/frame-specific*) or on a set of images yielding a single, *universal* perturbation which changes the predicted label for a large number of images. Next to these options, it is also possible to confine the *global* perturbations to a localized *patch* by only changing pixel values in a spatially confined domain. Patch attacks are particularly effective in the wild where a printed perturbation cannot fill the whole field of view [Huang et al., 2020; Wu et al., 2020; Ranjan et al., 2019].

PCFA

A strong attack specifically designed for adversarial flow is the global Perturbation-Constrained (Adversarial) Flow Attack by [Schmalfuss et al., 2022b]. It seeks the perturbation $\hat{\delta} = \delta_1, \delta_2$ for each image of the current time step which minimizes the distance $\mathcal{L}(\check{f}, f^t)$ between the perturbed flow $\check{f}$ and a predefined target flow f^t under the constraint that the perturbation remains within a certain bound $\varepsilon_2 > 0$. Thus, the PCFA solves the optimization problem

$$\underset{\hat{\delta}}{\operatorname{argmin}} \mathcal{L}(\check{f}, f^t) \text{ s.t. } \|\hat{\delta}\|_2 \leq \varepsilon_2 \sqrt{2I},\ \mathcal{I}_z + \delta_z \in [0,1]^I,\ z = 1,2, \tag{2.6}$$

where the additional factor $\sqrt{2I}$ makes the perturbation bound independent of the image size $I = M \cdot N \cdot C$. Typical values for the maximal average relative distortion ε_2 range between $0.05\% = 5 \cdot 10^{-4} \leq \varepsilon_2 \leq 5 \cdot 10^{-2} = 5\%$. Keeping the perturbed image $\mathcal{I}_z + \delta_z \in [0,1]^I$ within the allowed range is a non-trivial task and requires particular attention. Besides clipping the values on either end to 0 or 1, a change of variables instead optimizes the auxiliary variable ω_z defined by $\delta_z = \frac{1}{2}(\tanh(\omega_z) + 1) - \mathcal{I}_z$ through which the perturbed images never leave the desired domain.

The PCFA in this thesis is fixed to the change of variables with disjoint perturbations, i.e. δ_1 and δ_2 are separately optimized quantities. The target flow is chosen to be zero-flow $f^t = 0$ and its distance to the perturbed flow $\mathcal{L}(\check{f}, f^t)$ is measured in the average endpoint error $\mathrm{AEE}(\check{f}, f^t) = \frac{1}{I}\sum_{i \in I} \|\check{f}_i - f_i^t\|_2$. Defining the short hand $\hat{\varepsilon}_2 = \varepsilon_2 \sqrt{2I}$, squaring the constraint

on both sides $\|\hat{\delta}\|_2^2 \leq \hat{\varepsilon}_2^2$ and using a penalty method with Lagrangian multiplier $\mu \in \mathbb{R}$, the optimization problem now reads

$$\underset{\hat{\delta}}{\operatorname{argmin}} \frac{1}{I} \sum_{i \in I} \|\check{f}_i - f_i^t\|_2 + \mu \max(0, \|\hat{\delta}\|_2^2 - \hat{\varepsilon}_2^2),\ \mathcal{I}_z + \delta_z \in [0,1]^I,\ z = 1, 2, \tag{2.7}$$

which is then solved with the L-BFGS optimizer [Nocedal, 1980]. As one can confirm in figure 2.3, the images do not significantly change, but the RAFT predicted flow (almost) completely vanishes for $\varepsilon_2 = 5 \times 10^{-3} = 0.5\%$. With a weaker attack ($\varepsilon_2 = 1 \times 10^{-3} = 0.1\%$) the outline of the correct motion can still be recognized.

Adversarial Robustness

Adversarial attacks are directly linked to (adversarial) robustness by providing a lower bound to the model's Lipschitz constant Λ which controls the extend an input can change the output

$$\|f(x) - f(x+\delta)\|_p \leq \Lambda \|\delta\|_p,\ p > 1,\ \text{for any operation } f(\cdot) \text{ and inputs } x, x+\delta. \tag{2.8}$$

Upper bounds to the Lipschitz constant require analytic and architecture-specific considerations [Goodfellow et al., 2014; Szegedy et al., 2014] and computing the exact Lipschitz constant is NP-hard [Virmaux and Scaman, 2018]. Hence, adversarial attacks provide a suitable measure to quantify robustness for neural networks. The adversarial robustness is defined as

$$\mathrm{AEE}(\check{f}, f) \text{ s.t. } \|\hat{\delta}\| \leq \hat{\varepsilon}_2 \tag{2.9}$$

with f and $\check{f}$ being the initial and attacked flow. The lower the value the better the robustness, since, so assumed, the Lipschitz constant is smaller as well.

Defending against Adversarial Attacks

Defending against adversarial attacks can be done trough adversarial training or by controlling the Lipschitz constant of the model/network, not only making the models more robust but also improving generalization. Adversarial training can be seen as a form of data augmentation, since it directly incorporates adversarial samples into the training loss, e.g by averaging the loss of the clean and perturbed inputs [Goodfellow et al., 2014; Zhang and Wang, 2019].

Directly constraining the Lipschitz constant from above has been done through means of weight decay designed to minimize the squared largest singular value of the weight matrices [Yoshida and Miyato, 2017] and through rescaling said matrices to smaller Lipschitz constants via $W \leftarrow \frac{\lambda}{\max(\lambda, \|W\|_p)} W,\ p \in \{1, 2, \infty\},\ \lambda \in \mathbb{R}$ [Gouk et al., 2021]. They note that the actual choice of the Lipschitz bound λ for each layer is crucial to the performance and a lower constant does not always imply improvements.

On the other hand, [Cisse et al., 2017a] confine the optimization space of neural networks to *Parseval tight frame* weight matrices by adding an approximate projection step onto the *Stiefel manifold*

$$\mathcal{S} := \{W \in \mathbb{R}^{d_{out} \times d_{in}} : \ d_{out} \leq d_{in}, \ W^\top W = I_{d_{in} \times d_{in}}\}. \tag{2.10}$$

Here, d_{in} and d_{out} are the number of input and output dimensions and in the CNN case the respective number of channels, thus, giving the weights a largest singular value and Lipschitz constant of 1. The drawback is that the output dimension must necessarily be smaller than the input dimension which is not always desired.

(a) Unattacked prediction

(b) Attacked prediction $\epsilon_2 = 5 \cdot 10^{-3}$ and $\mu = 5 \cdot 10^5$

(c) Attacked prediction $\epsilon_2 = 1 \cdot 10^{-3}$ and $\mu = 1 \cdot 10^6$

Figure 2.3: The Perturbation-Constrained Adversarial Attack visualized. An unattached RAFT [Teed and Deng, 2020] prediction on the Sintel data set [Butler et al., 2012] (a). The same prediction with attacked images of different strength (b,c), i.e. ϵ_2 and μ.

2.2 Improvement of Neural Networks with Filter Dictionaries

RFCNNs have also been successfully used to enhance generalization performance on computer vision tasks by [Jacobsen et al., 2016; Safari et al., 2020; Schmalfuss et al., 2022a]. This thesis aims to transfer these achievements to the optical flow predicting network RAFT and improve its robustness against the aforementioned PCFA which it currently lacks [Schmalfuss et al., 2022b].

RFCNNs are a special class of convolutional neural networks, where the $n \times m$-dimensional convolutional kernels $K \in \mathbb{R}^{n \times m}$ are linear combinations of $d_{dict} \in \mathbb{N}$ filters from a predefined filter dictionary or filter bank

$$\{A\}_{0 \leq d < d_{dict}} \in \mathbb{R}^{d_{dict} \times n \times m}. \tag{2.11}$$

Hence, instead of training all kernel entries directly, the linear weights $\{\alpha\}_{0 \leq d < d_{dict}} \in \mathbb{R}^{d_{dict}}$ are now the trainable parameters and each kernel of a convolution reads

$$K = \sum_{d=0}^{d_{dict}-1} \alpha_d A_d. \tag{2.12}$$

Note that every conventional convolutional layer can be written as a RFCNN by simply letting $A_d = I_{d_{dict} \times n \cdot m}$, with I being the identity and $\mathbb{R}^{d_{dict} \times n \times m} \equiv \mathbb{R}^{d_{dict} \times n \cdot m}$. Therefore, a sparsity constraint is commonly implemented by using only a random, significantly smaller subset of the FD. A reasonable value is $d_{sparse} \coloneqq 3 < d_{dict}$. Most importantly, the subset is independently chosen for every kernel in the network with a side effect being the considerable reduction of parameters. Denoting a random but fixed permutation of the index set $\{d \in \mathbb{N}_0 \mid 0 \leq d < d_{dict}\}$ as $\pi(\cdot)$ the notation does not change significantly and the kernels become

$$K = \sum_{d=0}^{d_{sparse}-1} \alpha_d A_{\pi(d)}. \tag{2.13}$$

Furthermore, the learned kernels show enhanced interpretation ability over regular CNNs, since they are only a combination of but three appropriate filters, which makes the choice of the FD a very important task for RFCNNs.

Gaussian Derivative (GD) Filter Dictionaries

[Jacobsen et al., 2016] first introduce RFCNNs with Gaussian derivative filter dictionaries to regularize kernels for the extraction of smooth features up to fourth order. Results show improvements on classification tasks in computer vision, especially when data are scarce.

By interpreting images as functions in scale space they motivate using a basis of Gaussian derivatives $G^k(x, y, \sigma)$ to combine image derivatives of the scale space image instead of pixels

like in regular CNNs. The scale space of an image $\mathcal{J}$ is defined via the convolution of the original image $\mathcal{I}$ by a Gaussian kernel $G(x, y, \sigma)$ with scale σ^2

$$\mathcal{J}(x, y; \sigma^2) = \mathcal{I}(x, y) * G(x, y; \sigma). \tag{2.14}$$

Since $\mathcal{J}$ is infinitely differentiable it can be written as the Taylor expansion around a

$$\begin{aligned} \mathcal{J}(x; \sigma^2) &= \sum_{m=0}^{\infty} \frac{\mathcal{J}^k(a; \sigma^2)}{k!}(x-a)^k \\ &= \sum_{m=0}^{\infty} \frac{(\mathcal{I}(\cdot) * G(\cdot; \sigma))^k(a)}{k!}(x-a)^k \\ &= \sum_{m=0}^{\infty} \frac{(\mathcal{I}(\cdot) * G^k(\cdot; \sigma))(a)}{k!}(x-a)^k, \end{aligned} \tag{2.15}$$

where for the sake of argument $\mathcal{I}$ and by extension $\mathcal{J}$ are one-dimensional. The local geometry of the scale space image can be obtained by convolution with GDs. Since each GD fixes one independent degree of freedom, all GDs form a minimal, complete set. Thus, linearly combining Gaussian derivatives is the functional equivalent of weighting individual pixels in a standard CNN. One main advantage is that image derivatives in terms of pixels must not be learned by the network.

The GDs can be constructed via point wise multiplication with the orthogonal Hermite polynomials $H_m(\cdot)$ [Romeny, 2008], that is

$$G^k(x; \sigma) = G(x; \sigma) \circ \frac{(-1)^k}{\sqrt{\sigma}^k} \cdot H_m(\frac{x}{\sigma\sqrt{2}}). \tag{2.16}$$

In approximations of $\mathcal{J}$, GDs with orders above $k > 4$ are considered to not carry any more meaningful information to visual perception [Koenderink and Van Doorn, 1987]. Hence, they may be dismissed leaving the family $\{G^k(x; \sigma)\}_{k=0,1,2,3,4}$. In two dimensions the basis would include derivatives along both x and y axes, i.e. $G, G^x, G^y, G^{xx}, G^{xy}, G^{yy}, \ldots$, from which all other directions can be obtained through careful linear combinations [Freeman et al., 1991]. More details are given in Section 3.1.

Sparse Directional (SD) Filter Dictionaries

Another way of prescribing derivatives into a CNNs parameter space are sparse directional filter dictionaries proposed by [Safari et al., 2020]. Next to a low pass filter, e.g. a Gaussian $G(x; \sigma)$, the SD-FD contains finite difference derivative kernels of first and second order in all discrete directions. Finite difference kernels produce derivatives of images like their GD counterparts, but are also sparse and have compact support.

SD Parseval Extended Filter Dictionary Frames (SDp)

[Safari et al., 2020] and [Schmalfuss et al., 2022a] use SD Parseval extended dictionary frames to ensure the completeness of their basis and outperform regular CNNs when trained on few data while being competitive on large, extensive data sets. A SDp-FD induces a Parseval wavelet frame (framelet) in the space of square integrable functions $L^2(\mathbb{R}^s)$ ($s = 2$ for two-dimensional images) [Atreas et al., 2019]. In other words, the filters of the FD A have wavelet-like properties while also incorporating custom-selected filters. Details on this completion process is given in Section 3.4. This procedure ensures that the neural network not only extracts important features, like derivatives, but also a representation holding additional information for a full reconstruction of the original image. It is further indicated that these FDs lower the average spectral norm $\operatorname{avg}(\|K_{i,j}\|_2)$ of kernels in their layer which is closely connected to a network's generalization behavior [Safari et al., 2020; Long and Sedghi, 2019].

Though related, this concept must not be confused with the Parseval Networks in Section 2.1.3, since neither the dictionary $A \notin \mathcal{S} \Leftrightarrow A^\top A \neq I_{d_{dict} \times n \cdot m}$ nor the complete convolution operator $W = \{K_{i,0}, K_{i,1}, \dots, K_{i,d_{in}}\}_{0 \leq d < d_{out}} \notin \mathcal{S} \subset \mathbb{R}^{d_{out} \times n \cdot m \cdot d_{in}}$ are Parseval tight frames in $\mathbb{R}^{n \cdot m}$, i.e. in the Stiefel-manifold of Equation (2.10). Here, the kernels $K_{i,j}$ are subscripted by their corresponding input and output channels numbering $d_{in}, d_{out} \in \mathbb{N}$.

A First Improvement of RAFT using Filter Dictionaries.

Spreading SDp filter dictionaries to optical flow prediction, the author, [Walter, 2022], presented a first improvement of RAFT using filter dictionaries. Contrary to [Schmalfuss et al., 2022a], placing the receptive fields into the first layers of the feature and context encoders (*feature1*, *context1*) lowers both prediction quality and adversarial robustness. In the baseline RAFT these kernels already adequately show a smooth derivative behavior. *Flow1* in the motion encoder is another layer which is expected to extract derivative based features directly from the current flow estimate, but it is one that does not show derivatives as clearly as the first encoders. Here, a RFCNN layer does make RAFT more robust.

From said project work, the three still uninvestigated Questions **Q.1** to **Q.3** emerge and are now covered the following chapters. The first question aims at discerning different groups of FDs: GD-FDs, SD-FDs, and a novel FD which uses the freely trained kernels to impose frequently learned filters on the network. Secondly, this thesis also applies the Parseval completed FDs of the SD-FD and the novel FD to detect whether this additional property leads to performance improvements in Optical flow like it does in blind image inpainting [Schmalfuss et al., 2022a]. Since the receptive field is not located in the front of the network in the only beneficial configuration of the project work, the third and last question addresses the impact of different positions on the networks quality and robustness measures.

3 About Various Filter Dictionaries

As mentioned in the previous Section 2.2, the choice of a FD plays a vital role for RFCNNs. This chapter will analyze the three main groups of FDs which are inserted into RAFT in Chapter 4 to then answer research Questions **Q.1** and **Q.2**. Namely, it will list how GD- and SD-FDs can be created and how to obtain a Parseval frame completion of existing FDs. Since Question **Q.1.2** asks for a FD fitted to freely trained filters, Section 3.3 introduces the **principal component analysis filter dictionary** by taking each principal component of a set of trained filters as one member of the whole dictionary family. Lastly, Section 3.5 will provide an overview of all FDs used in Chapter 4.

3.1 Gaussian Derivative Filter Dictionaries

A closed formula for computing Gaussian derivatives $G^k(x, y; \sigma)$ of scale σ^2 for GD-FDs can also be derived in two dimensions by extending Equation (2.16) [Jacobsen et al., 2016]. The resulting equations become more complex, but the underlying principle stays the same. The now two-dimensional Gaussian has to be multiplied point wise by the respective Hermine polynomial $H_{\cdot}(\cdot)$ in x direction for every row and in y-directional polynomial for every column. This procedure can be written with the outer product $\otimes$ as

$$\frac{\partial^{k_x+k_y}}{\partial x^{k_x}\partial y^{k_y}} G(x, y; \sigma) = G(x, y; \sigma) \circ \frac{(-1)^{k_x+k_y}}{\sqrt{\sigma_x}^{k_x}\sqrt{\sigma_y}^{k_y}} \cdot \left[H_{k_x}(\frac{x}{\sigma_x\sqrt{2}}) \otimes H_{k_y}(\frac{y}{\sigma_y\sqrt{2}}) \right], \tag{3.1}$$

where the k_x-th derivative in x direction and k_y-th derivative in y direction of the Gaussian is taken.

Further, it can be shown that the k^{th} order Gaussian derivative in any direction is a linear combination of a minimal set of all whole number derivatives with $k_x + k_y = k$ [Freeman et al., 1991]. It is reasonable to ignore higher derivatives, since GDs of order $k > 4$ are considered to not carry any more meaningful information to visual perception [Koenderink and Van Doorn, 1987].

GD-FD up to Order 4 and Scale $\sigma = 1$ (GD4)

Hence, for an isotropic GD-FD up to order 4 and scale 1 ($\sigma \coloneqq \sigma_x = \sigma_y = 1$) it is sufficient to create the 15 kernels

$$G; G^x, G^y; G^{xx}, G^{xy}, G^{yy}; G^{xxx}, G^{xxy}, G^{xyy}, G^{yyy}; G^{xxxx}, G^{xxxy}, G^{xxyy}, G^{xyyy}, G^{yyyy},$$

where the arguments have been omitted and each x, y denotes one derivative in its direction, cf. Figure 3.1 for a visual representation. In general, a GD-FD of order k has $d_{dict} = \frac{(k+1)(k+2)}{2}$ filters each representing one independent degree of freedom.

