UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence

Master Thesis

# Fields of Experts for Network Inversion and Adversarial Restoration

by

Frederik Harder

10986847

January 5, 2018

36 Credits
April 2017 - January 2018

*Supervisors:*
Patrick Putzky
Jörn Jacobsen

*Examiner:*
Prof Max Welling

*Assessor:*
Dr Joris Mooij

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

## Acknowledgements

I would like to thank my supervisors Jörn Jacobsen and Patrick Putzky for their continued patience and support in bringing this project to a good conclusion despite some challenges along the way. Their perspectives on how to approach deep learning research have given me access to many parts of the field which had seemed impenetrable to me before.

My gratitude goes out to my defense committee, Max Welling and Joris Mooij for taking the time to read and evaluate my report.

Most of all I want to thank my parents. Without their continued support, this degree and in particular this thesis would not have been possible.

# Contents


**1 Introduction** **1**
1.1 The role of deep learning in AI . . . 1
1.2 Research topics . . . 2
1.2.1 Inverting neural networks . . . 2
1.2.2 Restoring adversarial examples . . . 2
1.3 Structure . . . 3

**2 Related Work** **5**
2.1 Network interpretability . . . 5
2.1.1 Network inversion . . . 5
2.1.2 Feature visualization . . . 6
2.2 Adversarial examples . . . 6
2.2.1 Central developments . . . 6
2.2.2 Denoising adversarial perturbations . . . 7
2.2.3 Utilizing feature map information . . . 7

**3 Methods** **9**
3.1 Convolutional networks for image classification . . . 9
3.1.1 The ImageNet recognition challenge . . . 9
3.1.2 Convolutional neural networks . . . 9
3.2 Whitening . . . 10
3.2.1 PCA-whitening . . . 10
3.2.2 ZCA-whitening . . . 10
3.3 The VGG-score . . . 11
3.4 Inversion methods . . . 11
3.4.1 Modular inverting networks . . . 11
3.4.2 Optimization based methods . . . 12
3.5 Fields of Experts priors . . . 13
3.5.1 The Product of Student-t model . . . 13
3.5.2 Score matching . . . 14
3.5.3 Markov random fields . . . 15
3.5.4 Prior variants . . . 15
3.6 Adversarial examples . . . 16
3.6.1 Threat models . . . 16
3.6.2 Adversarial attacks . . . 17
3.6.3 Defending against adversarial attacks . . . 18

**4 Inverting Neural Networks** **19**
4.1 Research questions . . . 19
4.2 Image reconstruction . . . 19
4.2.1 Experiment setup . . . 19
4.2.2 Modular inverting networks . . . 19
4.2.3 Reproduction of Mahendran and Vedaldi's method . . . 20

4.2.4 Fields of Experts priors . . . 20
4.2.5 Results under error metrics . . . 22
4.2.6 Comparison between AlexNet and VGG-16 . . . 23
4.3 Feature map reconstruction . . . 23
4.3.1 Inverting to feature maps . . . 24
4.3.2 Network implicit priors . . . 24
4.4 Discussion . . . 25

**5 Denoising Adversarial Examples** **27**
5.1 Research questions . . . 27
5.2 Experiments . . . 27
5.2.1 Experiment setup . . . 27
5.2.2 Mean filter baseline . . . 27
5.2.3 Denoising oblivious attacks . . . 28
5.2.4 Adaptive attacks . . . 28
5.2.5 Dropout regularizer . . . 30
5.2.6 Feature map transferability . . . 30
5.3 Discussion . . . 31

**6 Conclusion** **33**
6.1 Limitations . . . 33
6.2 Directions for future work . . . 34

**Appendices** **VII**

**A Additional visualizations** **VII**

**B Separable convolutions** **XV**

# Chapter 1

# Introduction

## 1.1 The role of deep learning in AI

Machine learning has enabled us to automate tasks that are not consciously understood. By learning from examples rather than definitions, models can, for instance, distinguish breeds of animals or displayed emotions on the basis of single natural images. This is possible despite the fact that many of the involved concepts are inherently vague and neither the class differences nor the underlying *set of natural images* have clean definitions. Machine learning models which learn from examples alleviate the need for human explanation by discovering structure in the data by themselves. Some models are easily interpretable and can give humans insight into their inductive biases and how they will generalize, while others are themselves not fully understood. Deep neural networks, which have found their way into most application areas of machine learning and are integral to state of the art models in, for instance, the field of computer vision, belong squarely in the latter category. What hierarchical features do they learn? Why do they generalize as well as they typically do? And under what circumstances do they fail? All these are topics of active research.

### A black box

One core question which has received a lot of attention recently concerns how models generalize from their training data to new and unseen inputs. Deep neural networks have empirically proven to generalize very well within certain bounds, but reasons for this behavior are the topic of some debate. A common explanation for generalization is that models compress information about specific inputs, often due to their limited modeling capacity, and in this process preserve those features, which signify relevant class differences, while filtering out contingent properties. Similar explanations have been proposed with regards to neural networks, for instance by Schwartz-Ziv and Tishby [46]. However, Zhang et al. [58] have shown that state of the art image classifiers posses sufficient expressive capacity to simply memorize a full training set and forthcoming work under blind review [1] demonstrates state of the art performance with a fully invertible architecture, which does not compress information by design.

As the use of deep learning models becomes commonplace and finds application in increasingly critical fields, such as autonomous driving or medical diagnostics, the need for a comprehensive understanding of the models is vital to anticipate their limitations and prevent fail-cases. Determining the role that compression plays in neural networks is an essential part of this endeavor.

### An ill-defined problem

As the space of valid datapoints, such as the *set of natural images* is often hard to define, the space of allowed inputs is typically chosen as some well-defined superset. In the case of image classification, for instance, the actual space of possible inputs to the classifier is the hypercube $[0, 255]^{H \times W \times C}$ spanning all possible pixel values over the image dimensions, in which natural images only occupy a small, highly nonlinear subspace. As a result, models trained on samples from the data distribution only receive information about a fraction of the actual input space. Discriminative models, such as most

image classifiers, are based on the assumption that their input belongs to the data distribution and usually exhibit undefined behavior when this assumption is violated [33].