To employ Gaussian derivatives in a neural network setting, one merely has to evaluate these filters on a $n \times m$ regular grid $\{(x, y) \in \mathbb{Z} \mid -\lfloor \frac{n}{2} \rfloor \leq x \leq \lfloor \frac{n}{2} \rfloor, -\lfloor \frac{m}{2} \rfloor \leq y \leq \lfloor \frac{m}{2} \rfloor\}$. When used in the network, this automatically scales the effective range down to pixel size, making the parameter σ invariant to the image resolution. Also note that d_{dict} is independent of the grid size. Thus, for $n, m = 3$, meaning 9 degrees of freedom, a GD-FD up to order 3 would theoretically contain enough filters ($d_{dict} = 10$) to span the space $\mathbb{R}^{n \times m}$ of all 3×3 filters. In practice this is not always true, because on a discrete grid the infinite support and continuity of the GDs is neglected and filters may contain the same information.

This property can be checked by computing the right-hand invertibility of the dictionary in matrix representation $A \in \mathbb{R}^{d_{dict} \times n \cdot m}$. If A is right-hand invertible, any kernel $K \in \mathbb{R}^{n \cdot m}$ can be written as the linear combination of the dictionary $K = \alpha A$ (Equation (2.12) in matrix notation) with weights $\alpha = KA^{-1} \in \mathbb{R}^{d_{dict}}$. In other words, A spans $\mathbb{R}^{n \cdot m} \equiv \mathbb{R}^{n \times m}$. By checking this criterion numerically it turns out that GD4 spans $\mathbb{R}^{3 \times 3}, \mathbb{R}^{1 \times 5}$ and $\mathbb{R}^{5 \times 1}$, and thus suffices to be used for filters of these sizes.

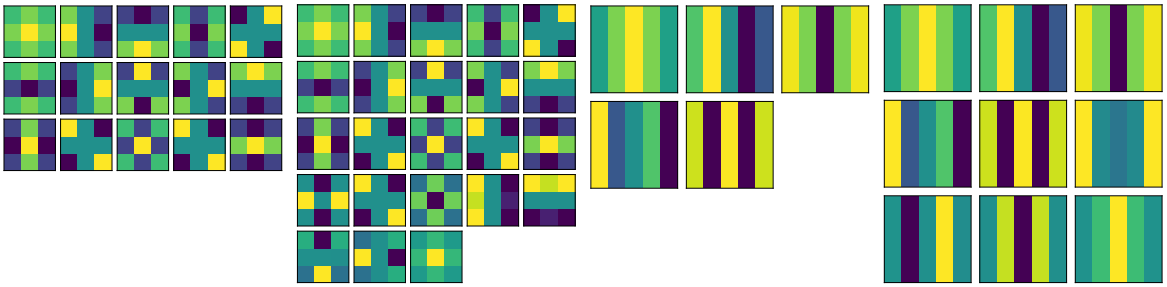
GD-FD up to Order 9 and Scale $\sigma = 1$ (GD9)

However, GD4 is not sufficient for $n, m = 7$ for which two alternatives are explored, cf. Figure 3.2. The first option is to use additional derivatives up to order 9, giving GD9 with $d_{dict} = 55$. Although being invertible, the inversion process would be numerically unstable, since A does not have full rank[1], i.e. $\mathrm{rank}(A) = 48 < 49 = 7 \cdot 7$. The space of kernels which are numerically hard to represent is shown in Figure 3.2b. Therefore, GD9 is used for 7×7 kernels only in Experiment 4.2.

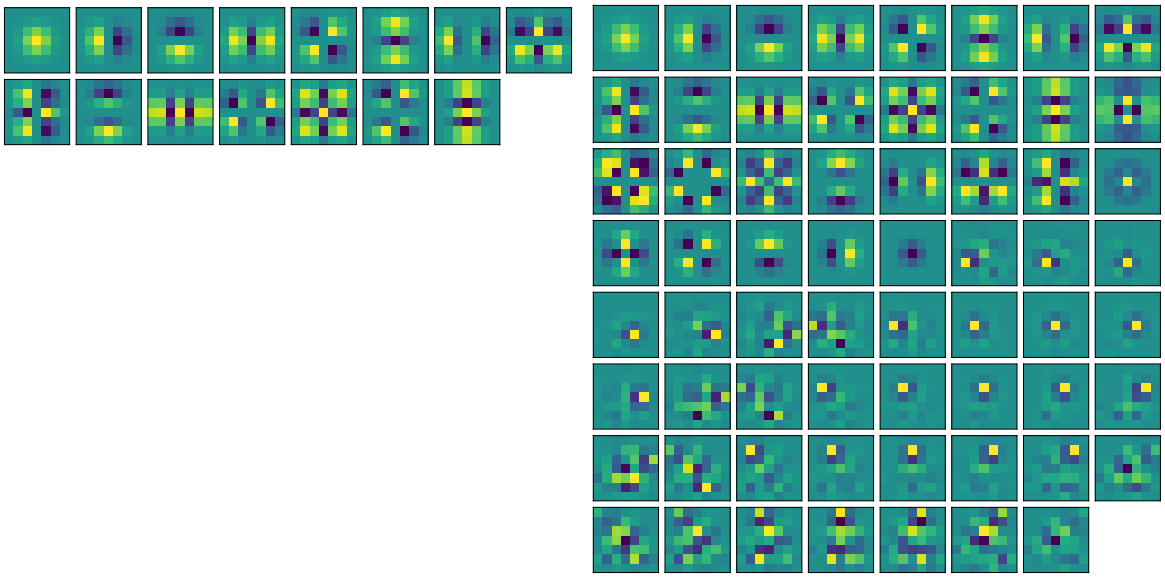
[1] The smallest eigenvalue is below the default tolerance of *numpy.linalg.matrix_rank()*: $2.2267e-14 = \sigma_{min} < 55 \cdot \sigma_{max} \cdot \epsilon = 5.8864e-12$, where ϵ is the double precision accuracy. https://numpy.org/doc/stable/reference/generated/numpy.linalg.matrix_rank.html, Accessed: 2023-01-18

GD4-FD with Four Scales $\sigma = 1, 2, 4, 8$ (GD4s) and $\sigma = 0.5, 1, 2, 4$ (GD4s05)

Another possibility is to use multiple GD4-FDs with varying scales, shown in Figure 3.2. 4 GD4-FDs are needed to complete the FD, giving $d_{dict} = 60 \geq 49$. Using $\sigma = 1, 2, 4, 8$ and $\sigma = 0.5, 1, 2, 4$ results in the FDs named GD4s and GD4s05 respectively. Not only are they right-hand invertible, but also incorporate different scales, giving information after different degrees of smoothing. In comparison GD4s05 tends towards sharper images, while GD4s represents more blurry scales. Both GD4s and Gd4s05 can be seen as the successors to GD9 and are is used in the subsequent Experiments 4.3 to 4.5.

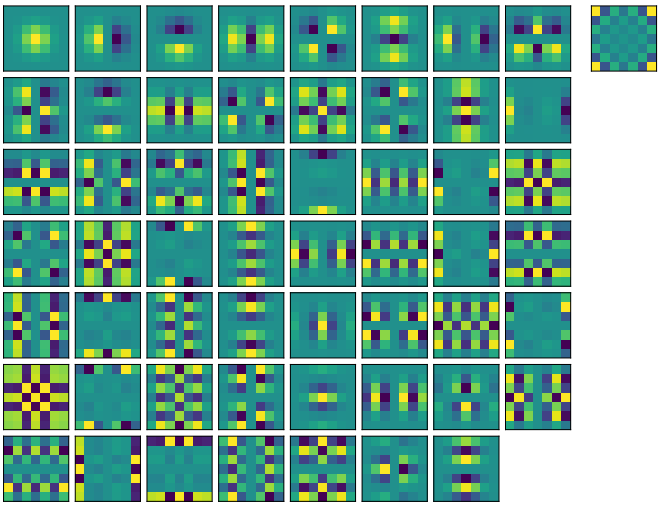


(a) GD4 3x3 **(b)** GD4 Parseval 3x3 **(c)** GD4 1x5 **(d)** GD4 Parseval 1x5



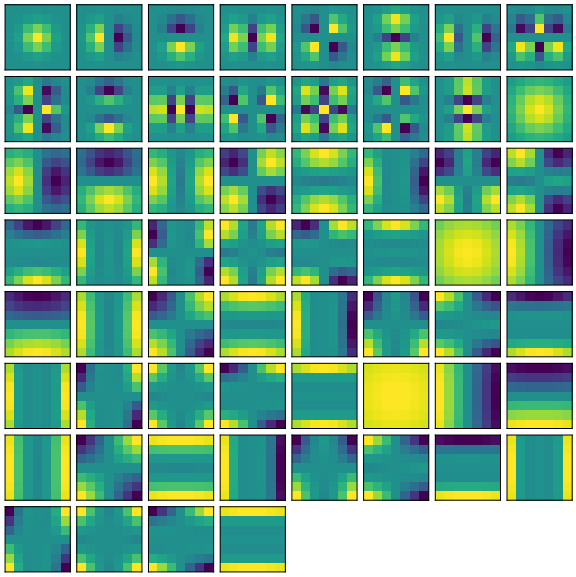
(e) GD4 7x7 **(f)** GD4 Parseval 7x7

Figure 3.1: Gaussian derivatives filter dictionaries up to order 4. 5×1 FDs are transposed versions of the 1×5 FD.

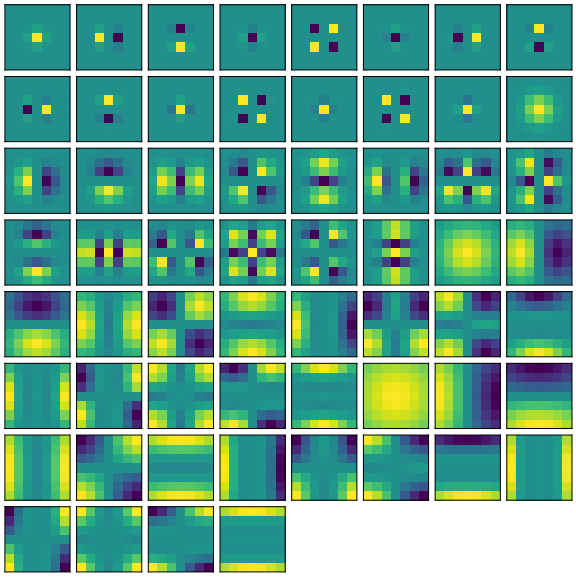


(a) GD9 7x7

(b) Numerically unrepresented kernel of GD9 7x7



(c) GD4s 7x7

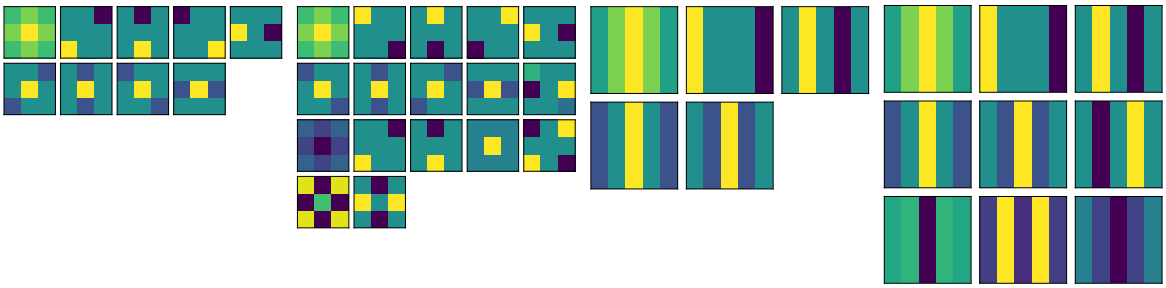


(d) GD4s05 7x7

Figure 3.2: Gaussian derivatives filter dictionaries up to order 9 (top left) and its numerically not represented filter (top right) and order 4 with different $\sigma = 1, 2, 4, 8$ (bottom left) and $\sigma = 0.5, 1, 2, 4$ (bottom right).

3.2 Sparse Directional Filter Dictionaries

Sparse directional FDs (SD) consist, as the name suggests, of a family of d_{dict} (two-dimensional) sparse filters $\{A\}_{1 \leq d < d_{dict}} \in \mathbb{R}^{n \times m}$ and one single low pass filter $A_0 \in \mathbb{R}^{n \times m}$. Here, the standard Gaussian $A_0 = G(x, y, 1)$ is chosen. The other filters are finite difference kernels in all possible discrete directions up to first and second order with 2 and 3 constant, non-zero entries respectively, cf. Figure 3.3. It is easy to verify that in this case all $d_{dict} = n \cdot m$ kernels form an orthogonal basis of $\mathbb{R}^{n \times m}$. Thus, every filter can be represented using a linear combination of SD filters.

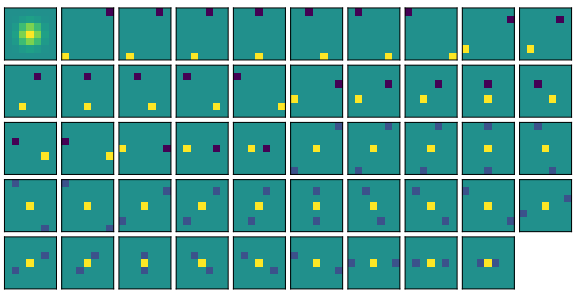


(a) Sparse directional 3×3

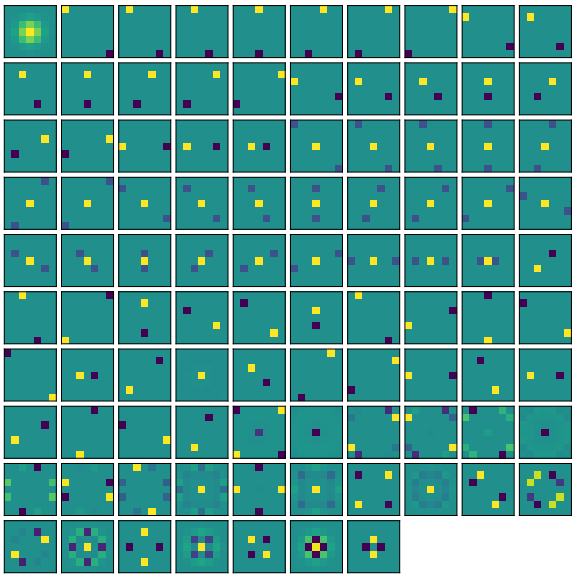
(b) Sparse directional Parseval 3×3

(c) Sparse directional 1×5

(d) Sparse directional Parseval 1×5



(e) Sparse directional 7×7



(f) Sparse directional Parseval 7×7

Figure 3.3: Sparse directional filter dictionaries. 5×1 FDs are transposed versions of the 1×5 FD.

3.3 Principal Component Analysis Filter Dictionaries

GD- and SD-FDs are predefined to be either smooth or sparse which might not be the perfect FDs for a task. Principal component analysis FDs try to solve this issue on a convolutional layer basis by taking learned filters from a baseline network into account. For this thesis the baseline network is the RAFT network trained in Experiment 4.1.

Principal Component Analysis FD (PCA)

The space of all d_{data} learned $n \times m$ filters in one particular layer, the data, can be seen as a point cloud $F \in \mathbb{R}^{d_{data} \times n \cdot m}$. By assuming that the data shows non-zero variance in all directions, its singular value decomposition has full rank:

$$F = U \Sigma V^*. \tag{3.2}$$

All orthogonal principal components $A = V^*$ will provide the PCA dictionary of size $d_{dict} = n \cdot m$ and thus span $\mathbb{R}^{n \times m}$. Each diagonal entry $\sigma_d = \Sigma_{d,d}, 0 \leq d < d_{dict}$ determines the explained variance of F as σ_d^2 in direction A_d. Thus, the sparsity constraint of Equation (2.13) forces the network to also learn filters in other directions than it has predominately learned. Figures 3.4 to 3.8 show the PCA dictionaries for each convolutional layer in the RAFT network.

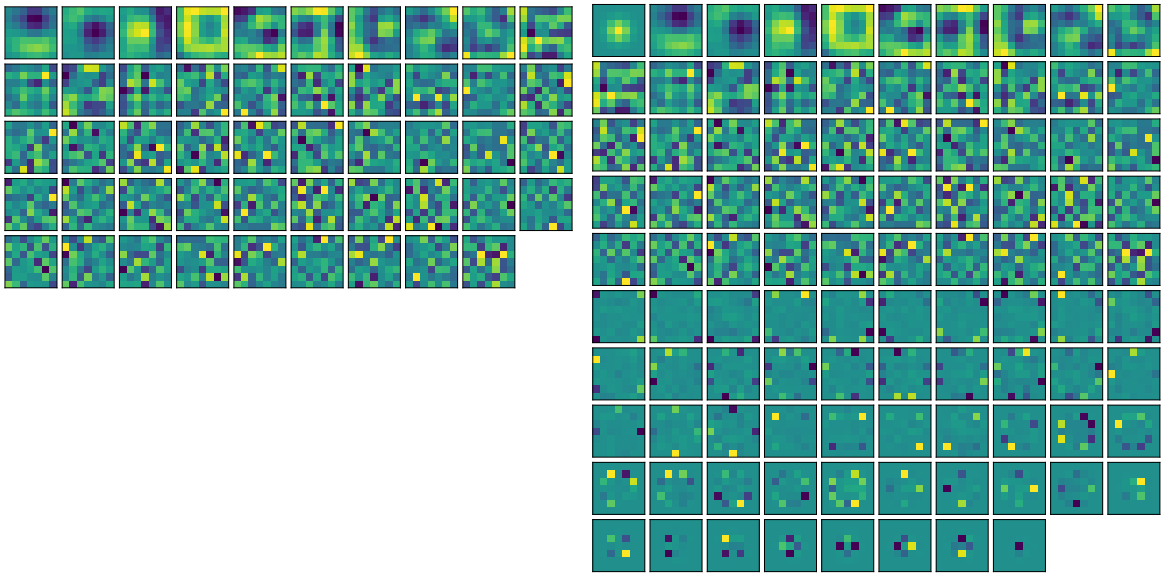
Truncated PCA-FD (PCAx)

Instead of using all components, it might be beneficial to focus on some principal components explaining the majority of the data's variability. A simple approach is to omit filters which explain less variance than a given threshold. Experiment 4.2 uses this approach for the RAFT layers *feature1* and *flow1* generating the FDs PCA14 and PCA24 respectively, cf. Figures 3.4a, 3.7a and 3.9. The first consisting of the 14 principal components with their explained variance above $\mathrm{PCA}14 = \{A_d \mid \sigma_d^2 > (0.5)^2, 0 \leq d < 49\}$ in the *feature1* layer, where the spectrum lies in $\sigma_d \in (0.19, 4.78)$. The latter consists of the 24 components above $\mathrm{PCA}24 = \{A_d \mid \sigma_d^2 > (0.87)^2, 0 \leq d < 49\}$ in the *flow1* layer, where the spectrum lies in $\sigma_d \in (0.09, 3.16)$. Due to the rather arbitrary threshold, PCA14 and PCA24 are only used in Experiment 4.2.

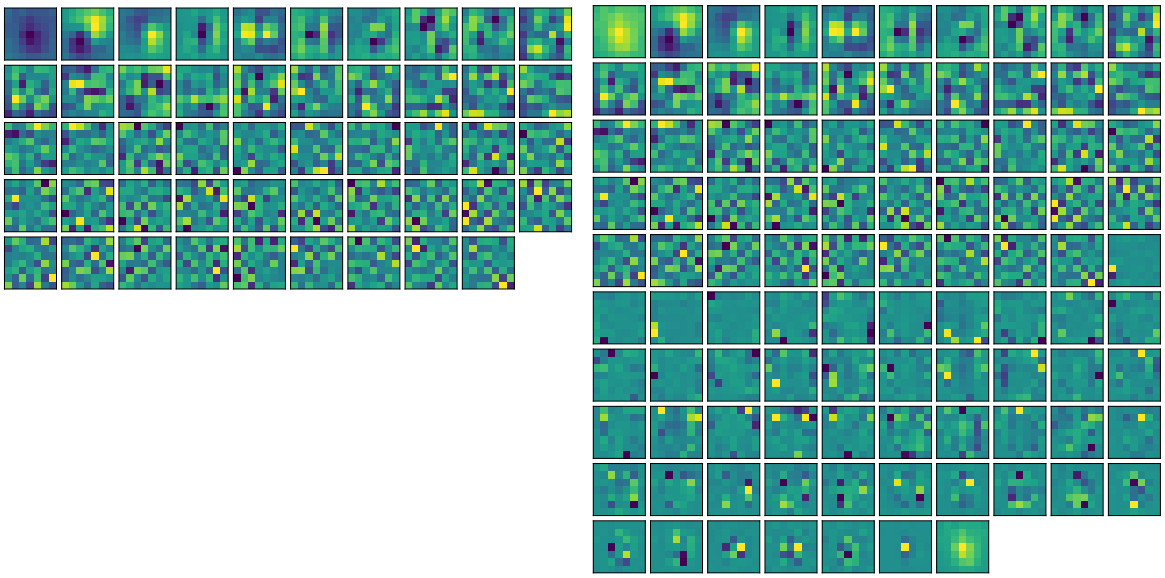
Data Related PCA-FD (PCArel)

A more universal method is to adjust the frequency for which each component is chosen to be in the sparsity constraint enforced subset of filters. That is which subset $\pi(\{d \in \mathbb{N}_0 \mid 0 \leq d < d_{sparse}\})$ spans the kernel $K = \sum_{d=1}^{d_{sparse}} \alpha_d A_{\pi(d)}$. Drawing the basis filters according to the discrete probability density function defined by the normalized spectrum of the

explained standard deviation $\{\sigma_d\}_{0 \leq d < d_{dict}}$, ensures that the components with high contributions are chosen more often. This FD is named PCArel as is chooses the basis filters in relation to their significance in the data. It replaces PCA14 and PCA24 in the Experiments 4.3 to 4.5.

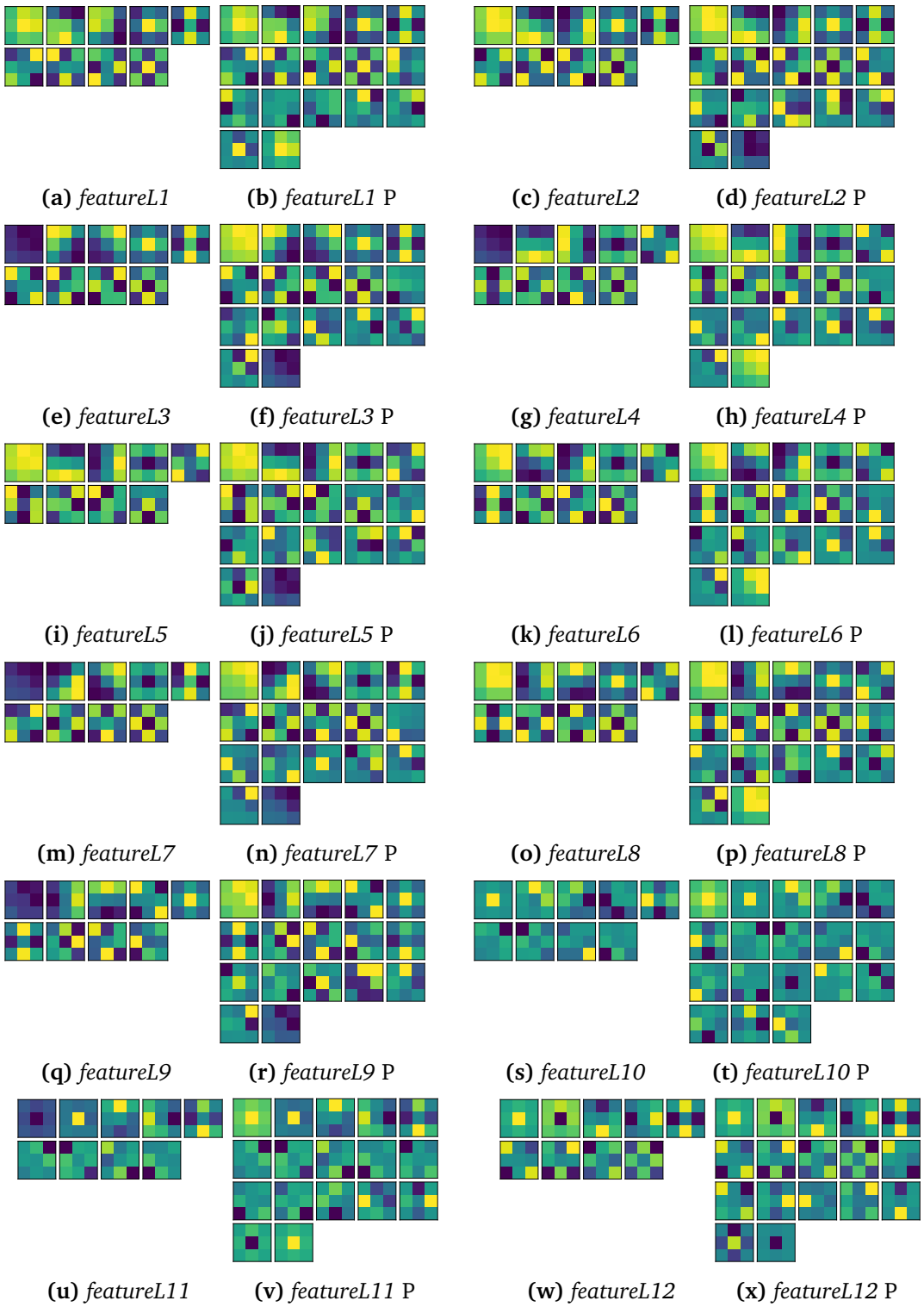


(a) *feature1* **(b)** *feature1* Parseval



(c) *context1* **(d)** *context1* Parseval

Figure 3.4: PCA filter dictionaries in the first layers of the image encoders.



(a) *featureL1* **(b)** *featureL1* P **(c)** *featureL2* **(d)** *featureL2* P

(e) *featureL3* **(f)** *featureL3* P **(g)** *featureL4* **(h)** *featureL4* P

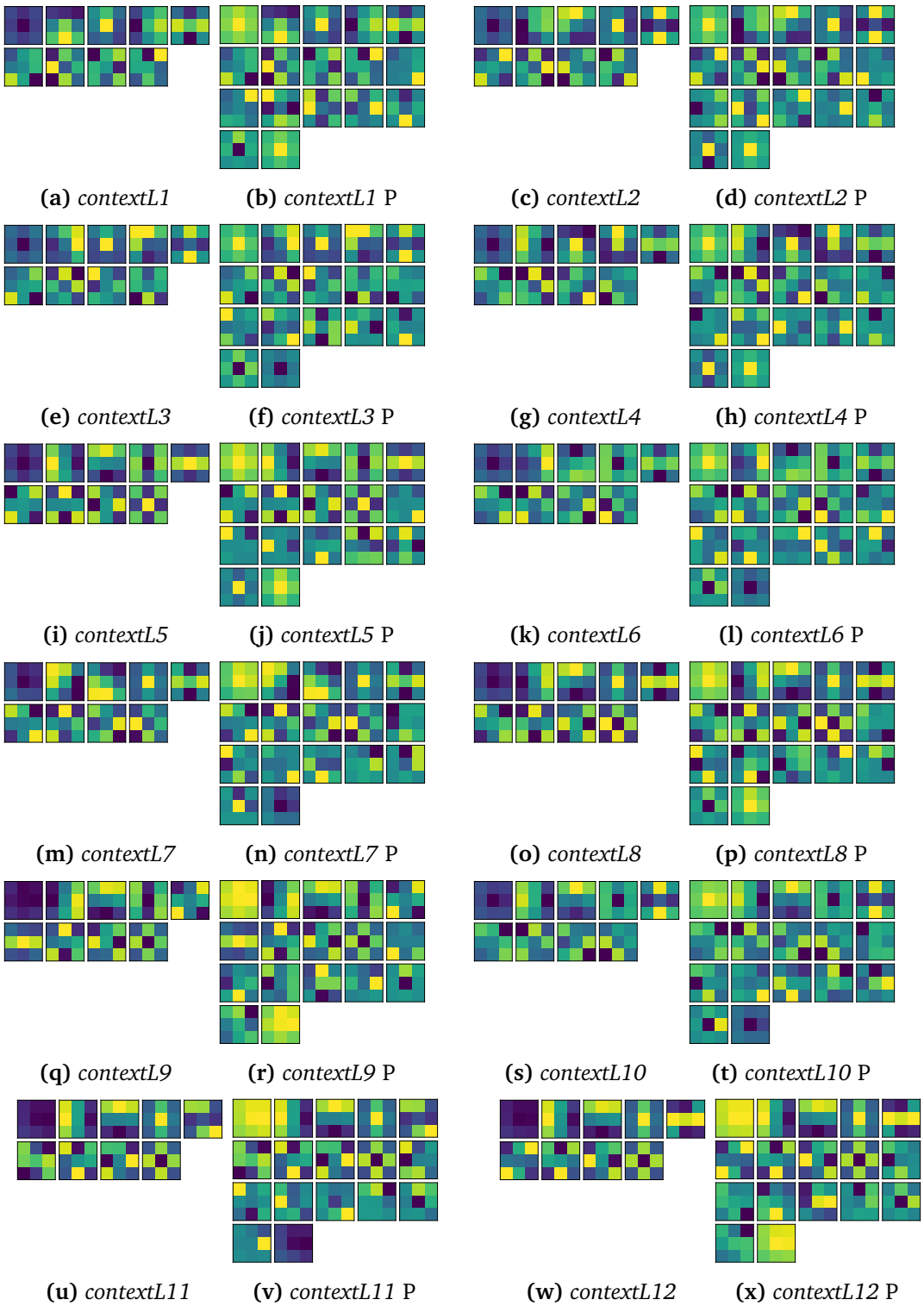
(i) *featureL5* **(j)** *featureL5* P **(k)** *featureL6* **(l)** *featureL6* P

(m) *featureL7* **(n)** *featureL7* P **(o)** *featureL8* **(p)** *featureL8* P

(q) *featureL9* **(r)** *featureL9* P **(s)** *featureL10* **(t)** *featureL10* P

(u) *featureL11* **(v)** *featureL11* P **(w)** *featureL12* **(x)** *featureL12* P

Figure 3.5: PCA filter dictionaries in the feature encoder residual layers. Parseval completed dictionaries are abbreviated with P.



(a) *contextL1* **(b)** *contextL1* P **(c)** *contextL2* **(d)** *contextL2* P

(e) *contextL3* **(f)** *contextL3* P **(g)** *contextL4* **(h)** *contextL4* P

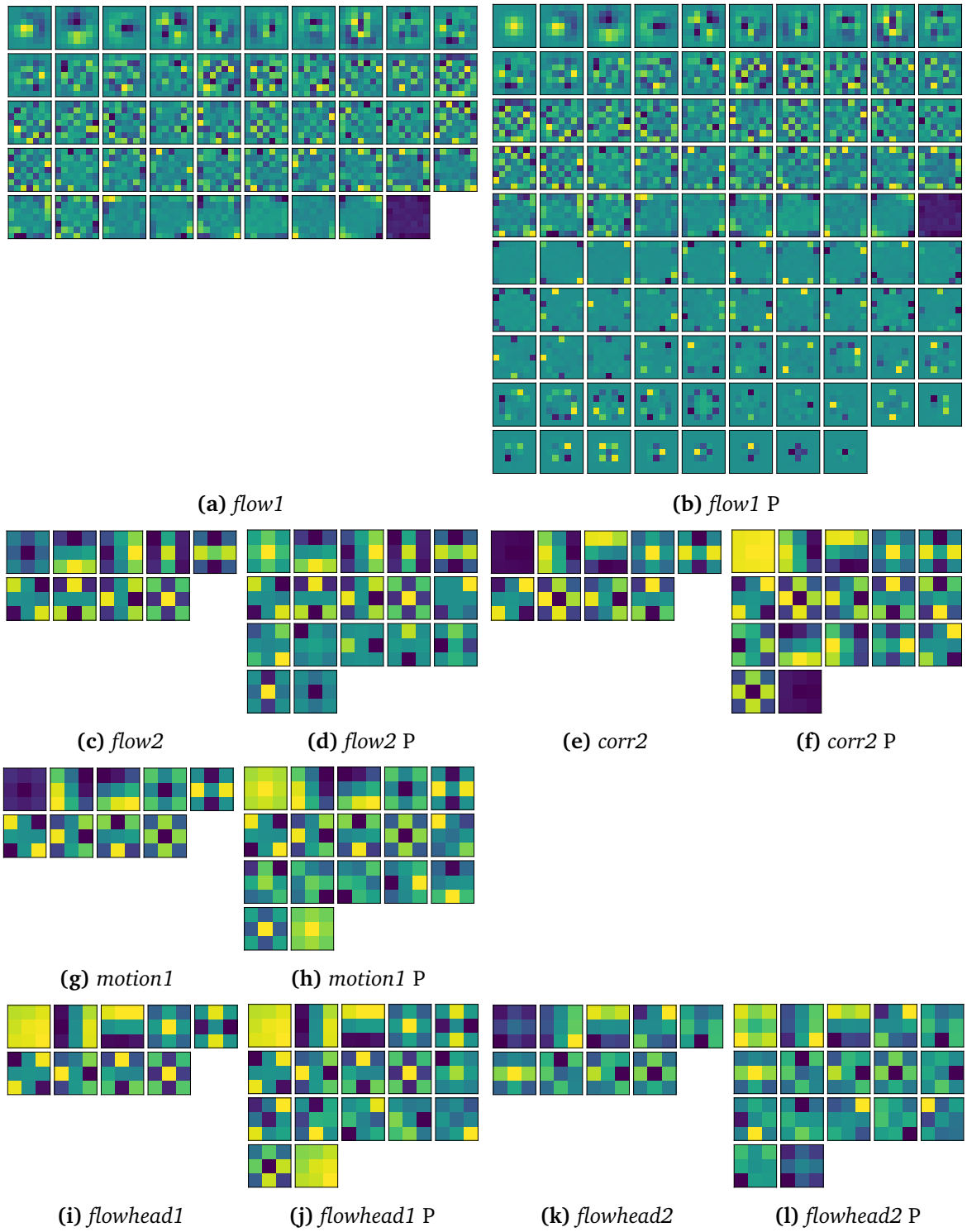
(i) *contextL5* **(j)** *contextL5* P **(k)** *contextL6* **(l)** *contextL6* P

(m) *contextL7* **(n)** *contextL7* P **(o)** *contextL8* **(p)** *contextL8* P

(q) *contextL9* **(r)** *contextL9* P **(s)** *contextL10* **(t)** *contextL10* P

(u) *contextL11* **(v)** *contextL11* P **(w)** *contextL12* **(x)** *contextL12* P

Figure 3.6: PCA filter dictionaries in the context encoder residual layers. Parseval completed dictionaries are abbreviated with P.



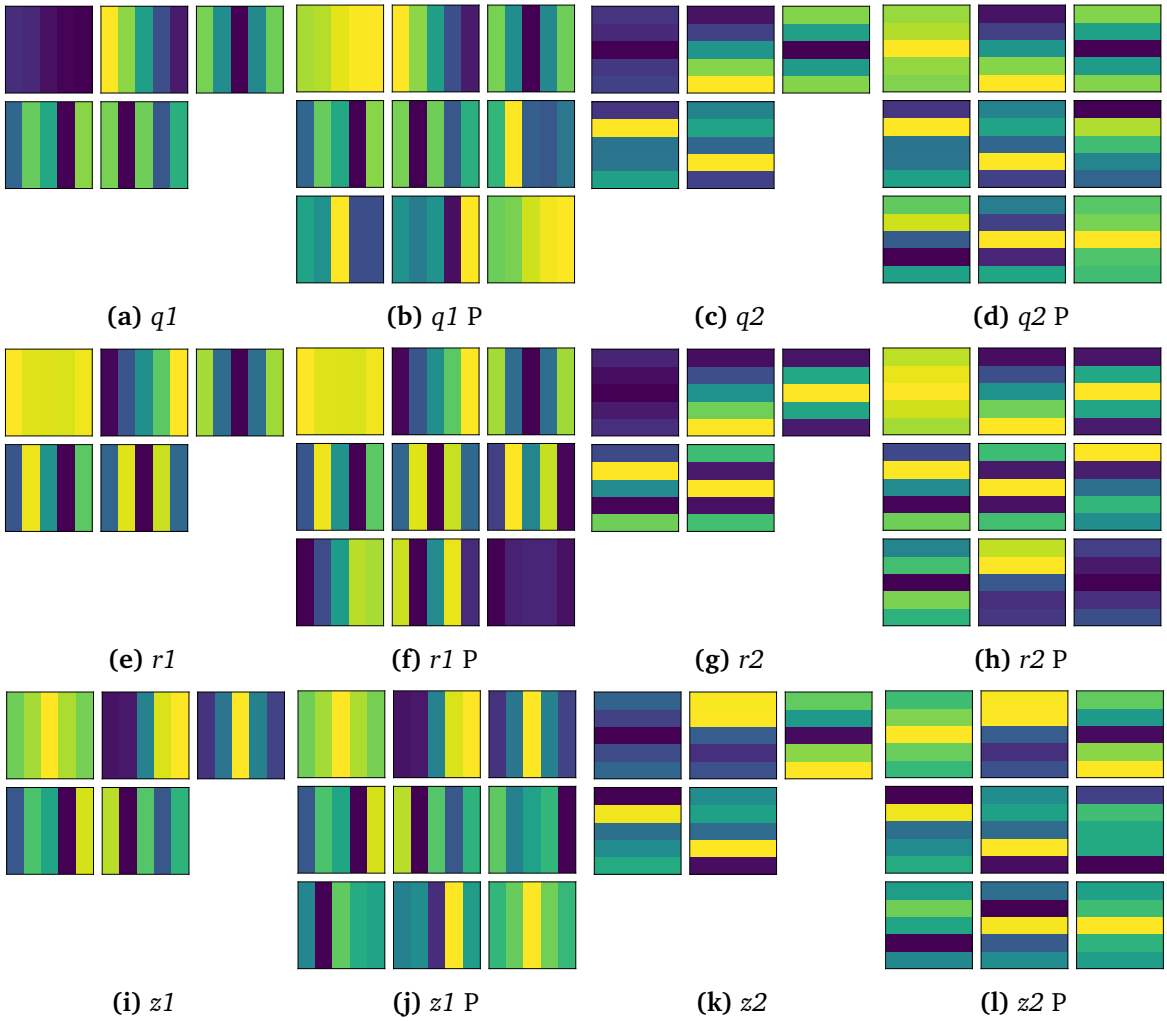
(a) *flow1* **(b)** *flow1* P

(c) *flow2* **(d)** *flow2* P **(e)** *corr2* **(f)** *corr2* P

(g) *motion1* **(h)** *motion1* P

(i) *flowhead1* **(j)** *flowhead1* P **(k)** *flowhead2* **(l)** *flowhead2* P

Figure 3.7: PCA filter dictionaries in the update encoder and flow-head. Parseval completed dictionaries are abbreviated with P.