Szegedy et al. [50] have found that image classifiers can be fooled to make arbitrary misclassification through imperceptibly small perturbations to an otherwise correctly classified input. Given the disparity in dimensionality between input and data, it should not come as a surprise, that classifiers can be tricked with artificial inputs. It is however surprising, just how small these changes need to be. On the other hand, these kinds of artificial inputs, known as adversarial examples, do not seem to manifest in the real world with any regularity, as predictions are usually robust to small random perturbations or naturally occurring changes, in lighting or perspective for instance. This suggests, that most adversaries lie outside the naturally occurring data manifold, as vague as this notion may be.

## 1.2 Research topics

This thesis investigates the two topics introduced above, network compression and adversarial examples, with the help of an approximation of the data distribution. First, it inverts image classifiers and relates hidden representations back to image space in order to analyze, how they compress information. In the second part, a defense against adversarial examples is constructed by constraining inputs to the set of natural images. In both cases, the data distribution is learned using a Fields of Experts prior.

### 1.2.1 Inverting neural networks

It is widely believed that deep convolutional neural network (CNN) classifiers owe a lot of their success to the learning of hierarchical feature representations. Shedding light on how intermediate layers reshape and compress information from the input is vital for improving current methods and making models transparent.

Especially in the field of computer vision, where learned visual representations can be easy to interpret, a number of different approaches have been published on this matter and will be discussed in some detail below. A common theme is that, in order to visualize representations at a certain layer in the network, the network's activations in that layer are, in some way or another, related back to the image space at the network input. This is not a trivial task, as the processing steps that transform the input up to that layer are highly non-linear and commonly use non-invertible operations, such as strided convolutions, pooling operations and ReLU non-linearities. For these reasons, the task of reconstructing an input from the activation of a single layer is usually under-constrained and there is a class of inputs, which cause the same or nearly identical activations in the layer.

Here, the project builds on work by Mahendran & Vedaldi [27, 28], who attempt to constrain the reconstruction problem to the space of natural images with a selection of hand-crafted priors. Showing, what parts of an image can be reconstructed from a hidden representation, provides a lower bound intuition on the information still present at that layer. With more accurate reconstructions, this lower bound becomes tighter. By employing trained Fields of Experts priors, this project improves their method in terms of reconstruction quality and by requiring fewer hand-selected parameters during the reconstruction.

### 1.2.2 Restoring adversarial examples

Above, I made the argument, that due to the under-constrained input space of image classifiers and given the typical training methods used on them, even a perfect model would be fooled by inputs, that are not on the data manifold. Of course, perfect classifiers rarely exist and adversarial examples can be created by exploiting badly modeled regions of imperfect models.

In either case, is has become clear that the large input dimensionality in computer vision tasks is partially responsible for the ease, with which adversaries can be created. Following the *curse of dimensionality*, with larger input dimension, the number of possible attack angles as well as the number of training samples needed to represent the input space increase exponentially. This is reflected in the

three datasets, on which adversarial examples are typically investigated: MNIST (greyscale, 28x28), CIFAR-10 (RGB 32x32) and ImageNet (RGB, typically resized to around 224x224) [26, 23, 44]. With an input dimensionality of 784, MNIST is considered the easiest set to defend, whereas ImageNet with around 150,000 input dimensions is the hardest.

Several defenses against adversarial attacks aim to contract the input space in such a way, that the adversary is mapped to a similar space as the original input by smoothing the image or with compression bottlenecks within the classifier structure. In this thesis I propose a regularizer, which maps the input image onto modes in the data manifold, by maximizing its probability under a learned Fields of Experts image prior. Doing so restores a large portion of adversarial examples to their original class labels and I show that modifications such as dropout and priors over feature maps can improve the defenses robustness when targeted by attacks itself.

## 1.3 Structure

The following chapter provides a broad overview of the related literature involving the areas of network interpretability and adversarial examples. After that, chapter 3 introduces the key theoretical concepts and methods this project relies on, such as different network inversion approaches, existing adversarial attacks and defenses and the Fields of Experts model, in detail. The following two chapters present the empirical results obtained in this project and discuss their relevance to the respective research areas. Chapter 4 shows results on the task of network inversion, followed by the treatment of adversarial examples in chapter 5. The implementation of these experiments is publicly available on GitHub[1]. Finally, in the conclusion I address some limitations of this thesis and suggest, which directions seem most promising for further exploration in future work.

[1] `https://github.com/frhrdr/deep_restoration`

# Chapter 2

# Related Work

Since the rising success of neural networks in machine learning since around 2012, empirical investigations of their inner workings have become a lively research area. This thesis builds on a number of these works and, as a foundation, relevant results from the fields of network interpretability and adversarial examples are introduced in this chapter.

## 2.1 Network interpretability

Olah et al. [37] distinguish attribution and feature visualization as the two major themes in current research on neural network interpretability. Attribution methods such as those by Zeiler and Fergus [57] Zintgraf et al. [59] and Kindermans et al. [21] are generally concerned with the task of finding a causal explanation for network activations by relating them back to a given input image. Feature visualization methods on the other hand are not concerned with a specific input image and instead generate inputs which would cause a desired activation in a neuron, channel or layer of the network. The motivation of this approach is, that through generating these samples, one can gain insight about the types of features, that a certain part of the network responds to.

Network inversion falls somewhere in the middle, as the problem is defined in relation to a specific input to be reconstructed, but knowledge of the image is not assumed during the reconstruction. Similar to attribution problems, inversion is based on feature map activations of a specific input, but the reconstruction process itself is closely related to work on feature visualization.

### 2.1.1 Network inversion

There are currently two approaches that deal with the explicit task of inverting CNN image classifiers to reconstruct a specific input.

The first approach by Dosovitskiy and Brox trains an up-convolutional network to reconstruct images based on a given feature map. The first version of this method [8] optimizes the squared reconstruction error and obtains color accurate images. While shallow feature maps are inverted almost perfectly, reconstructions based on deeper feature maps and densely connected layers quickly become blurry to the point that the reconstruction is unrecognizable. In a subsequent publication [7], the authors improve these results by adding two loss terms to the optimization objective. A fixed comparator network, e.g. a part of the classifier, computes feature representations of reconstruction and target images and their mean difference defines a feature loss. In addition, the generator is trained as part of a generative adversarial network [14], in which the discriminator learns to distinguish reconstruction and target image.