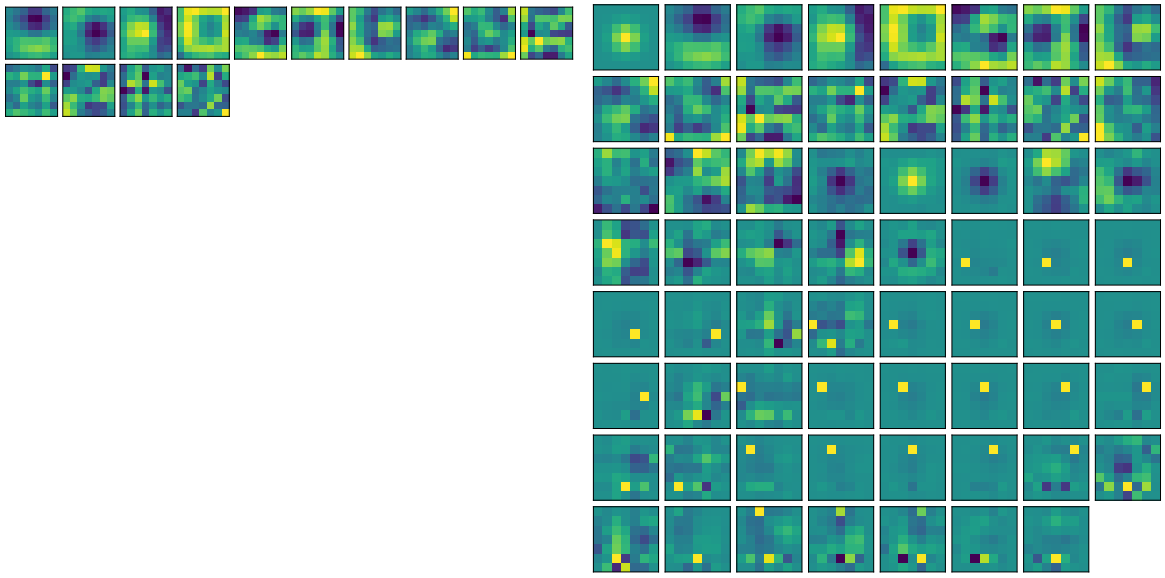


(a) *q1* **(b)** *q1* P **(c)** *q2* **(d)** *q2* P

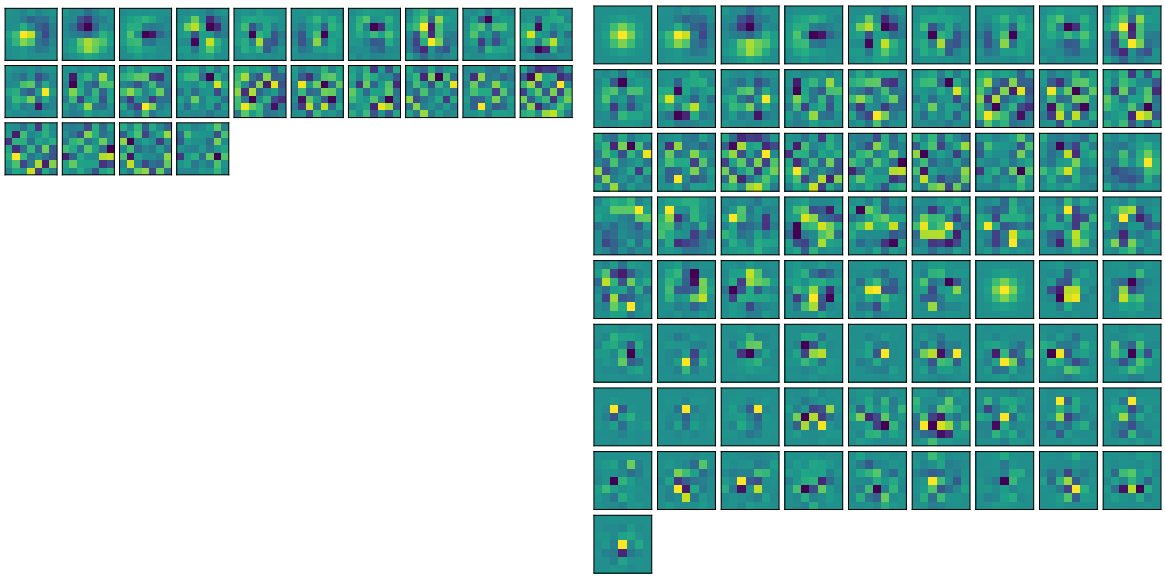
(e) *r1* **(f)** *r1* P **(g)** *r2* **(h)** *r2* P

(i) *z1* **(j)** *z1* P **(k)** *z2* **(l)** *z2* P

Figure 3.8: PCA filter dictionaries in the GRU of the update block. Parseval completed dictionaries are abbreviated with P.



(a) *feature1* 14 filters **(b)** *feature1* 14 filters Parseval



(c) *flow1* 24 filters **(d)** *flow1* 24 filters Parseval

Figure 3.9: Truncated PCA filter dictionaries of *feature1* and *flow1*.

3.4 Parseval Frame Completed Filter Dictionaries

The goal of Parseval completing a FD is to achieve a filter-set which is a Parseval framelet, without loosing custom-selected filters. The important part about framelets is that a Parseval framelet is a generalized wavelet fulfilling the Parseval condition for which [Atreas et al., 2019] prove a sufficient constraint on the set of underlying filters while also allowing room for custom filters. Thus the learned feature representation contains information useful to reconstruct the image in addition to important features such as derivatives.

Sparse Directional Parseval Completed FD (SDp)

[Safari et al., 2020] and [Schmalfuss et al., 2022a] employ this strategy to complete SD-FDs to sparse directional Parseval filter dictionaries, seen in Figure 3.3. The crux is to use the existing dictionary of filters $\{A\} \in \mathbb{R}^{L+1 \times N}$ in vectorized form with $N \coloneqq n \cdot m$ and $L+1 \coloneqq d_{dict} \leq N$. Then take the point wise square-root of the low-pass filter $c = \sqrt{A_0}$, given $A_0 > 0$, scale all high pass filters by some optimized $\lambda^* \in \mathbb{R}^{d_{dict}}$

$$\{D_1(\lambda^*)\} = \{A_d \cdot \frac{\lambda_d^*}{c_d}\}_{1 \leq d \leq L} \tag{3.3}$$

and add a completing set $\{D_2\} \in \mathbb{R}^{\upsilon \times N}$ such that the following condition, equivalent to Equation (2.10), holds:

$$\begin{bmatrix} c \\ D_1(\lambda^*) \\ D_2 \end{bmatrix}^\top \begin{bmatrix} c \\ D_1(\lambda^*) \\ D_2 \end{bmatrix} = I_{N \times N}. \tag{3.4}$$

The resulting dictionary

$$P = \begin{bmatrix} A_0 \\ B \end{bmatrix} = \begin{bmatrix} c \\ D_1(\lambda^*) \\ D_2 \end{bmatrix} \cdot \operatorname{diag}(c) \tag{3.5}$$

contains the low pass filter, all custom high pass filters up to scalar multiplication and some necessary high pass filters D_2.

The complete process of [Schmalfuss et al., 2022a] is shown in Algorithm 3.1, where several changes are marked in color. One conceptual change (red) is to formally extend the completion to rectangular and one-dimensional filters such as the 1×5 filters necessary in the RAFT GRU block. Since the pseudo algorithm works with the vectorized dimension N anyway, only the input and output changes. Additionally, one must not forget to convert c back to the low pass filter A_0 in Line 22.

Algorithm 3.1 Parseval Frame Dictionary Construction

Require: The filter size $n \times m$, a low-pass filter A_0 and L high-pass filters $A_1 \dots A_L$ with $L+1 \leq N \coloneqq n \cdot m$

1 **function** SDPF($N, A_0, A_1, \dots, A_L, \epsilon, \epsilon_{\text{single}}$)
2 $a \leftarrow \Lambda(A_0) \in \mathbb{R}^N$ // Λ: filter to vector
3 $c \leftarrow (\sqrt{a_0}, \sqrt{a_1}, \dots, \sqrt{a_{N-1}}) \in \mathbb{R}^N$ // Add normalization
4 $\lambda \leftarrow (1, \dots, 1) \in \mathbb{R}^L$ // Initialize λ
5 **for** $i = 1 \dots L$ **do**
6 $d_i(\lambda) \leftarrow \lambda_i \left(\frac{\Lambda(A_i)_1}{c_1}, \dots, \frac{\Lambda(A_i)_N}{c_N} \right)$
7 **end for**
8 $D_1(\lambda) \leftarrow \begin{pmatrix} d_1(\lambda) \\ \vdots \\ d_L(\lambda) \end{pmatrix} \in \mathbb{R}^{L \times N}$
9 $\lambda^{init} \leftarrow \frac{(1,\dots,1)}{0.9 \cdot \|(c^\top c + D_1(\lambda)^\top D_1(\lambda))\|} \in \mathbb{R}^L$ // Good initialization, then recurse lines $5-8$
10 $\lambda^* \leftarrow \underset{\lambda}{\text{argmax}} \ \text{trace}(c^\top c + D_1(\lambda)^\top D_1(\lambda)) \text{ s.t. } \|c^\top c + D_1(\lambda)^\top D_1(\lambda)\| \leq 1 - \epsilon_{\text{single}}$
// Adjust bounds from $0 \leq \lambda \leq 1$ to $0 \leq \lambda \leq 1.1 \cdot \max(\lambda^{init})$
11 $Q \leftarrow \begin{pmatrix} c \\ D_1(\lambda^*) \end{pmatrix} \in \mathbb{R}^{(L+1) \times N}$
12 $V, \Sigma_1 \leftarrow$ SVD of $Q = U \Sigma_1 V^\top$
13 **for** $i = 0 \dots L$ **do**
14 $s_i \leftarrow 1 - (\Sigma_1)_{i,i}^2$ // Moved $\sqrt{\cdot}$ to Line 19
15 **if** $s_i < \epsilon$ **then**
16 $s_i \leftarrow 0$
17 **end if**
18 **end for**
19 $\Sigma_2 \leftarrow \text{diag}(0, \sqrt{s_1}, \dots, \sqrt{s_L}, 1, \dots, 1) \in \mathbb{R}^{N \times N}$ // $s_0 \approx 0$
20 $D_2 \leftarrow \Sigma_2 V^\top \in \mathbb{R}^{N \times N}$
21 $B \leftarrow \begin{pmatrix} D_1(\lambda^*) \\ D_2 \end{pmatrix}$ diag(c), eliminate 0-rows
22 $P \leftarrow \begin{pmatrix} c^2 \\ B \end{pmatrix} \in \mathbb{R}^{(\upsilon+1) \times N}, \quad \upsilon \geq N - 1$ // $c^2 = a$
23 **return** P
24 **end function**

Ensures: Parseval filter bank $P \in \mathbb{R}^{(\upsilon+1) \times N}$, with υ high-pass filters and 1 low-pass filter, filters obtained by reshaping the rows of P into $n \times m$ matrices.

Violet adjustments mainly address the numerical stability of the optimization step in Line 10, where the constraint has been sharpened by a single precision $\varepsilon \approx 1e-6$. Secondly, the low pass vector c is implicitly normalized in Line 3. Line 9 empirically chooses a good initialization for λ^{init}, such that the constraint should be slightly violated:

$$\|c^\top c + D_1(\lambda^{init})^\top D_1(\lambda^{init})\| \leq 1 + \|D_1(\lambda^{init})^\top D_1(\lambda^{init})\| = \frac{19}{9} \gtrapprox 1. \tag{3.6}$$

The range of λ^* during optimization in Line 10 has been adjusted from $[0,1]^L$ to $[0, 1.1 \cdot \max(\lambda^{init})]^L$.

Additionally, the singular values $\sqrt{s_i}$ of Σ_2 (Line 14) are cut off already if their square s_i is smaller than ε. Because c is orthogonal to all other filters and normalized, the largest singular value $(\Sigma_1)_{0,0}$ is, up to numerical errors, exactly 1, this does not make a difference here, as it is the only one close enough to yield a cut off in all of the discussed Parseval dictionaries. In fact $\operatorname{rank}(D_2) = N - 1$, and the resulting Parseval frame includes $\upsilon = L + N$ vectors in $\mathbb{R}^N$. To explicitly state said result, Line 13 and 19 formally include a 0 in blue. The initialization of $\lambda^{init} \in \mathbb{R}^N$ in Line 4 is presumed to be a minor writing mistake and marked in blue as well.

Other Parseval Completed FDs (GD4p, PCAp, PCA14p, PCA24p)

With this Parseval framelet completion algorithm the FDs GD4p (Figure 3.1), PCAp for each layer (Figures 3.4 to 3.8), PCA14p for *feature1* and PCA24p for *flow1* (Figure 3.9) can be constructed. The GD4p-FDs are shown for the proof of concept only as they are not used in any experiment. Like PCA14 and PCA24, PCA14p and PCA24p are only used on Experiment 4.2.

Lastly, it is not certain whether PCA trained FDs do have a (reasonable) low pass filter. Hence, one can check if the first filter $P_0 \coloneqq |A_0| > 0$ is strictly positive. If this is not the case, a Gaussian $A_0 = G(x, y, 1)$ is inserted in front of all other filters before Parseval completion. PCA-FDs which do not have a low pass filter are in the layers *feature1* in Figure 3.4a and *featureL10* in Figure 3.5s. Lastly, PCArelp collapses back to PCAp, since the concept of explained variance is not feasible for the additional filters $D_2 \operatorname{diag}(c)$.

3.5 Overview of Filter Dictionaries

This section is dedicated to provide an overview of the most important properties of the mentioned FDs and list the experiments in which they are used. The first three columns of Table 3.1 show the experiment in which the FD is used in, if it is Parseval completed and, if applicable, which scale σ is used for Gaussian kernels. All other columns state for every FD and kernel size the dictionary size d_{dict} and whether it is used in any experiment.

FD	Experiments	Parseval	σ	1×5, 5×1		3×3		7×7	
				d_{dict}	used	d_{dict}	used	d_{dict}	used
GD9	4.2	–	1	55	–	55	–	55	✓
GD4	4.2 – 4.5	–	1	15	✓	15	✓	15	–
GD4s	4.3 – 4.5	–	$1, 2, 4, 8$	60	–	60	–	60	✓
GD4s05	4.3 – 4.4	–	$0.5, 1, 2, 4$	60	–	60	–	60	✓
SD	4.2 – 4.4	–	1	5	✓	9	✓	49	✓
PCA	4.2 – 4.4	–	–	5	✓	9	✓	49	✓
PCAx	4.2	–	–	x	–	x	–	x	✓
PCArel	4.3 – 4.4	–	–	5	✓	9	✓	49	✓
GD4p	–	✓	–	63	–	63	–	63	–
SDp	4.2 – 4.4	✓	1	9	✓	17	✓	97	✓
PCAp	4.3 – 4.4	✓	–,1	9	✓	$17, 18$	✓	$97, 98$	✓
PCAxp	4.2	✓	–,1	$x + 4$ $(+1)$	–	$x + 8$ $(+1)$	–	$x + 48$ $(+1)$	✓

Table 3.1: Overview of all FDs and in which Experiments 4 they are used.

4 Experiments

By taking the various filter dictionaries from Chapter 3, this chapter addresses Questions **Q.1** and **Q.2** to identify the most beneficial FD type and whether a Parseval completion is necessary to improve results. Determining in which layers FDs are most effective, Question **Q.3**, is also investigated by placing receptive fields with various FD types into different layer positions of the RAFT network.

It starts by first presenting the reference RAFT baseline and the common training schedule, evaluation metrics and attack method in Experiment 4.1. Experiment 4.2 then inserts FDs into single layers of the network, followed by Experiments 4.3 and 4.4 which replace the whole feature encoder and update block with receptive field layers. Lastly, Experiment 4.5 tries to replicate and improve the performance gains reached with GD-FDs in the update block by constraining the Lipschitz constant of selected layers.

4.1 Baseline

The RAFT network baseline is trained and evaluated similar to the description of [Teed and Deng, 2020] by pre-training on FlyingChairs and FlyingThings then finetuning on Sintel in combination with KITTI and HD1K data for $100k$ iterations each [Dosovitskiy et al., 2015; Mayer et al., 2016; Butler et al., 2012; Menze and Geiger, 2015; Kondermann et al., 2016]. It provides the filters to construct the PCA-FDs in Section 3.3. During training the computations are done in single precision, whereas mixed precision is used for inference. The baseline and all other configurations are evaluated using the previous flow estimate as the initial guess (warm-start) and the number of flow update iterations is kept unaltered at 32 on Sintel and 24 on KITTI.

Quantities of interest include the average endpoint error $\mathrm{AEE}(f, f^g) = \frac{1}{I}\sum_{i\in I}\|f_i - f_i^g\|_2$ towards the ground truth, averaged on the Sintel clean and final training sets, and the clean and final AEE score of the Sintel submission. Additionally and without further finetuning, the cross data-set generalization on the KITTI training data is considered in terms of the AEE and the percentage of outliers F1-all exceeding 3 pixels and 5% of the ground truth value. Lastly, the robustness against the PCFA, as explained in Section 2.1.3, is evaluated with respect to the $\mathrm{AEE}(\check{f}, f)$ of the attacked flow against the initial flow prediction, the smaller the better, and the $\mathrm{AEE}(\check{f}, f^t)$ of the attacked flow against the target flow, the bigger the better. Discussions

of results in the following chapters neglect the training performance and focus predominantly on the Sintel test final, KITTI F1-all, and the robustness $\mathrm{AEE}(\check{f}, f)$ metrics.

The target flow of the attack is chosen to be the zero flow $f^t = 0$, while the perturbations for each image are kept below $\delta_1, \delta_2 \leq \varepsilon_2 = 1 \times 10^{-3} = 0.1\%$ by using the Lagrange multiplier $\mu = 10^6$. This attack corresponds to the weaker attack shown in Figure 2.3c and thus allows not only potential improvements in robustness but also leaves room to recognize more susceptible RAFT configurations.

To reduce the number of uploads to the Sintel benchmark, the Sintel training-test-split of [Zhao et al., 2020a] is used to evaluate all configurations. Finetuning with this split omits 20% of sequences during the final $100k$ iterations which are reserved for subsequent evaluation. Only selected configurations are then trained on the full Sintel training data-set, starting with the already trained weights after $200k$ iterations.

Discussion

Even though results are similar to [Teed and Deng, 2020], there is a non-negligible spread of quality and robustness, c.f. Table 4.1, where values differ in the second or third digits. To partially compensate for this effect, the baseline is trained and evaluated twice on the Sintel split. The trained baselines are then attacked up to three times for both training scenarios. By assuming normal distributed values, a crude estimate of the expected range can be made with which the modified configurations are compared against. The mean μ and the 2- and 3-σ confidence intervals (CI) of the split and full Sintel trained baselines are shown in Table 4.2.

Training	Sintel (train)		Sintel (test)		KITTI (train)		PCFA AEE	
	Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
split	0.627	0.984	1.667	3.695	1.469	5.442	11.982	5.768
	0.637	0.997	1.690	3.569	1.453	5.339	11.944	5.703
							12.078	5.572
full	0.739*	1.198*	1.613*	2.743*	1.557	5.755	12.60*	5.46*
							12.157	5.918

Table 4.1: Evaluation of the baseline RAFT network configuration trained on the Sintel benchmark and the Sintel split [Butler et al., 2012; Zhao et al., 2020a]. Attacks are performed on the Sintel test final benchmark. Results with asterix* are taken from [Walter, 2022].

Training	Bound	Sintel (train)		Sintel (test)		KITTI (train)		PCFA AEE	
		Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
split	$\mu - 3\sigma$	0.611	0.963	1.630	3.365	1.427	5.172	11.794	5.381
	$\mu - 2\sigma$	0.618	0.972	1.646	3.454	1.438	5.245	11.863	5.481
	μ	0.632	0.991	1.679	3.632	1.461	5.391	12.001	5.681
	$\mu + 2\sigma$	0.646	1.009	1.711	3.810	1.484	5.536	12.139	5.881
	$\mu + 3\sigma$	0.653	1.018	1.727	3.899	1.495	5.609	12.209	5.981
full	$\mu - 3\sigma$	–	–	–	–	–	–	11.439	4.717
	$\mu - 2\sigma$	–	–	–	–	–	–	11.752	5.041
	μ	0.739	1.198	1.613	2.743	1.557	5.755	12.379	5.689
	$\mu + 2\sigma$	–	–	–	–	–	–	13.005	6.337
	$\mu + 3\sigma$	–	–	–	–	–	–	13.318	6.661

Table 4.2: Mean and confidence intervals of the baseline quality and robustness evaluations. Attacks are performed on the Sintel test final benchmark.

4.2 Different Filter Dictionaries in a Single Layer

This first experiment is a direct continuation of the authors project work [Walter, 2022], by extending the choice of FDs placed in the first layers of the feature encoder and update block. For *feature1* all in Table 3.1 described categories of FDs are employed, while for *corr2*, *flow1* only the SDp and in the latter case also PCA24 is used. The results are shown in Tables 4.3 and 4.4 with the joint evaluation of the metrics of focus, i.e. quality on Sintel split/full final data, KITTI training set F1-all and its robustness $\mathrm{AEE}(\check{f}, f)$, being visualized in Figure 4.1 where the expected CIs of the baseline are indicated by filled and framed boxes.

Discussion

When trained on the Sintel split, the majority of configurations stay within the 2σ CI for both quality measures and are less robust, i.e. outside the 3σ CI. Exceptions are PCA14p and SDp in *feature1* which show a significantly better quality on the test split, while SDp in *corr2* is just outside the 2σ CI . Similarly, SDp in *flow1* is more robust, yet remains in the 3σ CI and with comparable quality as the baseline. GD9 yields the worst fit on the Sintel split but still within the 3σ CI and is the least robust among all configurations.

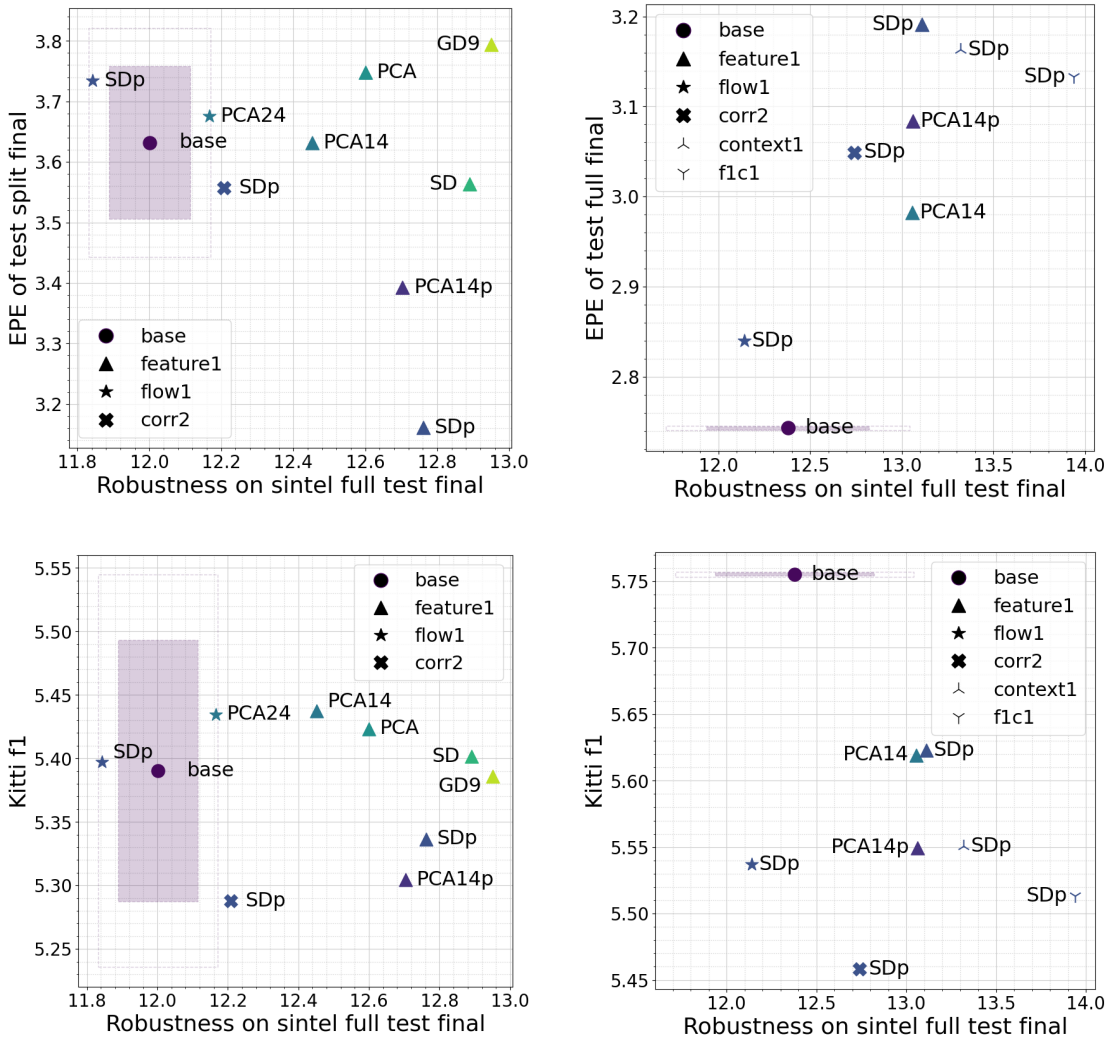
Training on the full Sintel data-set shows a much larger CI on the robustness axis, while no information is available in the quality direction. Nevertheless, all submitted configurations display a worse evaluation on the Sintel benchmark but better quality on the KITTI training

Configuration		Split (train)		Split (test)		KITTI (train)		PCFA AEE	
Layer	FD	Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
base		0.627	0.984	1.667	3.695	1.469	5.442	11.982	**5.768**
		0.637	0.997	1.690	3.569	1.453	5.339	11.944	5.703
								12.078	5.572
feature1	GD9	0.624	0.990	1.734	3.794	1.468	5.386	12.951	4.714
	SD	0.627	0.980	**1.602**	3.564	1.451	5.401	12.890†	4.906†
	SDp	0.621	1.029	1.687	**3.161**	**1.450**	5.337	12.762	5.016
	PCA	**0.615**	0.971	1.652	3.748	1.462	5.423	12.600	4.982
	PCA14	0.630	0.973	1.621	3.631	1.471	5.437	12.452	5.226
	PCA14p	0.625	0.981	1.627	3.393	1.454	**5.304**	12.704	5.024
flow1	SDp	0.623	**0.962**	1.716	3.734	1.485	5.397	**11.843**	5.730
	PCA24	0.636	0.981	1.597	3.675	1.452	5.435	12.166	5.558
corr2	SDp	0.636	0.970	1.738	3.557	1.466	5.288	12.208	5.438

Table 4.3: Evaluation of the single layer RAFT network configurations trained on the Sintel split [Zhao et al., 2020a]. Attacks are performed on the Sintel test final benchmark. Attacks with dagger$^{\dagger x}$ have $x \leq 5$ frames filtered out such that the average perturbation is below 0.0012. Best results are displayed in **bold** font.

Configuration		Sintel (train)		Sintel (test)		KITTI (train)		PCFA AEE	
Layer	FD	Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
base		0.739*	1.198*	1.613*	**2.743***	1.557	5.755	12.60*	5.46*
								12.157	**5.918**
f1c1	SDp	0.752*	1.354*	1.639*	3.133*	1.538	5.513	13.94*	4.00*
context1	SDp	0.767*	1.321*	1.635*	3.163*	1.512	5.551	13.32*	4.80*
flow1	SDp	0.751*	**1.182***	1.639*	2.840*	1.535	5.537	**12.14***	5.90*
feature1	SDp	0.743*	1.245*	1.616*	3.191*	1.578	5.623	13.11*	4.81*
	PCA14	**0.735**	1.184	1.715	2.982	1.527	5.619	13.056	4.913
	PCA14p	0.708	1.184	1.648	3.084	1.521	5.550	13.061	4.895
corr2	SDp	0.760	1.214	**1.553**	3.049	**1.501**	**5.458**	12.741	5.306

Table 4.4: Evaluation of the single layer RAFT network configurations trained on the Sintel benchmark. Attacks are performed on the Sintel test final data-set. Results with asterix* are taken from [Walter, 2022]. Best results are displayed in **bold** font.



(a) Trained on Sintel split data-set

(b) Trained on full Sintel data-set

Figure 4.1: Joint evaluation of the baseline and single layer RAFT network configurations by prediction quality and adversarial robustness on Sintel (top) and KITTI (bottom). All RAFT network configurations are trained on the Sintel split (left) and full Sintel data-set (right). Assuming a normal distribution of the baseline trials, the $2-$ and $3-\sigma$ confidence intervals are shown. In Figure 4.1b the confidence interval is known only for robustness. *feature1context1* is abbreviated as *f1c1*.

data. Their robustness is generally worse with the ones having receptive fields in the update block being close to the upper bound of the 2σ CI. SDp in *flow1*, although better than the baseline, is now located well within the 2σ CI.

Overall, the configurations which are significantly better on Sintel split test final, i.e. PCAp and SDp in *feature1*, do not transfer this trait to the fully trained case. Thus, all configurations do not change the network enough to make a detectable difference, disregarding the worse robustness. When trained on the full Sintel data-set, there seems to be a trade-off between the quality on the test data and the generalization capability. Yet, one must keep in mind that the CI is not computed for the full training scenario in the scope of this thesis and this trade-off is not observed in the split scenario. Added with the best results being spread over the configurations in the other quality metrics, as seen in Tables 4.3 and 4.4, no clear trend is visible to deduce a strong conclusion besides for receptive fields in the feature encoder decreasing robustness no matter the FD choice.

Layer Wise Mean Neuron Activation

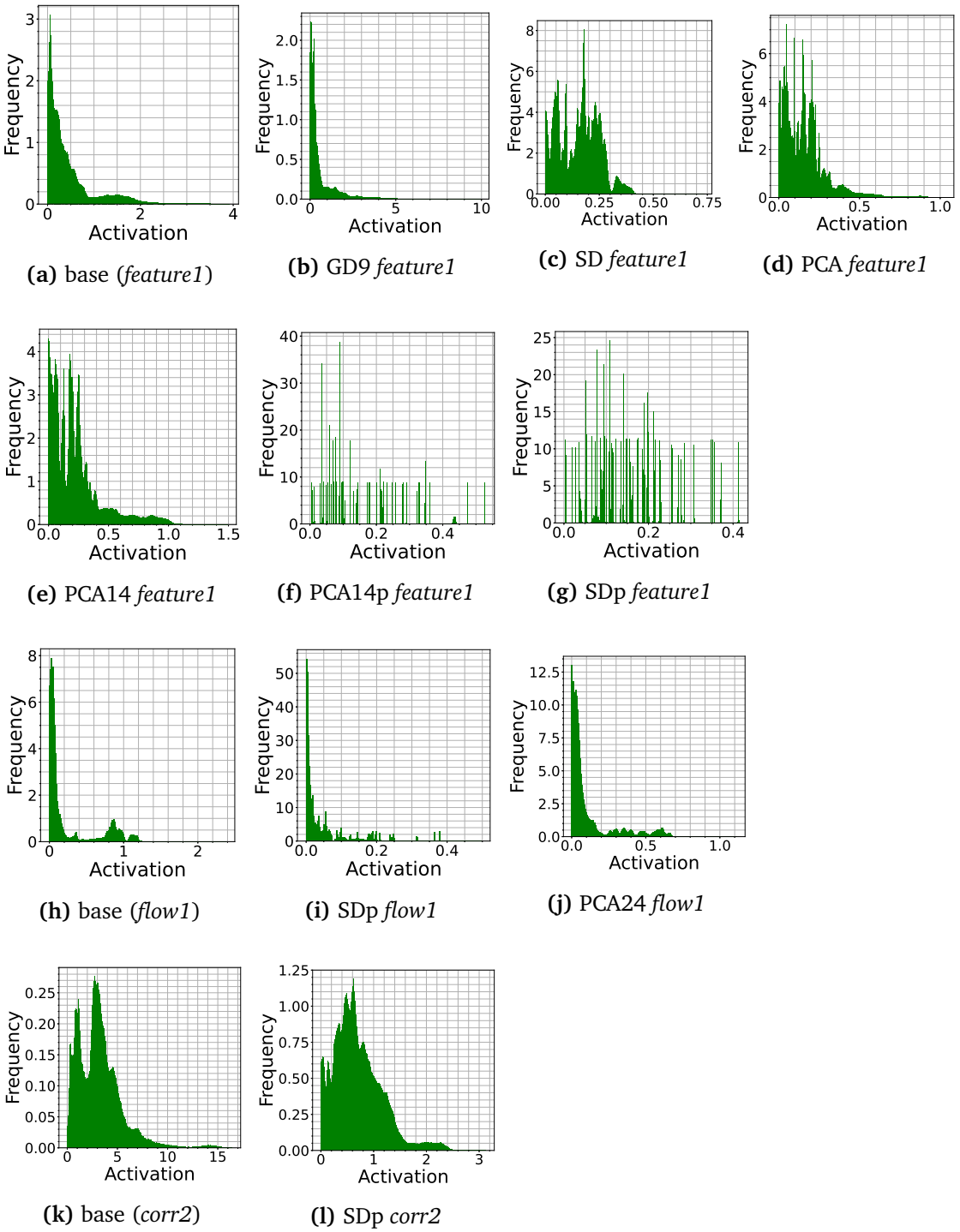
With the quality and robustness results being largely inconclusive, the magnitude of the networks neuron activations gives insight into the changes occurring through the use of receptive fields. [Srivastava et al., 2014] show that the regularization technique of dropout improves generalization and induces a more sparse activation of neurons, since the mean of the mean neuron activations decreases. The normalized histogram of mean neuron activation, during the evaluation on the Sintel training and test split sampled to one third of their size, seen in Figures 4.2 and 4.3, also exhibit a drop of highly activated neurons for receptive field layers. One single exception being the GD9 in *feature1* which is the least robust configuration and the worst fit to the Sintel split test.

Next to much smaller activations for the remaining FDs, it can be seen in these histograms in Figures 4.2 and 4.3 that especially Parseval completed FDs in *feature1* show many specific amplitudes which get activated rather often. This could possibly be caused by some neurons that are activated the same across the whole data-set. Thus they do not hold any significant additional information and remain unused and untrained, effectively diminishing the networks performance in the first layers of the network. An attack would therefore need to trigger less neurons to develop an adversarial behavior. The later layers of *flow1* and *corr2* are much less impaired, coinciding with the similar performance as the baseline.