The second, optimization-based approach to network inversion was first explored by Erhan et al. [10], who visualized features of deep belief networks by optimizing inputs to maximize specific feature activations. Mahendran and Vedaldi [27, 28] take this approach and instead of maximizing feature activation optimize to match the feature map representation of a target image. In order to improve reconstruction quality and remove artifacts, a number of handcrafted natural image priors are added as regularizers to the optimization. The details of this method are discussed in section 3.4.2.

### 2.1.2 Feature visualization

The optimization-based approach taken by Mahendran and Vedaldi is closely related to works on feature visualization. Previously, Simonyan, Vedaldi and Zisserman had produced visualizations using only $L_2$ regularization and Yosinski et al. [56] experimented with a combination of $L_2$ regularization, Gaussian blur and clipping.

Olah et al. [37] present a state of the art account of this approach, visualizing features of the GoogleNet architecure [51]. They optimize the pre-image without any explicit regularization losses, using only de-correlation and transformation robustness techniques such as jittering (small random translations), scaling and rotation on each optimization step.

Nguyen et al. [34, 35] integrate the work by Dosovitskiy and Brox [7] of encoding the image prior in a trained generator network into feature visualization. The input variable to the generator is then updated using activation maximization, while constrained by a range of priors. As the objective of this work was image generation, Gaussian noise is added to each update step, in order to explore different modes of the data distribution and generate diversity.

Beyond that, Feature visualization has also been popularized by two Image augmentation methods, which build on these techniques. The Inceptionism [31] and DeepDream [32] method by Mordvintsev et al. uses activation maximization to alter existing images. Neural style transfer by Gatys et al. [11] creates combinations of two images by minimizing differences between feature map representations to both of them under different metrics.

## 2.2 Adversarial examples

Since their discovery by Szegedy et al. [50] in 2014, research into adversarial examples has progressed rapidly. Below, I first provide an overview of the most important developments and then discuss a number of approaches that are related to the defense developed in this thesis.

### 2.2.1 Central developments

In [50], Szegedy et al. introduce the concept of adversarial examples. Similar to feature visualization techniques, adversarial examples are generated by updating the network input based on the gradients from the classifier. The attack by Szegedy et al. is framed as a constrained optimization problem to find the minimal perturbation $\boldsymbol{r}$ to an image $\mathbf{x}$, under the conditions that the classifier $F$ assigns the target label $l$ to the perturbed image and that it lies in the range of valid pixel values.

$$\boldsymbol{r}^* = \arg\min_{\boldsymbol{r}}(\|\boldsymbol{r}\|_2) \text{ with } F(\mathbf{x}+\boldsymbol{r}) = l, \mathbf{x}+\boldsymbol{r} \in [0,255]^m \tag{2.1}$$

This problem is optimized using L-BFGS, which results in some computational effort. The choice of similarity metric between image and adversarial example is a point of some controversy. The authors originally use $L_2$ as seen above, but both $L_\infty$ and $L_0$ are used in other methods as well.

Papernot et al. [38] establishes the finding that adversarial examples are transferable both within one type of model and even, to a lesser extent, across different models. This gives rise to the most common type of attack in a *black-box* scenario, where gradients of the attacked model are not available to the attacker. Instead, adversarials are generated based on a similar, separately trained model and transferred to the attacked model.

The attacks introduced by Carlini & Wagner [3, 4] constitute the current state of the art in gradient-based targeted attacks. Their main innovation is to formulate the constraint $F(\mathbf{x}+\boldsymbol{r}) = l$ as part of the optimization objective by defining a function $\ell$ such that $\ell(\mathbf{x}+\boldsymbol{r}) \leq 0$ if and only if $F(\mathbf{x}+\boldsymbol{r}) = l$. Several candidates for $\ell$ are tested, the best performing function is chosen. The box constraint is addressed by defining $r$ as a bounded variable through a tanh function.

$$\ell(\mathbf{x}+\boldsymbol{r}) = \max(\max\{Z(\mathbf{x}+\boldsymbol{r})_i : i \neq l\} - Z(\mathbf{x}')_t, -\kappa) \tag{2.2}$$

$$\boldsymbol{r}_i = \frac{1}{2}(\tanh(w_i)+1) - \mathbf{x}_i \tag{2.3}$$

Here, $Z(x)$ refers to the logits of the classifier. The now unconstrained optimization problem is defined as follows:

$$\boldsymbol{r}^* = \arg\min_{\boldsymbol{r}} ||\boldsymbol{r}||_2 + c \cdot \ell(\mathbf{x} + \boldsymbol{r}) \tag{2.4}$$

Weighting constant $c$ is chosen empirically and the objective is minimized using the Adam optimizer plus some added details such as random restarts. Notably, they do ensure that all their results are quantized to the common 256 discrete values per channel, which was not taken into account by previous attacks. With this attack, the authors overcome a collection of previously published defenses, including some with non-differentiable and stochastic components. Non-differentiable defenses are defeated by transferring attacks from similarly designed differentiable models and stochastic methods such as dropout are attacked by creating and ensemble representing the range of possible states and transferring attacks that fool every member of the ensemble.

While defense techniques usually generalize to different classifiers, the current state oder adversarial examples research suffers from a lack of testing conventions. Partially owed to the fields rapid development, publications vary in the datasets, classifiers and attacks they use. Libraries supplying benchmark attacks, such as Cleverhans [41] or Foolbox [42], with the associated robust vision benchmark[1] present attempts to create reliable standards. However, some of the lack of comparability is potentially inherent in the adversarial nature of the subject matter. Individual defenses, which may seem to do well under generic circumstances can often be overcome by specifically modified attacks, as demonstrated by Carlini and Wagner [3, 4].

### 2.2.2 Denoising adversarial perturbations

There are several approaches which attempt to overcome adversarial perturbation by smoothing or denoising them. The goal of these methods is always to manipulate the input in such a way that both natural images and adversarial examples are classified correctly afterwards.

Li & Li [25] use a simple $3 \times 3$ mean filter to smooth the input and report that this recovers around 70% of original labels when used against L-BFGS attacks on ImageNet. In a similar experiment, Dziugaite et al. [9] use JPG compression to counter fast gradient sign attacks. They report, that small perturbations can be counteracted, but when the size of the perturbations is increased, the compression does nothing.