Yet, this line of reasoning does only explain why the Parseval completed FDs in *feature1* are less robust, but not for the other configurations. Since each kernel of this first layer lies within a fixed three-dimensional space, it is plausible to assume it being easier for the PCFA to trigger more kernels with perturbation patterns. Some kernels might also have overlapping bases which is less likely in the baseline and thus one single pattern easily changes multiple neurons

at once. For the receptive fields in the hidden layers, *flow1* and *corr2*, this is more difficult for the attack as there are other network layers to trick first.

In conclusion of this first experiment, it can be established that the Parseval completion might introduce filters which provide no practical use for the network. Additionally, replacing only one layer in RAFT does not yield definite conclusions other than including receptive fields into the first layers of the feature encoder makes the networks more volatile against the PCFA. This stands very much in contrast to [Schmalfuss et al., 2022a], where the first layers have the most positive impact. Hence, the next sections deal with changing the whole feature encoder or update block.



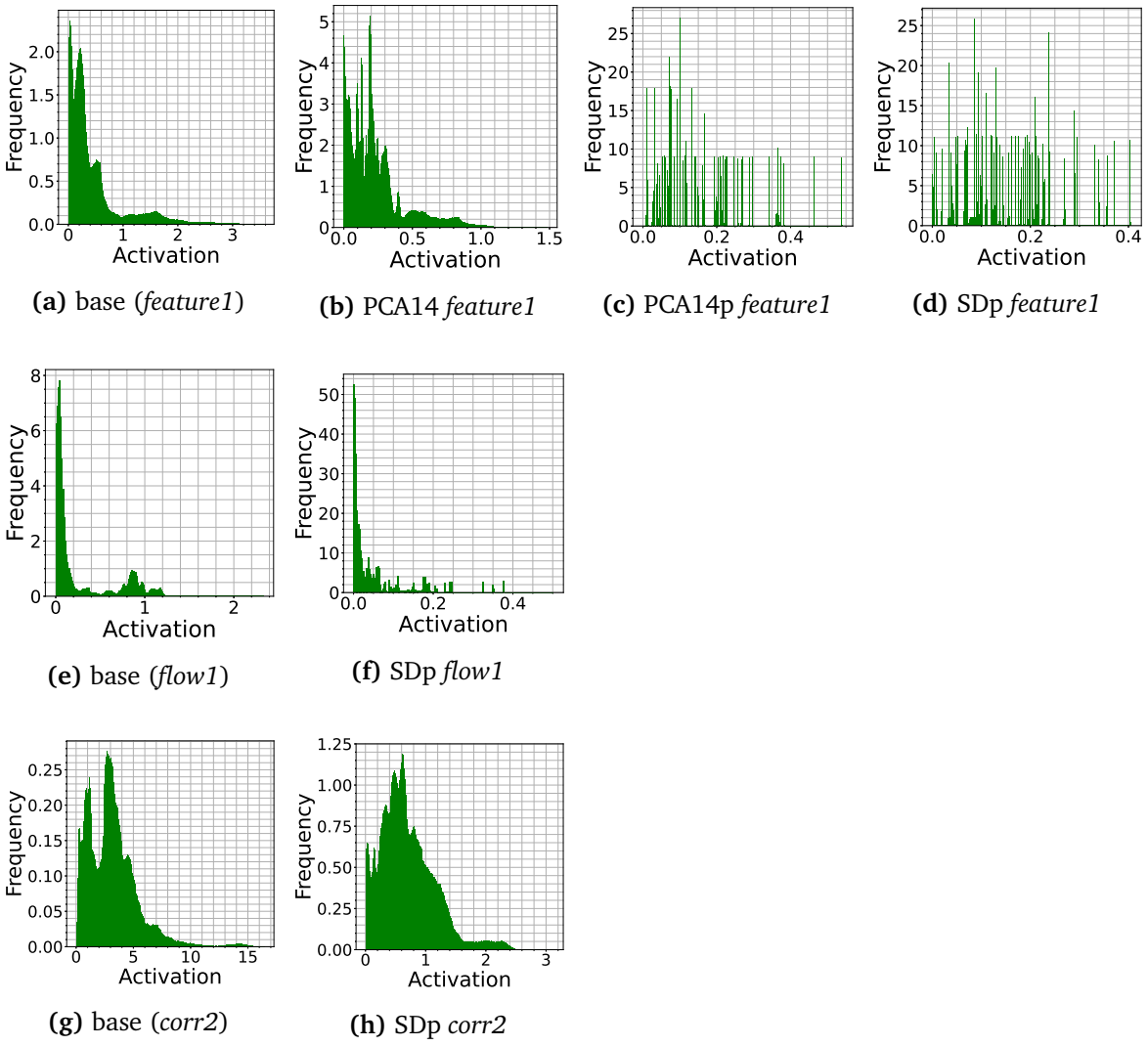
(a) base (*feature1*) **(b)** GD9 *feature1* **(c)** SD *feature1* **(d)** PCA *feature1*

(e) PCA14 *feature1* **(f)** PCA14p *feature1* **(g)** SDp *feature1*

(h) base (*flow1*) **(i)** SDp *flow1* **(j)** PCA24 *flow1*

(k) base (*corr2*) **(l)** SDp *corr2*

Figure 4.2: Normalized histogram of the average neuron activation on the full Sintel train and test data-set for receptive field layers *feature1*, *flow1* and *corr2* (top to bottom row). All RAFT network configurations are trained on the Sintel split data-set.



(a) base (*feature1*) **(b)** PCA14 *feature1* **(c)** PCA14p *feature1* **(d)** SDp *feature1*

(e) base (*flow1*) **(f)** SDp *flow1*

(g) base (*corr2*) **(h)** SDp *corr2*

Figure 4.3: Normalized histogram of the average neuron activation on the full Sintel train and test data-set for receptive field layers *feature1*, *flow1* and *corr2* (top to bottom row). All RAFT network configurations are trained on the full Sintel data-set.

4.3 Different Filter Dictionaries in the Complete Feature Encoder

Since inserting only one receptive field layer in the previous experiment does not result in clear trends, this section places FDs into all convolutional layers of the RAFT feature encoder: *feature1*, *featureL1-featureL12*. All in Table 3.1 described categories of FDs are used and results are shown individually in Tables 4.5 and 4.6 at the end of this section. Again, the quality metrics of focus are also shown jointly with the robustness in Figure 4.5.

Discussion

In the joint quality and robustness evaluations of the split training in Figure 4.5a at the end of this section it is visible that GD4s, SDp and PCArel show the same inconclusive behavior as in the single layer experiment by being less robust and in the 2σ CI in quality on the Sintel test. GD4s05 and SD have a worse fit on the data with the first being on the lower bound of the 3σ robustness CI and the latter again worse than the baseline. The only improvement in quality, still in the 3σ CI, is achieved by the PCA-FD which inherits filters learned by RAFT on the full Sintel training, thus, including the test data in this scenario making the result not surprising. KITTI evaluations reveal similar trends for the mentioned configurations, while SD shows stronger generalization by moving up relative to the others and PCA being worse than the baseline.

The PCAp FD proves to be a significantly more robust configuration in trade-off with quality both on Sintel and KITTI. Therefore, it is trained but not submitted on the Sintel benchmark, because the train and KITTI evaluations are about ten times as large as the baseline. Although being more robust in the split scenario, PCAp still seems to worsen performance.

Lipschitz Constants

Gaining insight into the changes of the receptive fields and identifying the cause of PCAp's unusual behavior is still of interest. As discussed in Section 2.1.3, the Lipschitz constant plays a crucial role in the network's ability to generalize and defend against adversarial attacks. In individual two-dimensional convolutional layers l with kernel size $2k+1$ the Lipschitz constant can be computed by using the bound of [Cisse et al., 2017a]:

$$\Lambda_l \leq (2k+1)\|W\|_2, \tag{4.1}$$

where $W \in \mathbb{R}^{d_{out} \times (2k+1)^2 d_{in}}$ are the weights of all convolutions in flattened representation. Instead of this upper bound which suffices here, [Gouk et al., 2021] propose to use the power method to exactly compute the largest singular value of $W^\top W$ by iteratively applying a forward and a backward pass on an arbitrary input vector.

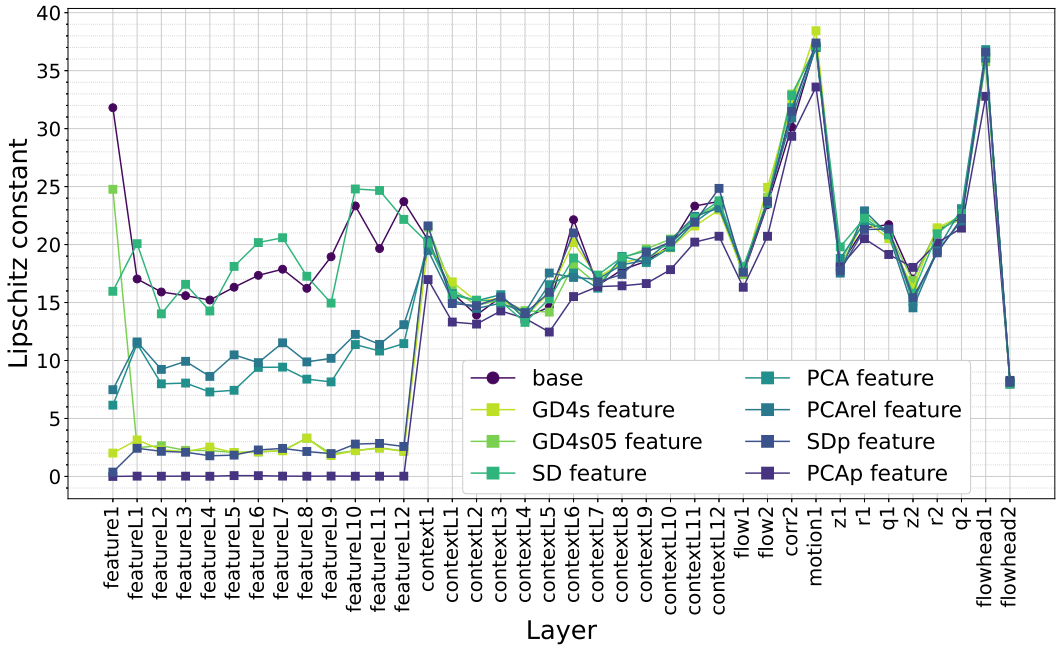


Figure 4.4: Lipschitz constants of every named convolutional layer in the baseline and feature encoder RAFT network configurations. Layers are grouped into feature block, context block, update block from left to right. All RAFT network configurations are trained on the Sintel split data-set.

The calculated Lipschitz constants of each named layer can be seen in Figure 4.4, where FDs reduce the Lipschitz bounds, with the exception of SD, and the remaining layers stay close to the baseline. While PCA and PCArel have Lipschitz constants in the feature encoder at about 10, GD4s, GD4s05 and SDp lie between 2 and 3. In the extreme case of PCAp the bound is, with a magnitude of 10^{-4} to 10^{-3}, very close to zero. Since the Parseval completed filters are imposed on the network by the receptive field in addition to the less used principle components, these could make up many kernels with for optical flow uninformative features and extremely small weights bringing down the Lipschitz constant. This is further supported by similar findings in the neuron activations of the previous Experiment 4.2. Hence, this is most likely the reason why the performance suffers significantly during full Sintel training, since gradients are hardly propagated into the earlier layers due to the small weights caused by many, presumably non-relevant filters.

The question remains why the lower Lipschitz constants of the feature encoder, seen in Figure 4.4, do not make an impact on performance for the other configurations. To answer this, it is important to note, that the overall Lipschitz constant is also influenced by activations functions, residual connections and batch norms. Non-linear activation functions like ReLU have a trivial Lipschitz constant of 1 and can be neglected. Batch norms, however, are present in the RAFT encoders after every convolutional layer (*feature1* excluded). It is thus reasonable

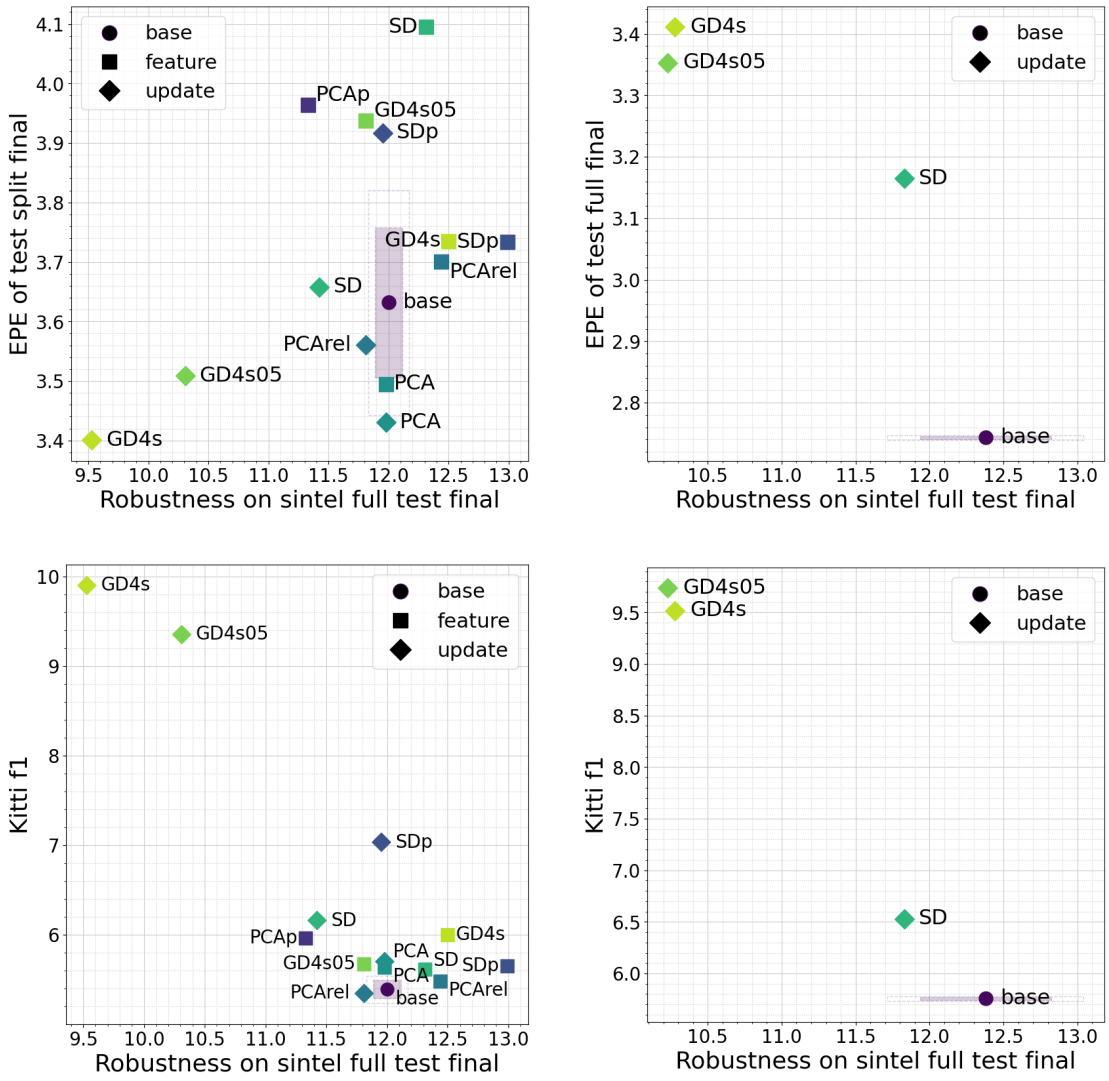
to assume that this dissipates any effects of Lipschitz constants by normalizing the range of activations fittingly to similar bounds for the baseline and its modifications. Therefore, no configurations are reliably able to improve the networks quality nor robustness.

Configuration		Split (train)		Split (test)		KITTI (train)		PCFA AEE	
Layer	FD	Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
base		**0.627**	**0.984**	**1.667**	3.695	1.469	5.442	11.982	5.768
		0.637	0.997	1.690	3.569	**1.453**	**5.339**	11.944	5.703
								12.078	5.572
feature	GD4s	0.683	1.049	1.970	3.735	1.543	6.000	12.501	5.230
	GD4s05	0.671	1.024	1.868	3.937	1.537	5.676	$11.812^{\dagger}$	$5.735^{\dagger}$
	SD	0.668	1.052	1.812	4.095	1.527	5.612	12.314	5.378
	SDp	0.662	1.035	1.877	3.734	1.495	5.654	12.993	4.607
	PCA	0.664	1.052	1.937	3.495	1.507	5.636	11.978	5.651
	PCArel	0.645	1.012	1.783	3.700	1.479	5.480	12.440	5.397
	PCAp	0.699	1.065	2.003	3.963	1.586	5.967	11.330	6.270
update	GD4s	0.800	1.204	1.990	**3.401**	2.085	9.901	$\mathbf{9.529}^{\dagger}$	$\mathbf{7.899}^{\dagger}$
	GD4s05	0.787	1.245	1.941	3.509	1.960	9.354	$10.307^{\dagger 2}$	$7.011^{\dagger 2}$
	SD	0.660	1.089	1.824	3.658	1.592	6.161	$11.423^{\dagger}$	$5.840^{\dagger}$
	SDp	0.733	1.138	2.129	3.917	1.791	7.034	11.951	5.283
	PCA	0.641	1.064	1.717	3.430	1.524	5.706	11.978	5.597
	PCArel	0.634	1.041	1.692	3.561	1.475	5.351	11.812	5.810
	PCAp	–	–	–	–	–	–	–	–

Table 4.5: Evaluation of the baseline, feature encoder and update block RAFT network configurations trained on the Sintel split [Zhao et al., 2020a]. Attacks are performed on the Sintel test final data-set. Attacks with dagger$^{\dagger x}$ have $x \leq 5$ frames filtered out such that the average perturbation is below 0.0012. Best results are displayed in **bold** font.

Configuration		Sintel (train)		Sintel (test)		KITTI (train)		PCFA AEE	
Layer	FD	Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
base		**0.739***	**1.198***	**1.613***	**2.743***	**1.557**	**5.755**	12.60* 12.157	5.46* 5.918
feature	PCAp	7.102	9.480	–	–	10.681	42.983	$11.957^{\dagger}$	$5.786^{\dagger}$
update	GD4s	0.981	1.431	2.178	3.411	2.062	9.519	10.278	7.206
	GD4s05	0.938	1.472	2.075	3.353	2.021	9.737	**10.230**	**7.700**
	SD	0.785	1.285	1.736	3.165	1.680	6.530	11.828	5.908

Table 4.6: Evaluation of the baseline, feature encoder and update block RAFT network configurations trained on the Sintel benchmark. Attacks are performed on the Sintel test final data-set. Results with asterix* are taken from [Walter, 2022]. Attacks with dagger$^{\dagger x}$ have $x \leq 5$ frames filtered out such that the average perturbation is below 0.0012. Best results are displayed in **bold** font.



(a) Trained on Sintel split data-set

(b) Trained on full Sintel data-set

Figure 4.5: Joint evaluation of the baseline, feature encoder and update block RAFT network configurations by prediction quality and adversarial robustness on Sintel (top) and KITTI (bottom). All RAFT network configurations are trained on the Sintel split (left) and full Sintel data-set (right). Assuming a normal distribution of the baseline trials the $2-$ and $3-\sigma$ confidence intervals are shown. In Figure 4.5b the confidence interval is known only for robustness.

4.4 Different Filter Dictionaries in the Complete Update Block

Performing the same trials as in the previous Section, now with FDs in the RAFT update block, results are shown again in Tables 4.5 and 4.6 and jointly in Figure 4.5 located immediately in front of this section. To reiterate, all in Table 3.1 described categories of FDs are inserted into the complete update block: *flow1-2*, *corr2*, *motion1*, *z1-2*, *r1-2*, *q1-2* and *flowhead1-2*.

Discussion

Similar to FDs in the feature encoder, the Sintel fitted FDs PCA performs slightly better on the Sintel split lying just outside the 3σ CI as well as being worse on KITTI. PCArel is more robust than the baseline also by the 3σ margin. Both of these improvements can be traced back to having the underlying filters learned on Sintel data.

In this experiment, PCAp is not able to learn the first pre-training phase FlyingChairs and is prematurely stopped after 28k iterations. It is assumed that this is caused by too many uninformative filters being enforced onto the network, coinciding with findings from the previous two experiments. The other Parseval completed SDp-FD performs worse on the Sintel and KITTI test sets, but is still able to be competitive, likely because half of the dictionary is composed of a low pass filter and useful derivative filters, while the additional filters also contain multiple first derivative filters, c.f. the visual representation in Figure 3.3.

Considerable improvements in robustness are achieved by the remaining SD-, GD4s05- and GD4s-FDs and boost performance on the Sintel splits test while declining on the KITTI train data. These configurations are then trained and submitted to the Sintel benchmark loosing accuracy on both data-sets for a gain in robustness. GD4s and GD4s05 achieve the largest development thus far, the first of which being over 20% more robust. Since the employed Gaussian derivatives compute derivatives of smoothed versions of images (for *flow1* even on multiple scales), suggests that they are more invariant under noisy changes in the correlation volume and altered features caused by attacks. Most importantly, a consistent trend is visible when submitting these three configurations to the Sintel benchmark.

Lipschitz Constants

Similar to FDs in the feature encoder, Lipschitz constants seen in Figure 4.6 decrease in receptive field layers by small amount for the baseline similar configurations of PCA and PCArel. SDp shows a decrease in all update block layers, however, the robust cases of SD, GD4s and GD4s05 contrarily increase the constant in the GRU. This hints to a different, not robustifying reduction of Lipschitz induced by the Parseval completed FD, possibly caused by the additional, for optical flow as uninformative understood Parseval filters. Further investigating the robust configurations, the decrease in the layers apart from the GRU correlate to the

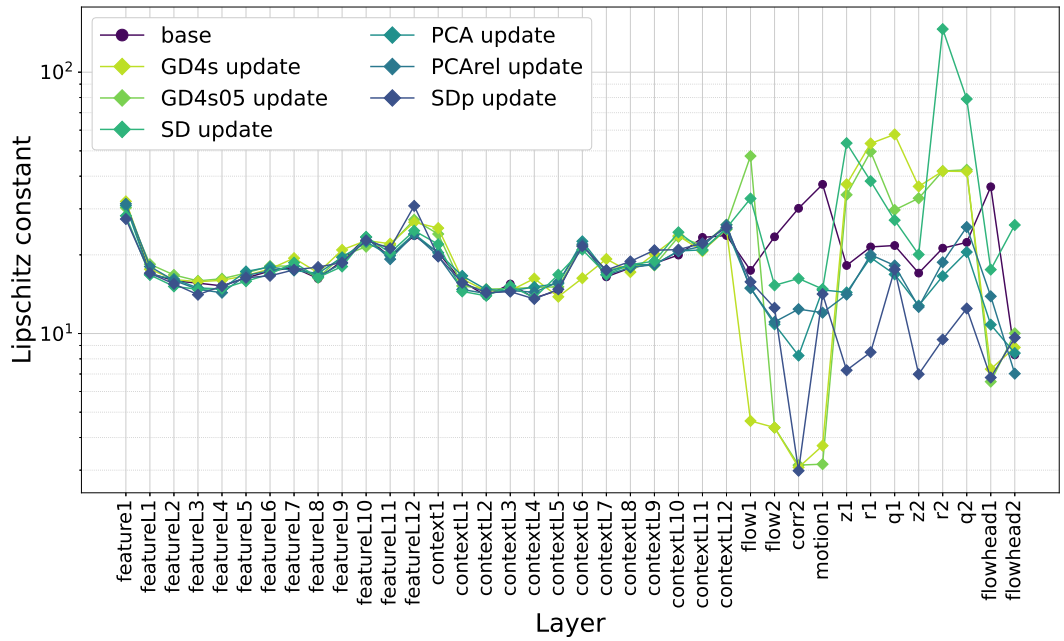
achieved robustness with SD reducing the constant by a smaller amount than the Gaussian derivative FDs and hence only improving to a lesser extent.

A closer look into the RAFT architecture identifies the hyperbolic tangent and sigmoid as activation functions in the GRU, mapping into $[-1, 1]$ and $[0, 1]$ respectively. Like the batch norms in the feature encoder, these contractions are likely mitigating any effects of changing Lipschitz constants. The remaining layers of the motion encoder and flowhead (*flow1*, *flow2*, *corr2*, *motion1*, *flowhead1*, *flowhead2*) only have the ReLU as their activation function. Thus, the in comparison to the baseline amplifying GRU might compensate for smaller neuron activations produced by previous attenuating layers, while not affecting the robustness negatively.

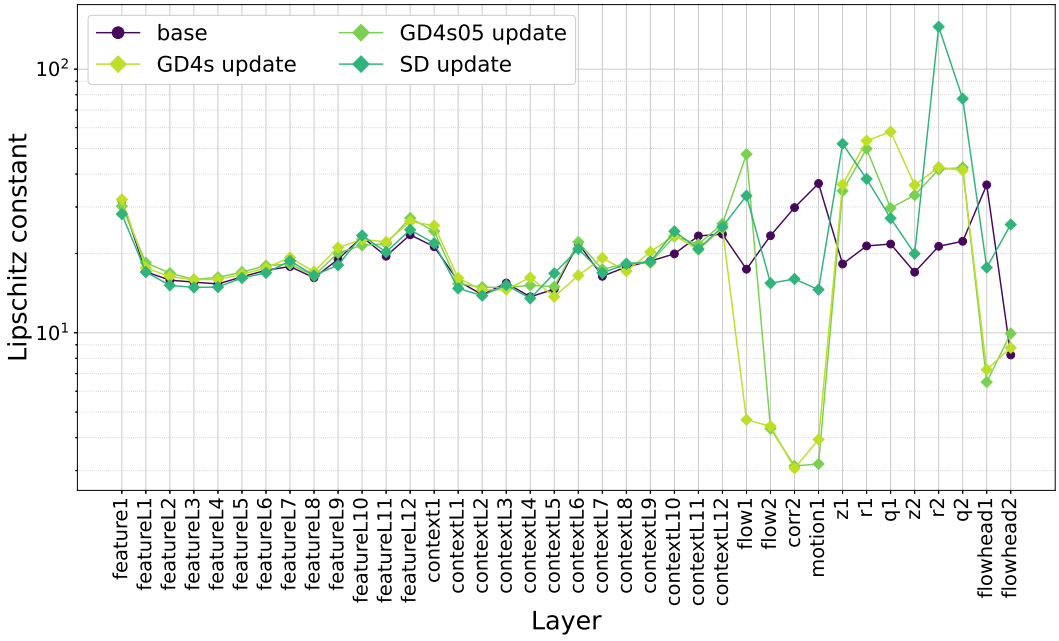
Besides the GD- and SD-FDs enforcing derivative filters upon the network's kernels, it is thus concluded that lower Lipschitz constants in these layers cause improved robustness. This effect is also clearly visible, since the update block is passed through multiple times during evaluation, in contrast to the feature encoder. Counter-intuitively, the higher Lipschitz constants in the GRU might also play a role in retaining the majority of the networks quality, since SDp has got lower constants, but lacks the higher values like the others.

Additional Remark

An additional remark is that the increased robustness of GD4s manifests itself in very few scenes where the network largely remains unimpaired under attack and is considerably better than the baseline. The most notable examples can be seen in Figure 4.7. Other patches show similar volatility as the baseline, leading to the conclusion that GD4s robustifies only in these particular scenes instead of partly improving every flow estimation.



(a) Trained on Sintel split data-set



(b) Trained on full Sintel data-set

Figure 4.6: Lipschitz constants of every named convolutional layer in the baseline and update block RAFT network configurations. Layers are grouped into feature block, context block, update block from left to right. All RAFT network configurations are trained on the Sintel split (top) and full Sintel (bottom) data-set.

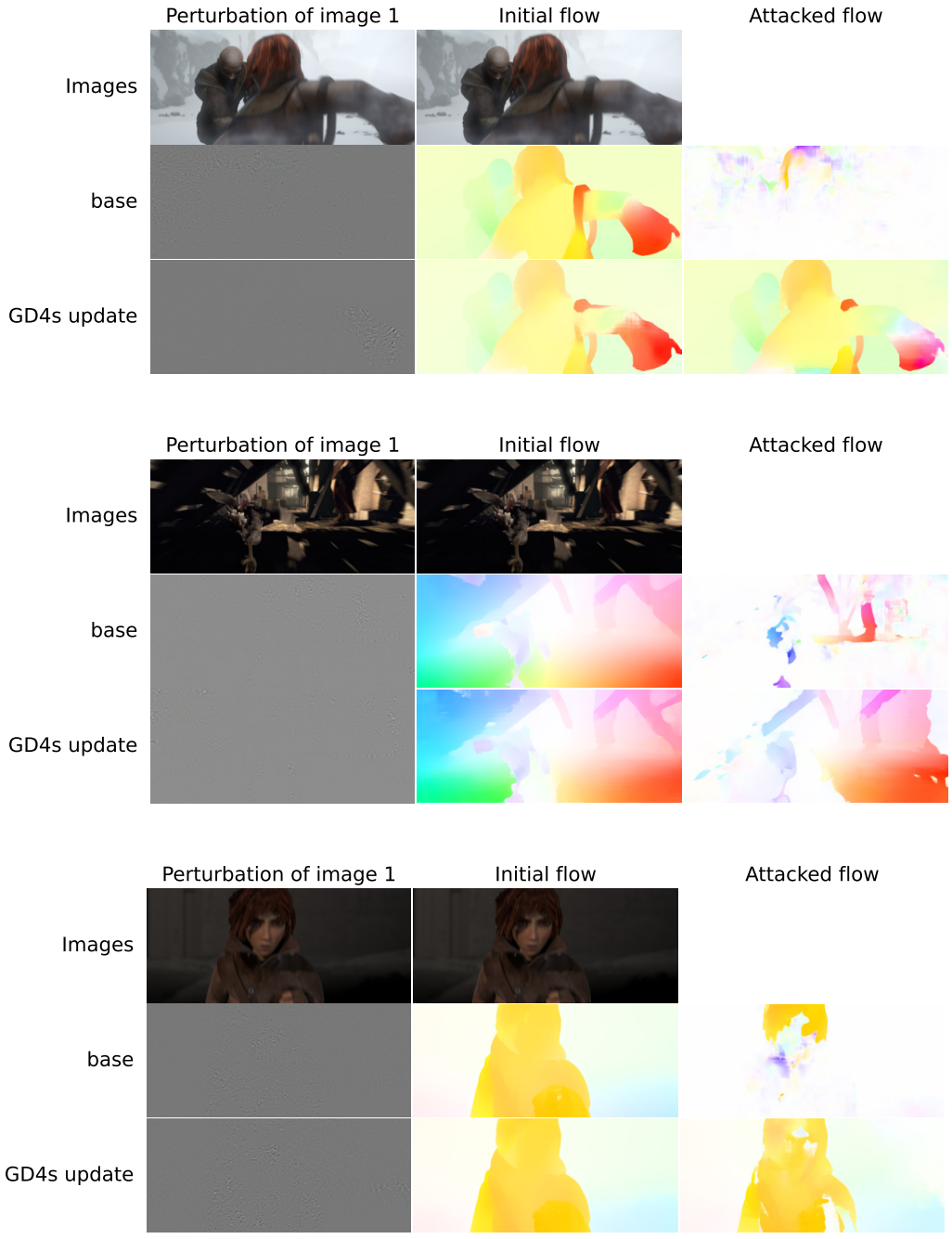


Figure 4.7: PCFA attack comparison of the baseline RAFT network against the more robust RAFT with GD4s in the update block. The target flow is zero flow. Shown is only a subset of frames where a significant difference in the attacked flow is present. Both configurations are trained on the full Sintel data-set.

4.5 Lipschitz Constant Constraint Regularization

This last experiment builds upon the successful improvement of robustness achieved by SD-, GD4s05- and GD4s-FDs in the update block, c.f. Experiment 4.4. Their common trend in Lipschitz constants is a decrease in all update block layers except in the GRU, i.e. *flow1-2 corr2*, *motion1* and *flowhead1-2*. Hence, GD4s as the most robust configuration and the baseline are considered for Lipschitz constant constraint regularization (LCC) following the method of [Gouk et al., 2021] in these five layers. While the regularized GD4s aims at further improving the robustness, the baseline seeks to replicate the success without receptive fields.

Lipschitz Constants

To hold the Lipschitz constant below $\lambda > 0$, all weights are scaled down accordingly after each backward pass, if the bound is exceeded. By choosing $\lambda = 1$ and $p = 2$ the additional update, as discussed in Section 2.1.3, reads for fully connected layers $W \leftarrow \frac{1}{\max(1, \|W\|_2)} W$. Instead of the exact spectral norm for convolutions, the easy to compute bound $(2k+1)\|W\|_2$ of Equation (4.1) is used.

Verifying the regularization technique, the layers in question indeed have a Lipschitz constant of 1 which can be seen in Figure 4.8. As it is already the case in the robust GD- and SD-configurations configuration of Experiment 4.4, the Lipschitz constants in the GRU also grow larger. In terms of Lipschitz constants, the same behavior as the robust configurations can be achieved through plain Lipschitz regularization.

Discussion

Results of the Sintel split trained scenario are shown in Table 4.7 and jointly in in Figure 4.9. In the joint quality and robustness evaluation the dictionary EYE labels the regularized baseline in which the identity FD is used as an implementation trick. Both Lipschitz constant constrained configurations are worse than their counterparts. The regularized baseline has a worse fit and no effect onto the robustness, while the LCC GD4s looses in both metrics to GD4s. Each regularization diminishes the quality on the KITTI training data-set by roughly 100% effectively quadrupling the F1-all score with respect to the baseline in the case of LCC GD4s.

Overall, the usage of GD-FDs is, in this small scenario, neither replicable nor improvable by constraining the Lipschitz constant with the method of [Gouk et al., 2021] who note that their results heavily depend on the choice of the bounding hyperparameter λ. GD-FDs are therefore not improving RAFT only by reducing the layer's Lipschitz constants, but rather through their geometric interpretation with the smaller constants being a mere side effect. Hence, Gaussian derivative FDs provide a unique approach to robustify neural networks in optical flow.

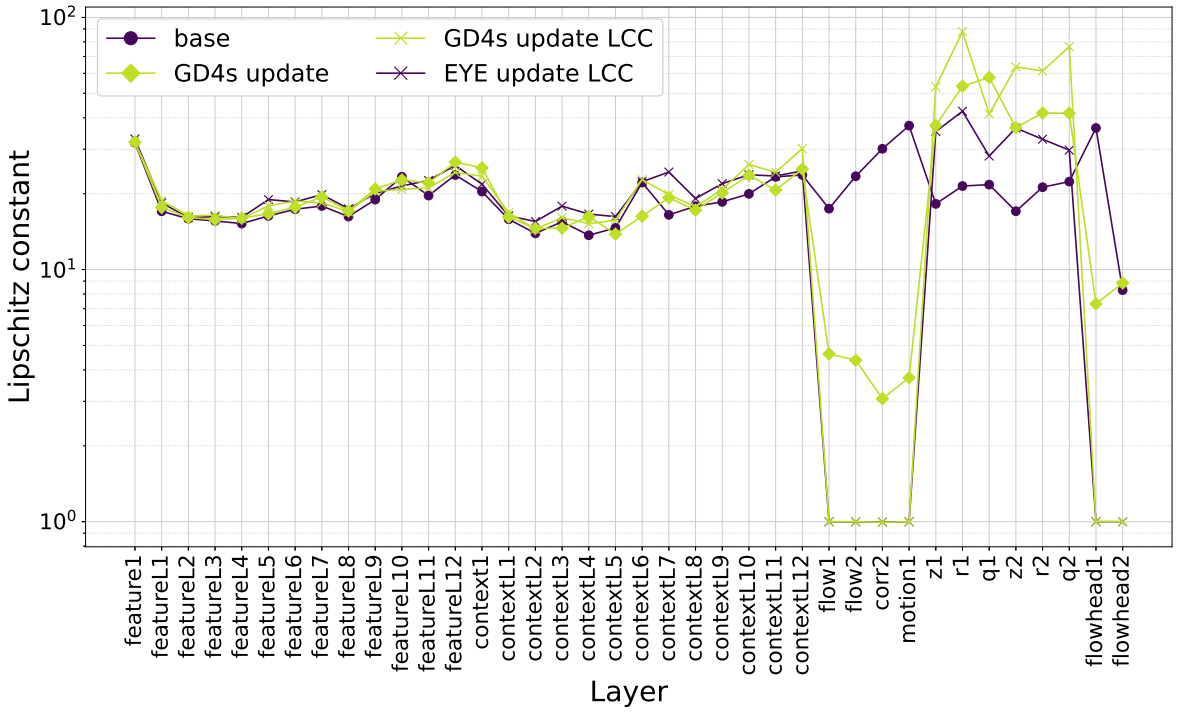


Figure 4.8: Lipschitz constants of every named convolutional layer in the baseline, GD4s and Lipschitz regularized RAFT network configurations. Layers are grouped into feature block, context block, update block from left to right. All RAFT network configurations are trained on the Sintel split data-set.