Guo et al [16] test a defense based on pixel dropout, followed by smoothing based on total variance minimization. They use a ResNet architecture on ImageNet with a variety of attacks, including Deepfool [30] and the Carlini & Wagner $L_2$ attack [3] and report restoration performance of 60% on white-box attacks and 90% on black-box attacks. Given the non-differentiable nature of dropout, adaptive attacks are not tested. How an Ensemble style attack, as in [4] would perform here, is not explored.

Gu et al [15] explicitly look at adversarial examples as a denoising problem by training Denoising Autoencoders (DAE) on MNIST, which pre-process the network input. The denoised output is then fed into the classifier. They show that DAEs can restore over 90% of adversarial perturbations. When the DAEs are additionally trained on adversarial examples with original images as targets, the method performs slightly better. They observe that using DAEs in this manner makes the model more susceptible to adaptive attacks, which require even smaller perturbations to be successful.

### 2.2.3 Utilizing feature map information

Metzen et al. [29] attach detector networks to various layers of a ResNet classifier and train them to discriminate between normal and adversarial inputs. They investigate fast and iterated gradient sign method as well as Deepfool attacks and the best detector obtains an accuracy of about 78% for the hardest attack (DeepFool with $L_2$ constraint). Notably, the classifier at image level performs worse, which suggests that feature representations add value to this task. For the adaptive scenario where the attack has access to both classifier and detector gradients, they propose a dynamic training

[1] https://robust.vision/benchmark/

method, which can improve robustness against such attacks. Carlini and Wagner [4] report that this method does not improve resistance against adaptive attacks in a white-box setting and requires a 34% increase in perturbation in a black-box setting, when using their C&W $L_2$ attack. They also mention that attempts to use an ensemble of detector networks did not improve the defense.

Li and Li [25] train support vector machine (SVM) classifiers on a number of layers, each taking the principal component analysis of the feature map as input. The model is set up as a *cascade classifier*, which means that the input is classified as adversarial, if at least one SVM recognizes the input as adversarial. Training with L-BFGS attacks they report 96% accuracy with 0% false positive rate. This model also fails under state of the art attacks [4], even in the oblivious case. On the harder to detect C&W $L_2$ attack, a single detector obtains a true positive rate of 62% on MNIST and a false positive rate of 37%, which under the cascade model leads to rejection of over 90% of valid inputs

In a separate work exploring the relations between network inversion and adversarial examples, Sabour et al [45] look at the task of matching a feature map representation in the classifier instead of optimizing for a label. These representations are then visualized using the pre-image method by Mahendran and Vedaldi.

# Chapter 3

# Methods

## 3.1 Convolutional networks for image classification

### 3.1.1 The ImageNet recognition challenge

The ImageNet Large Scale Visual Recognition Challenge was an annual computer vision challenge hosted from 2010 to 2017. Over the years, the challenge has accepted submissions on a variety of computer vision tasks, but in the following, only the 2012 image classification challenge will be relevant. The 2012 image classification challenge is based on a subset of the ImageNet dataset, containing 1,281,167 labeled training images belonging to 1000 classes. In addition, a validation set is made available which consists of 50,000 images with an even distribution of 50 per class.

In this task, models are evaluated based on their top-5 accuracy, which means, that the prediction of a model is rated as correct, if the ground truth label is among the 5 labels the model predicts to be most likely. Despite efforts to clean it, some images in the dataset contained multiple or ambiguous classes and for image recognition models in 2012, top-5 accuracy presented a hard challenge. For these reasons it was chosen over the perhaps more intuitive top-1 accuracy metric.

### 3.1.2 Convolutional neural networks

The contest in 2012 marked a turning point in the field, when a deep convolutional neural network (CNN) won the image recognition challenge for the first time. Since then, deep neural networks have won the challenge every year.

#### AlexNet

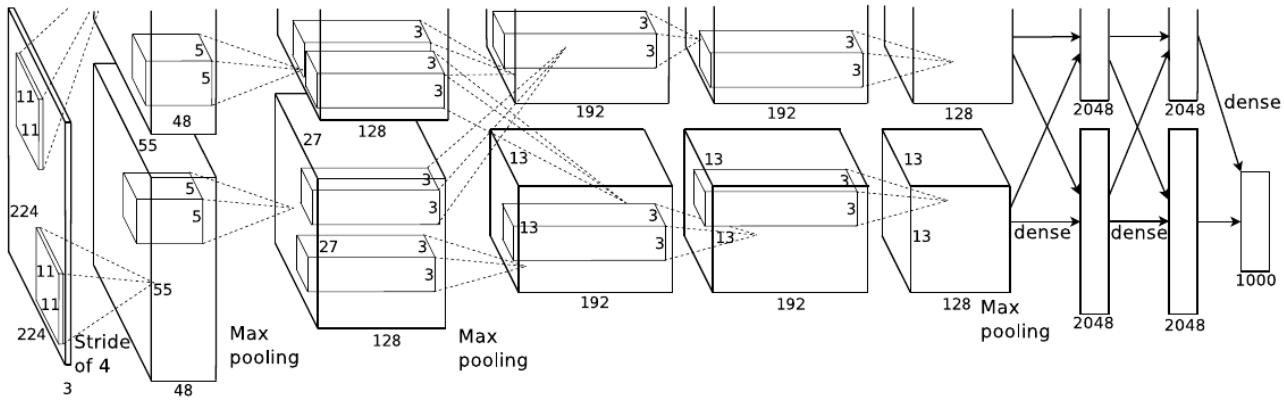



**Figure 3.1:** AlexNet Architecture (from [24])

The winning deep convolutional network architecture of 2012 was developed by Krizhevsky et al. [24], commonly referred to as AlexNet contains 60 million trained parameters in 5 convolutional and 3 densely connected layers. ReLU activation is used throughout the network and as shown in

| | Block 1 | Block 2 | Block 3 | Block 4 | Block 5 | Block 6 | Block 7 | Block 8 |
|---|---|---|---|---|---|---|---|---|
| AlexNet | conv1/lin<br>conv1/relu<br>lrn1<br>pool1 | conv2/lin<br>conv2/relu<br>lrn2<br>pool2 | conv3/lin<br>conv3/relu | conv4/lin<br>conv4/relu | conv5/lin<br>conv5/relu<br>pool5 | fc6/lin<br>fc6/relu | fc7/lin<br>fc7/relu | fc8/lin<br>fc8/relu |
| channels | 96 | 256 | 384 | 384 | 256 | | | |
| VGG-16 | conv1_1/lin<br>conv1_1/relu<br>conv1_2/lin<br>conv1_2/relu<br>pool1 | conv2_1/lin<br>conv2_1/relu<br>conv2_2/lin<br>conv2_2/relu<br>pool2 | conv3_1/lin<br>conv3_1/relu<br>conv3_2/lin<br>conv3_2/relu<br>conv3_3/lin<br>conv3_3/relu<br>pool3 | conv4_1/lin<br>conv4_1/relu<br>conv4_2/lin<br>conv4_2/relu<br>conv4_3/lin<br>conv4_3/relu<br>pool4 | conv5_1/lin<br>conv5_1/relu<br>conv5_2/lin<br>conv5_2/relu<br>conv5_3/lin<br>conv5_3/relu<br>pool5 | fc6/lin<br>fc6/relu | fc7/lin<br>fc7/relu | fc8/lin<br>fc8/relu |
| channels | 64 | 128 | 256 | 512 | 512 | | | |

**Figure 3.2:** Overview of the network layers in AlexNet and VGG-16

figure 3.1, max-pooling operations take place before the second, third and sixth layer. In addition, the architecture contains two *local response normalization* (lrn) operations before the first and second pooling.

#### VGG-16

In 2014, Simonyan and Zisserman from the Visual Geometry Group at Oxford designed a set of larger, but simplified architectures known as VGG nets [49]. The networks consist of only of convolutions with 3x3 filters, ReLU activations and 2x2 max pooling operations as listed in figure 4.1. Larger filter sizes are effectively generated through stacking of multiple convolutional layers. This project is mostly concerned with the 16 layer version of this architecture, VGG-16, which has a total of 138 million parameters. Unlike AlexNet, VGG-16 does not start with a strided convolution, leading to much larger feature maps in the early layers of the network, despite the lower number of channels.

## 3.2 Whitening

Whitening is a linear transformation on a set of n-dimensional data-points, which decorrelates their dimensions and scales each dimension to unit variance, such that the covariance matrix of the set is the identity matrix $\mathbf{I}_n$. There are multiple transformations which satisfy this condition and different whitening methods introduce additional constraints which uniquely define it. Two common ones are explained here.

### 3.2.1 PCA-whitening

The first whitening method is based on principal component analysis, which already decorrelates the data. Given a data set $\mathbf{X}$ with mean $\boldsymbol{\mu}$ and sample covariance $\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N} \mathbf{x}_i - \boldsymbol{\mu}$, the Eigendecomposition $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ yields the Eigenvectors $\mathbf{U}$ which decorrelate the data. The transformed data $\mathbf{U}^\top\mathbf{X}$ has diagonal covariance matrix $\boldsymbol{\Lambda}$. The one additional step of dividing each dimension by the corresponding eigenvalue completes the whitening transform, which is summarized in equation 3.1. A useful property of PCA whitening is, that it allows for efficient dimensionality reduction, by omitting the dimensions with smallest eigenvalues after the transform.

$$\mathbf{x} \mapsto \boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top(\mathbf{x} - \boldsymbol{\mu}) \tag{3.1}$$

### 3.2.2 ZCA-whitening

Zero-phase component analysis, also known as Mahalanobis whitening is related to PCA-whitening, and can be understood as the PCA-whitened data, transformed back into the original coordinate system by the rotation $\mathbf{U}$. Adding the final rotation to the transformation matrix makes it equivalent

to the square root of the Moore-Penrose inverse of the covariance: $\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top = \mathbf{\Sigma}^{-\frac{1}{2}}$. Therefore, the transformation can be written as shown in equation 3.2. It can be shown, that this is the whitening transform with minimal euclidean distance from the original data [20], which makes it a good choice for preserving localized structure in image data.

$$\mathbf{x} \mapsto \mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{x} - \boldsymbol{\mu}) \tag{3.2}$$

## 3.3 The VGG-score

The VGG-score proposed by Ustyuzhaninov et al. [54] and defined in equation 3.3 computes a similarity metric between a target image $\mathbf{x}$ and reconstruction $\mathbf{y}$. It takes the feature maps of the two images in the VGG-19 architecture [49] with average instead of max-pooling layers and accumulates the differences in their Gram matrices $G^\ell$ over layers $\ell$. This way, it measures the extent, to which the images share the same types of higher order features. The chosen layers are the five convolutional layers which follow pooling operations in the network.

$$E(\mathbf{y}) = \frac{1}{5}\sum_{\ell=1}^{5} E^\ell(\mathbf{y}) \tag{3.3}$$

$$E^\ell(\mathbf{y}) = \frac{1}{\sum_{i,j=1}^{N} G^\ell_{ij}(\mathbf{x})^2} \sum_{i,j=1}^{N} \left( G^\ell_{ij}(\mathbf{x}) - G^\ell_{ij}(\mathbf{y}) \right)^2 \tag{3.4}$$

## 3.4 Inversion methods

For the task of network visualization I have studied two methods for projecting feature map representations of a network back into image space. The first approach follows prior work by Dosovitskiy and Brox [8] and trains a CNN model with transpose convolutions, which takes a feature map as input and outputs a reconstruction of the image. I compare this to the optimization based method by Mahendran and Vedaldi (M&V) [28], which optimizes a pre-image so that its feature map representations match those of a target image, yielding qualitatively very different results. I improve on the results by Mahendran and Vedaldi by using Fields of Experts priors in place of their handcrafted image priors.

### 3.4.1 Modular inverting networks

Dosovitskiy and Brox [8] have shown that some information from the input image $\mathbf{x}$ can be recovered by training a neural network with transpose convolutions $f$, which takes a feature map $\phi(\mathbf{x})$ as input and outputs a reconstruction of the image $f(\phi(\mathbf{x}))$. Finding the $\hat{f}$ which minimizes the expected mean squared reconstruction error $\mathbb{E}_{\mathbf{x},\phi}\|\mathbf{x} - f(\phi(\mathbf{x}))\|^2$ yields the conditional expectation.

$$\hat{f}(\phi_0) = \mathbb{E}_{\mathbf{x}}[\mathbf{x}|\phi = \phi_0] \tag{3.5}$$

One shortcoming of their model is, that it does not restore the original image dimensions of the classifier. This makes it ill-suited for a comparison to other reconstruction methods and I propose a modified version, which fixes this drawback.

In the modular approach I have taken, small modules are trained to invert a part of the network, such as a pooling or convolution operation and a non-linearity. Each module consists of a transpose convolution followed by a ReLU and a convolution layer. The modules can then be stacked as shown in figure 3.3 and trained to invert a given portion of the target classifier. First, each module is trained individually to minimize the mean squared reconstruction error, which makes them reusable for different tasks. Then the stacked model is trained for a short period of time, optimizing the reconstruction error of the final module, which prevents parts of the errors in each module from adding up and thus improves the reconstruction quality.

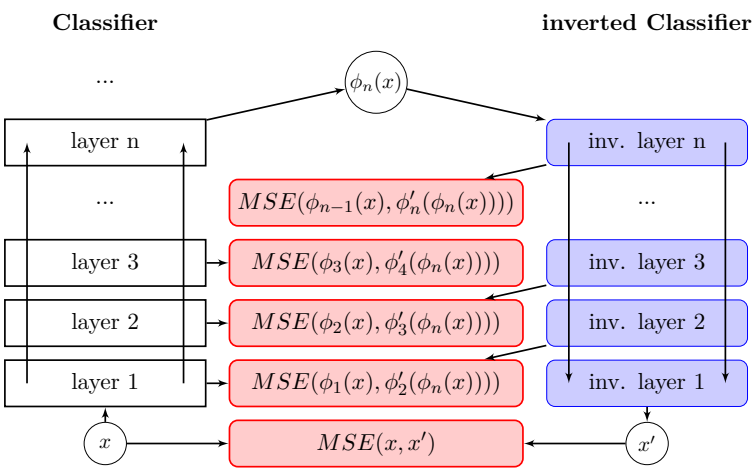



**Figure 3.3:** Modular inverting network architecture

### 3.4.2 Optimization based methods

The central approach to network visualization in this project is based on work by Mahendran & Vedaldi [27, 28]. Their method, illustrated in figure 3.4, optimizes a *pre-image* variable to match one of its feature map representations with that of an image, while constraining the pre-image additionally by a natural image prior.

Take a target image $x$, a pre-image variable $x'$ of equal dimensions, and the function $\phi_n$, defined by a neural network, which maps an image to the feature map representation at layer $n$. A feature map representation $\phi_0$ is mapped back to image space by optimizing the error term

$$\arg\min_{x'} \mathcal{L}(\phi_0, \phi(x')) + \lambda R(x') \tag{3.6}$$

where $\mathcal{L}$ is a loss function comparing the two feature map representations and $R$ is a regularizer constraining the values, the pre-image can take Typically $\phi_0 = \phi(x)$ for some target image $x$, but in principle, the feature map can also be constructed, e.g. by setting a subset of the channels to 0.

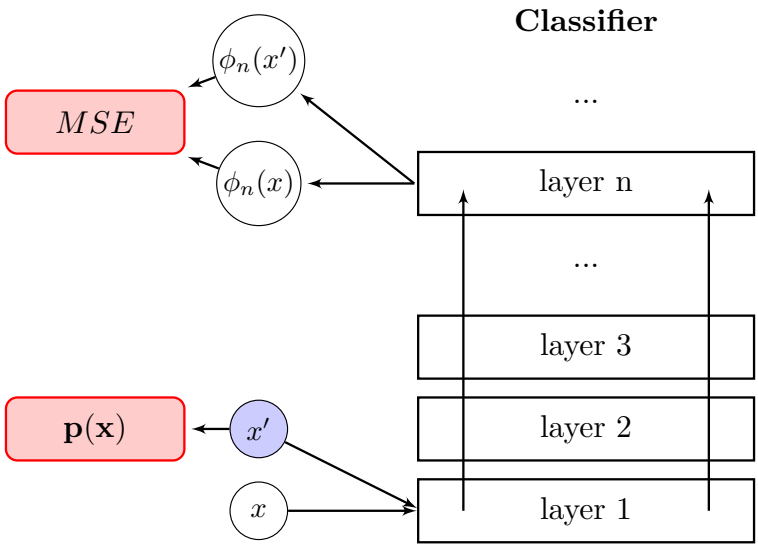



**Figure 3.4:** Illustration of optimization-based inversion

#### Mahendran & Vedaldi's work and handcrafted image priors

In [27], the authors tackle the above-mentioned optimization problem, using a normalized mean squared error loss on the feature maps as $\mathcal{L}$ and combine two image priors as a regularizer. The normalized mean squared error takes the form:

$$||\phi(\sigma\mathbf{x}) - \phi_0||_2^2 / ||\phi_0||_2^2 \tag{3.7}$$

where $\phi_0$ is the feature map representation of the target image and $\phi(\sigma\mathbf{x})$ is the representation of the re-scaled pre-image $\mathbf{x}$. The scaling factor $\sigma = 2.7098\text{e}{+4}$ is chosen as the average euclidean norm of natural images in the training set to the effect that $\mathbf{x}$ has roughly unitary Euclidean norm. Dividing the mean squared error by the squared Euclidean norm of the target image confines errors near an optimum to the interval $[0, 1]$. The authors argue that this predictability can be convenient when considering the scaling of the priors. The authors constrain the optimization of the feature level error with two hand-designed priors. The first one is an $\alpha$-norm prior, defined as

$$\mathcal{R}_\alpha(\mathbf{x}) = ||\mathbf{x} - \text{mean}(\mathbf{x})||_\alpha^\alpha \tag{3.8}$$

It is meant to limit the overall range of pixel values. Results are reported with $\alpha = 6$. In addition, a total variation prior is used, which penalizes the difference between neighboring pixels and thus encourages smoothness in the pre-image. It takes the following form, with $\beta = 2$ being chosen in most experiments:

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left( (x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2 \right)^{\frac{\beta}{2}} \tag{3.9}$$

In their follow-up publication [28], the authors maintain the MSE loss and total variation prior, but refine the range prior to

$$N_\alpha(\mathbf{x}) = \sum_{i,j} \left( \sum_k \mathbf{x}_{ijk}^2 \right)^{\frac{\alpha}{2}} \tag{3.10}$$

where $i, j$ are pixel positions and $k$ denotes the channel. The authors argue that this version resolves a problem of color bias introduced by the alpha norm prior, because unlike the previous one, it is isotropic in RGB space. They couple this soft constraint with a hard one, which enforces $\sqrt{\sum_k \mathbf{x}_{ijk}^2} \leq B_+$ for each image position $(i, j)$. They also develop the weighting of the different parameters further and adopt an additional regularization method called jittering, following Mordvintsev et al. [31]. Jittering makes small shifts to the pre-image on each iteration and thus ensures that the feature map representation of the pre-image remains stable under these translations. The method thus enforces another property observed in natural images and prevents undesirable optima, which leads to crisper images overall.

## 3.5 Fields of Experts priors

Fields of Experts [43] are learned priors with more expressive power and less dependency on handpicked parameters than the heuristics used by Mahendran and Vedaldi. The Fields of Experts (FoE) model by Roth and Black is defined as a Markov random field, which is composed of Product of Student-t Experts by Welling et al. [55] for clique potentials. In this way it aggregates scores from local priors over image patches into an unnormalized probability estimate for the whole image. This section lays out the model in detail, by first explaining the Product of Student-t model, then going into the method used for training the model, and finally deriving, how it is used in a Markov random field.

### 3.5.1 The Product of Student-t model

The Product of Student-t (PoT) model by Welling et al. [55] can be trained to learn a sparse representation over image patches $\mathbf{x}_k$. Each of the $N$ *experts* extracts a feature of the patch through the filter $\mathbf{w}_i$ and models the feature as a random variable with a Student-t distribution.

$$p(\mathbf{x}) = \frac{1}{Z(\Theta)} \prod_{i=1}^{N} \left( 1 + \frac{1}{2} (\mathbf{w}_i^T \mathbf{x})^2 \right)^{-\alpha_i} \tag{3.11}$$

where $\|\mathbf{w}_i\| = 1$ and $\alpha_i > 0$ for all $i$. The norms of $\mathbf{w}_i$ are constrained, so filters do not decay to 0. As this makes the model unable to adapt to different scales found in the features, the input data must be whitened. This model can be viewed as an energy-based generalization for overcomplete independent component analysis (ICA) as discussed in detail by Teh et al. [52]. Where in regular ICA the marginal distribution over the source vectors is independent by assumption, this form of overcomplete ICA does not guarantee that all combinations of source vectors correspond to a valid observation, as the observation is overdetermined. This modeling decision makes it hard to sample from the overcomplete model, but preserves another useful property. Namely, the conditional distribution of sources given a sample remains deterministic and independent and thus the probability density over valid observations is well-defined. Given that there seems to be no compelling reason for assuming that image patches are generated from a set of independent sources, dropping the marginal independence assumption is justifiable.

Overcomplete ICA models have been shown to exhibit a number desirable qualities. The filters learned by such a model form an overcomplete basis of the input space, which is sparse and in the case of natural images typically takes the form of wavelets, which are intuitive to understand. The model makes the filters independent given the input, which leads to a very transparent model. In addition, over-completeness allows for dependencies between filters to be modeled, which makes the model more expressive than regular ICA.

### 3.5.2 Score matching

In [19], Hyvarinen introduced the score matching estimator for unnormalized probability distributions. Given a model probability distribution $p$ with parameters $\boldsymbol{\theta}$ over variables $\boldsymbol{\xi}$,

$$p(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} q(\boldsymbol{\xi};\boldsymbol{\theta}), \tag{3.12}$$

score matching provides an estimator based on the derivative of the logarithm of $p$ with respect to $\boldsymbol{\xi}$, which is referred to as the score function $\boldsymbol{\psi}$ of $p$ in statistics.

$$\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\xi}} \log p(\boldsymbol{\xi};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\xi}} \log q(\boldsymbol{\xi};\boldsymbol{\theta}) \tag{3.13}$$

The estimate seeks to approximate the true data distribution $p_{\mathbf{x}}(\boldsymbol{\xi})$ by minimizing the expected difference between data and model score functions:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \int_{\boldsymbol{\xi}\in\mathbb{R}^n} p_{\mathbf{x}}(\boldsymbol{\xi}) \|\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta}) - \boldsymbol{\psi}_{\mathbf{x}}(\boldsymbol{\xi})\|^2 d\boldsymbol{\xi} \tag{3.14}$$

Hyvarinen shows that under some weak conditions[1] the objective function $J$ can be re-written and computed without an estimate of the data score function. This makes the computation much more efficient and gives rise to the sample objective function $\tilde{J}$ for a number of $T$ observations:

$$J(\boldsymbol{\theta}) = \int_{\boldsymbol{\xi}\in\mathbb{R}^n} p_{\mathbf{x}}(\boldsymbol{\xi}) \sum_{i=1}^{N} \left[ \partial_i \psi_i(\boldsymbol{\xi};\boldsymbol{\theta}) + \frac{1}{2}\psi_i(\boldsymbol{\xi};\boldsymbol{\theta})^2 \right] d\boldsymbol{\xi} + const \tag{3.15}$$

$$\tilde{J}(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \left[ \partial_i \psi_i(\mathbf{x}(t);\boldsymbol{\theta}) + \frac{1}{2}\psi_i(\mathbf{x}(t);\boldsymbol{\theta})^2 \right] + const \tag{3.16}$$

In general, $J$ is non-convex and minima can be found via gradient descent optimization. For the given case of a Student-t Product of Experts prior, the objective function resolves to the following formulas.

$$\boldsymbol{\psi}(\mathbf{x};\mathbf{w},\alpha) = \sum_{k=1}^{m} \alpha_k \mathbf{w}_k g(\mathbf{w}_k^\top \mathbf{x}) \tag{3.17}$$

[1] $p_{\mathbf{x}}(\boldsymbol{\xi})$ is differentiable, $\mathbb{E}_{\mathbf{x}}[\|\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta})\|^2]$ and $\mathbb{E}_{\mathbf{x}}[\|\boldsymbol{\psi}_{\mathbf{x}}(\boldsymbol{\xi})\|^2]$ are finite for any $\boldsymbol{\theta}$, and $p_{\mathbf{x}}(\boldsymbol{\xi})\boldsymbol{\psi}(\boldsymbol{\xi};\boldsymbol{\theta})$ goes to 0 for any $\boldsymbol{\theta}$ when $\|\boldsymbol{\xi}\| \to \infty$

$$\tilde{J} = \sum_{k=1}^{m} \alpha_k \frac{1}{T} \sum_{t=1}^{T} g'(\mathbf{w}_k^\top \mathbf{x}_{(t)}) + \frac{1}{2} \sum_{j,k=1}^{m} \alpha_j \alpha_k \mathbf{w}_j^\top \mathbf{w}_k \frac{1}{T} \sum_{t=1}^{T} g(\mathbf{w}_k^\top \mathbf{x}_{(t)}) g(\mathbf{w}_j^\top \mathbf{x}_{(t)}) \tag{3.18}$$

where $g(s) = -\frac{2s}{s^2+2}$ and $g'(s) = \frac{2(s^2-2)}{(s^2+2)^2}$ are the first and second derivative of the log density function $G(s) = -\log(1 + \frac{1}{2}s^2)$.

### 3.5.3 Markov random fields

A Markov random field defines the independence relations of a set of random variables $\boldsymbol{X}$ through a graph $G = (\boldsymbol{X}, E)$, where $\boldsymbol{X}$ constitutes the set of vertices connected by the edge set $E$, which denotes dependence relations. The graph defines local and global Markov properties between variables. The local Markov property states that a random variable $\boldsymbol{x}_u \in \boldsymbol{X}$ is conditionally independent of other variables $\boldsymbol{x}_u \in \boldsymbol{X}$ given the set of its immediate neighbors $\boldsymbol{X}_{N(u)} = \{\boldsymbol{x}_v | \{\boldsymbol{x}_v, \boldsymbol{x}_u\} \in E\}$.

$$\boldsymbol{x}_u \perp\!\!\!\perp \boldsymbol{x}_v | \boldsymbol{X}_{N(u)} \tag{3.19}$$

Further, given sets of variables $\boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{X}_S \subset \boldsymbol{X}$, such that all paths from $\boldsymbol{X}_A$ to $\boldsymbol{X}_B$ on $G$ go through $\boldsymbol{X}_S$, the global Markov property holds if

$$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B | \boldsymbol{X}_S \tag{3.20}$$

Let $C$ be the set of maximal cliques of $G$ and $\phi_c(c), c \in C$ a function called the clique potential. According to the Hammersley-Clifford theorem [17], the joint probability density function $p(\boldsymbol{X})$ can be factorized into a Gibbs distribution based on the clique potentials of $G$.

$$p(\boldsymbol{X}) = \frac{1}{Z(\Theta)} \exp\Big( - \sum_{c \in C} \phi_c(c) \Big) \tag{3.21}$$

In the Fields of Experts model, $\boldsymbol{X}$ is taken to be an image or a feature map, with pixels at every height, width and channel constituting random variables. All clique potentials $\phi_c$ are expressed by the same Product of Student-t model. Accordingly, the edge set $E$ is selected such that the set of maximal cliques $C$ constitutes all the patches of the size, which the PoT model is trained on.

Leaving the graph-based notation behind and enumerating the set of images patches $\{\mathbf{x}_1, \mathbf{x}_2 ... \mathbf{x}_K\}$, the Fields of Experts is written as follows

$$p(\mathbf{x}) = \frac{1}{Z(\Theta)} \prod_{k}^{K} \prod_{i=1}^{N} \Big( 1 + \frac{1}{2} (\mathbf{w}_i^\top \mathbf{x}_k)^2 \Big)^{-\alpha_i} \tag{3.22}$$

### 3.5.4 Prior variants

In the Product of Student-t model, the size of each filter is equal to the the size of the patches it is trained on $(H \times W \times C)$, with $H$ and $W$ being the height and width of the image patches the prior is trained on and $C$ being the number of channels Given that the learned representation should be overcomplete, the number of filters must exceed the number of variables in each filter, which makes the filter bank $\mathbf{W}$ scale quadratically in the patch size. For patches of feature maps with between 50 and several hundred channels, $\mathbf{W}$ grows too large to be feasible, due to this scaling factor. In the following I describe two modified versions of the PoT model, which attempt to avoid the quadratic scaling factor.

#### Mixing clique potentials

The first option for making the prior scale more efficiently is to introduce additional independence assumptions to the Markov random field the FoE prior is based on. This is expressed by erasing edges and reduces the size of the maximal cliques the graph. Erasing all edges from the graph which link different channels introduces an independence assumption between channels and allows for modeling

the MRF with $C$ channel-priors of size $(H \times W)$. Assuming, each of the channel prior is over-complete by a factor $\beta$, the collective filter bank is of size $C \times \beta(H \times W)^2$, scaling linearly in $C$.

Because complete independence of channels is an unreasonably strong assumption, some edges between channels should be preserved, e.g. between values at the same height and width position of the feature map. This leads to an additional maximal clique type of size $(1 \times 1 \times C)$. While the overcomplete filter bank for this clique potential is of size $\beta C^2$ an so does scale quadratically in $C$, it is likely small enough to be feasible.

Under these considerations, the resulting FoE model takes the following form, where $\mathbf{w}_c$ is the filter bank for channel $c$ and $\mathbf{w}'$ is the filter bank for the inter-channel potential.

$$p(\mathbf{x}) = \frac{1}{Z(\Theta)} \left( \prod_{k=1}^{K'} \prod_{i=1}^{N'} \left(1 + \frac{1}{2}(\mathbf{w}'^{\top}_i \mathbf{x}_k)^2\right)^{-\alpha'_i} \right) \prod_{k=1}^{K} \prod_{c=1}^{C} \prod_{i=1}^{N} \left(1 + \frac{1}{2}(\mathbf{w}^{\top}_{ci} \mathbf{x}_{kc})^2\right)^{-\alpha_{ci}} \tag{3.23}$$

The problem with this approach lies in the fact that the product of these clique potentials is only proportional to the field potential when they are weighted correctly, as score matching trains each clique