Configuration		Split (train)		Split (test)		KITTI (train)		PCFA AEE	
Layer	FD	Clean	Final	Clean	Final	AEE	F1-all	$\check{f} \leftrightarrow f$	$\check{f} \leftrightarrow f^t$
base		**0.627**	**0.984**	**1.667**	3.695	1.469	5.442	11.982	5.768
		0.637	0.997	1.690	**3.569**	**1.453**	**5.339**	11.944	5.703
								12.078	5.572
update	GD4s	0.800	1.204	1.990	3.401	2.085	9.901	**9.529**†	**7.899**†
update LCC	–	1.028	1.400	2.370	4.264	2.273	9.882	12.009$^{\dagger 5}$	4.804$^{\dagger 5}$
	GD4s	1.574	2.063	3.014	4.808	3.534	17.727	10.099	6.968

Table 4.7: Evaluation of the baseline, GD4s and Lipschitz regularized RAFT network configurations trained on the Sintel split [Zhao et al., 2020a]. Attacks are performed on the Sintel test final benchmark. Attacks with dagger$^{\dagger x}$ have $x \leq 5$ frames filtered out such that the average perturbation is below 0.0012. The perturbation for the LCC regularized baseline is at 0.001456. Best results are displayed in **bold** font.

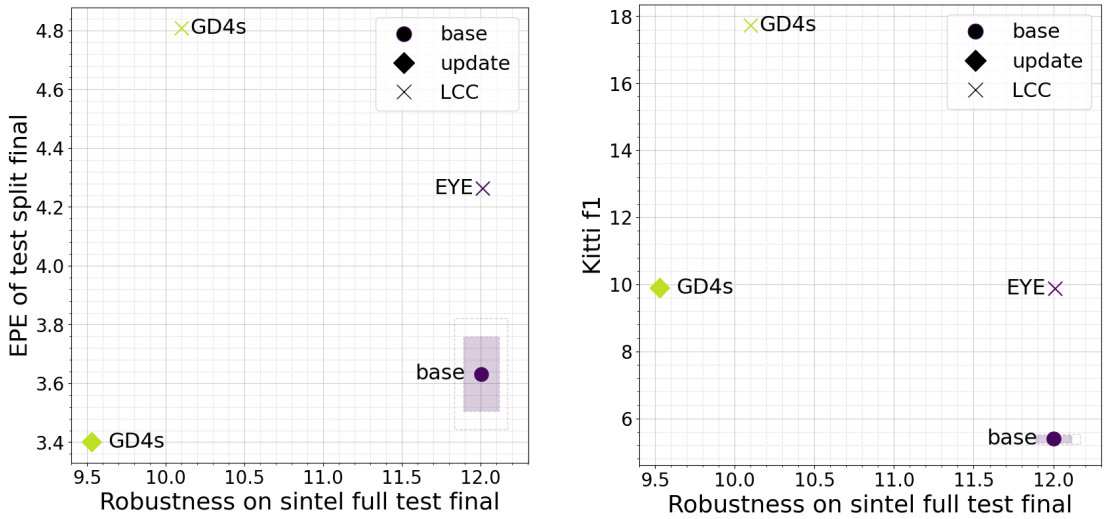


Figure 4.9: Joint evaluation of the baseline and Lipschitz regularized RAFT network configurations by prediction quality and adversarial robustness on Sintel (left) and KITTI (right). All RAFT network configurations are trained on the Sintel split data-set. EYE is the identity FD and denotes the regularized baseline.

5 Conclusions and Outlook

In summary, this thesis employs RFCNNs in the context of optical flow prediction and adversarial robustness by using various FDs and a sparsity constraint of three basis-filters per kernel. Starting with the baseline RAFT network, unmodified Sintel training schedule and inserting GD-, SD-, SDp- Parseval completed FDs with useful geometric properties, it introduces a novel PCA-FD which consists of the principal components of the previously trained network's kernels. While the ablation PCAp enforces Parseval frame properties onto the FD, PCArel mimics the original network as close as possible albeit reducing the learnable space for the kernels.

Conclusions

Results show that modifying single layers in RAFT do not positively impact robustness and often lead to minuscule changes in quality indicating the benefits of the already extensive training schedule. Larger shifts are achieved by extending receptive field layers to the complete feature encoder and update block. The trained PCA-FDs slightly improve the quality on the test split of the training data-set Sintel by choosing appropriately fitting filters to start with. A Comparison between the SD- and PCA-FDs and their Parseval completed counterparts reveals worse and in the case of PCAp sometimes catastrophic performance in both metrics. Since these Parseval completed FDs introduce additional filters holding useful information for subsequent image reconstruction, they may hold advantages for blind image inpainting in [Schmalfuss et al., 2022a], but have adverse effects in the non-reconstructive task of optical flow generation.

While receptive fields in the feature encoder yield worse results in either quality or robustness, the geometric motivated SD- and GD-FDs improve adversarial robustness when placed in the update block. The smooth Gaussian derivatives are over 20% more robust. However, quality on the Sintel benchmark and KITTI training data diminishes in exchange. Hence, the two measures of robustness against adversarial attack and a network's generalization capability can be isolated in this scenario, meaning a more robust network might be worse on other, more distant data-sets.

Additionally, a connection between receptive fields, convolutional layer Lipschitz constants and adversarial robustness can be drawn. FDs generally reduce the layer's Lipschitz constant with the exception of the GRU, where an increase is observed. By further lowering the constant of a GD configuration and the unmodified RAFT to be below $\lambda = 1$ the results were not replicable. As [Gouk et al., 2021] note, the performance of Lipschitz constant constraint regularization

is very sensitive to the imposed bound λ. In this limited trial, GD-FDs thus pose a unique approach for improving a networks robustness which is more than just decreasing the Lipschitz constant.

Outlook

Overall, only the GD-FDs, and to a smaller extent SD-FDs, in the update block improve the robustness of RAFT, while Parseval completed dictionaries worsen performance in optical flow prediction. Conversely, it is not yet observed whether this applies to reconstructive tasks such as in blind image inpainting in [Schmalfuss et al., 2022a] as well or Parseval completion instead yields a better performance than plain FDs. Gaussian derivatives thus provide a promising research direction for any neural network involving the extraction of features linked to derivatives and the desire to become more robust against adversarial attacks. Besides optical flow, applications include but are not limited to other computer vision tasks [Voulodimos et al., 2018], e.g. adversarial attack susceptible object detection [Wu et al., 2020].

Remaining in the problem statement of this thesis, FDs when placed into the feature encoder of RAFT improve neither quality nor robustness. It is pointed out that these are the only layers holding normalization layers, i.e. a batch norm, which might interfere with the inserted FDs. Therefore, a trial with removed batch norms could conclude whether the position itself or the usage of normalization layers is unsuited for receptive fields.

A topic closely related is the reduction of individual layers' Lipschitz constants through the use of receptive fields. Since FDs only affect the convolutional layers directly and not adjacent normalization and activation functions, the overall Lipschitz constant might remain unaltered and thus leading to unchanged adversarial robustness. Next to receptive fields, a more sophisticated investigation of Lipschitz constraining, normalization techniques and contracting activation functions pose an alternative way to regularize the layers in the update block and therefore, improving the robustness of RAFT. Viable options for Lipschitz constraining neural networks could be computational intensive Parseval networks [Cisse et al., 2017a] or the rescaling technique of [Gouk et al., 2021] which is applied in this thesis, while also refraining from upper bounds of the Lipschitz constants by using the power method and including batch norms in the regularization.

Bibliography

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160. (Cited on page 15)

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430. (Cited on pages 7 and 18)

Akhtar, N., Mian, A., Kardan, N., and Shah, M. (2021). Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196. (Cited on page 18)

Atreas, N., Karantzas, N., Papadakis, M., and Stavropoulos, T. (2019). On the design of multi-dimensional compactly supported parseval framelets with directional characteristics. *Linear Algebra and its Applications*, 582:1–36. (Cited on pages 23 and 37)

Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77. (Cited on page 13)

Bertero, M., Poggio, T. A., and Torre, V. (1988). Ill-posed problems in early vision. *Proceedings of the IEEE*, 76(8):869–889. (Cited on page 13)

Bhambri, S., Muku, S., Tulasi, A., and Buduru, A. B. (2019). A survey of black-box adversarial attacks on computer vision models. *arXiv preprint arXiv:1912.01667*. (Cited on page 18)

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer. (Cited on pages 13, 14 and 15)

Bruhn, A., Weickert, J., Feddern, C., Kohlberger, T., and Schnörr, C. (2003). Real-time optic flow computation with variational methods. In *International Conference on Computer Analysis of Images and Patterns*, pages 222–229. Springer. (Cited on page 14)

Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pages 611–625. Springer. (Cited on pages 7, 12, 14, 20, 41 and 42)

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017a). Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863. PMLR. (Cited on pages 7, 20, 50 and 64)

Cisse, M. M., Adi, Y., Neverova, N., and Keshet, J. (2017b). Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. *Advances in neural information processing systems*, 30. (Cited on pages 7 and 18)

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327. (Cited on page 14)

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. (Cited on page 16)

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766. (Cited on pages 7, 14 and 41)

Fortun, D., Bouthemy, P., and Kervrann, C. (2015). Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21. (Cited on page 13)

Freeman, W. T., Adelson, E. H., et al. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906. (Cited on pages 22 and 25)

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*. (Cited on pages 7 and 19)

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. (2021). Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416. (Cited on pages 7, 9, 19, 50, 59, 63 and 64)

Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., and Yang, G.-Z. (2019). Xai—explainable artificial intelligence. *Science robotics*, 4(37):eaay7120. (Cited on page 15)

Horn, B. and Schunck, B. (1981). Determining optical flow. intechniques and applications of image understanding (vol. 281, pp. 319-331). *International Society for Optics and Photonics*. (Cited on pages 7 and 13)

Huang, L., Gao, C., Zhou, Y., Xie, C., Yuille, A. L., Zou, C., and Liu, N. (2020). Universal physical camouflage attacks on object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 720–729. (Cited on page 18)

Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., Dai, J., and Li, H. (2022a). Flowformer: A transformer architecture for optical flow. *arXiv preprint arXiv:2203.16194*. (Cited on pages 7, 14 and 16)

Huang, Z., Zhang, T., Heng, W., Shi, B., and Zhou, S. (2022b). Real-time intermediate flow estimation for video frame interpolation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIV*, pages 624–642. Springer. (Cited on page 7)

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470. (Cited on pages 7 and 14)

Ilic, F., Pock, T., and Wildes, R. P. (2022). Is appearance free action recognition possible? In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, pages 156–173. Springer. (Cited on page 7)

Jacobsen, J.-H., Van Gemert, J., Lou, Z., and Smeulders, A. W. (2016). Structured receptive fields in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619. (Cited on pages 8, 11, 21 and 25)

Jiang, S., Campbell, D., Lu, Y., Li, H., and Hartley, R. (2021). Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9772–9781. (Cited on pages 7 and 14)

Jonschkowski, R., Stone, A., Barron, J. T., Gordon, A., Konolige, K., and Angelova, A. (2020). What matters in unsupervised optical flow. In *European Conference on Computer Vision*, pages 557–572. Springer. (Cited on page 14)

Kim, D., Woo, S., Lee, J.-Y., and Kweon, I. S. (2019). Deep video inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5792–5801. (Cited on page 7)

Koenderink, J. J. and Van Doorn, A. J. (1987). Representation of local geometry in the visual system. *Biological cybernetics*, 55(6):367–375. (Cited on pages 22 and 25)

Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., and Jahne, B. (2016). The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. (Cited on page 41)

Kong, L. and Yang, J. (2022). Mdflow: Unsupervised optical flow learning by reliable mutual knowledge distillation. *IEEE Transactions on Circuits and Systems for Video Technology*. (Cited on page 14)

Li, Y. Y., Craft, J., Cheng, Y., Schapiro, W., Gliganic, K., Haag, E., and Cao, J. J. (2022). Optical flow analysis of left ventricle wall motion with real-time cardiac magnetic resonance imaging in healthy subjects and heart failure patients. *Annals of Biomedical Engineering*, 50(2):195–210. (Cited on page 7)

Liu, X., Liu, H., and Lin, Y. (2020). Video frame interpolation via optical flow estimation with image inpainting. *International Journal of Intelligent Systems*, 35(12):2087–2102. (Cited on page 7)

Long, P. M. and Sedghi, H. (2019). Generalization bounds for deep convolutional neural networks. *arXiv preprint arXiv:1905.12600*. (Cited on page 23)

Maurer, D., Stoll, M., Volz, S., Gairing, P., and Bruhn, A. (2017). A comparison of isotropic and anisotropic second order regularisers for optical flow. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 537–549. Springer. (Cited on pages 13 and 14)

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on page 41)

Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070. (Cited on pages 7, 14 and 41)

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773. (Cited on pages 7 and 18)

Nagel, H.-H. and Enkelmann, W. (1986). An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):565–593. (Cited on pages 13 and 14)

Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782. (Cited on page 19)

Olden, J. D. and Jackson, D. A. (2002). Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, 154(1-2):135–150. (Cited on page 15)

Ranjan, A. and Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 7 and 14)

Ranjan, A., Janai, J., Geiger, A., and Black, M. J. (2019). Attacking optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. (Cited on pages 7 and 18)

Romeny, B. M. H. (2008). *Front-end vision and multi-scale image analysis: multi-scale computer vision theory and applications, written in mathematica*, volume 27. Springer Science & Business Media. (Cited on page 22)

Safari, K., Haque, M., Karantzas, N., Shahraki, F. F., Prasad, S., and Labate, D. (2020). Improved image classification using receptive field convolutional neural networks. (Cited on pages 21, 22, 23 and 37)

Scheurer, E. (2022). An optimization approach to attacking the horn and schunck model. Bachelors' thesis, Institute for Visualisation and Interactive Systems, University of Stuttgart. (Cited on page 14)

Schmalfuss, J., Scheurer, E., Zhao, H., Karantzas, N., Bruhn, A., and Labate, D. (2022a). Blind image inpainting with sparse directional filter dictionaries for lightweight cnns. *Journal of Mathematical Imaging and Vision*. (Cited on pages 8, 11, 21, 23, 37, 47, 63 and 64)

Schmalfuss, J., Scholze, P., and Bruhn, A. (2022b). A perturbation-constrained adversarial attack for evaluating the robustness of optical flow. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 183–200. Springer. (Cited on pages 7, 11, 18 and 21)

Schrodi, S., Saikia, T., and Brox, T. (2022). Towards understanding adversarial robustness of optical flow networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8916–8924. (Cited on page 18)

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR. (Cited on page 15)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. (Cited on page 46)

Sun, D., Herrmann, C., Reda, F., Rubinstein, M., Fleet, D. J., and Freeman, W. T. (2022). Disentangling architecture and training for optical flow. In *European Conference on Computer Vision*, pages 165–182. Springer. (Cited on pages 11 and 14)

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943. (Cited on pages 7, 14 and 15)

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks: Proceedings of the international conference on learning representations. (Cited on pages 7, 18 and 19)

Teed, Z. and Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 402–419, Cham. Springer International Publishing. (Cited on pages 7, 14, 15, 17, 20, 41 and 42)

Tehrani, A., Mirzaei, M., and Rivaz, H. (2020). Semi-supervised training of optical flow convolutional neural networks in ultrasound elastography. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part III 23*, pages 504–513. Springer. (Cited on page 7)

Ullah, A., Muhammad, K., Del Ser, J., Baik, S. W., and de Albuquerque, V. H. C. (2018). Activity recognition using temporal optical flow convolutional features and multilayer lstm. *IEEE Transactions on Industrial Electronics*, 66(12):9692–9702. (Cited on page 7)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. (Cited on page 16)

Virmaux, A. and Scaman, K. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31. (Cited on pages 7 and 19)

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., et al. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018. (Cited on page 64)

Walter, P. (2022). Sparse filters for optical flow robustness. Project-work simtech, Institute for Visualisation and Interactive Systems, University of Stuttgart. (Cited on pages 8, 9, 11, 23, 42, 43, 44 and 53)

Wang, H., Cai, P., Fan, R., Sun, Y., and Liu, M. (2021). End-to-end interactive prediction and planning with optical flow distillation for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238. (Cited on page 7)

Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., and Bischof, H. (2009). Anisotropic huber-l1 optical flow. In *BMVC*, volume 1, page 3. (Cited on page 14)

Wu, Z., Lim, S.-N., Davis, L. S., and Goldstein, T. (2020). Making an invisibility cloak: Real world adversarial attacks on object detectors. In *European Conference on Computer Vision*, pages 1–17. Springer. (Cited on pages 7, 18 and 64)

Xu, R., Li, X., Zhou, B., and Loy, C. C. (2019). Deep flow-guided video inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3723–3732. (Cited on page 7)

Yan, W., Wang, Y., van der Geest, R. J., and Tao, Q. (2019). Cine mri analysis by deep learning of optical flow: Adding the temporal dimension. *Computers in biology and medicine*, 111:103356. (Cited on page 7)

Yoshida, Y. and Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*. (Cited on pages 7 and 19)

Yu, H., Chen, X., Shi, H., Chen, T., Huang, T. S., and Sun, S. (2020). Motion pyramid networks for accurate and efficient cardiac motion estimation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VI 23*, pages 436–446. Springer. (Cited on page 7)

Yu, J. J., Harley, A. W., and Derpanis, K. G. (2016). Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision*, pages 3–10. Springer. (Cited on page 14)

Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, pages 214–223. Springer. (Cited on pages 13 and 14)

Zhang, H. and Wang, J. (2019). Towards adversarially robust object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 421–430. (Cited on page 19)

Zhao, S., Sheng, Y., Dong, Y., Chang, E. I., Xu, Y., et al. (2020a). Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287. (Cited on pages 14, 42, 44, 52 and 61)

Zhao, Y., Man, K. L., Smith, J., Siddique, K., and Guan, S.-U. (2020b). Improved two-stream model for human action recognition. *EURASIP Journal on Image and Video Processing*, 2020:1–9. (Cited on page 7)

Zimmer, H., Bruhn, A., and Weickert, J. (2011). Optic flow in harmony. *International Journal of Computer Vision*, 93(3):368–388. (Cited on pages 13 and 14)

All links were last followed on February, 26, 2023.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature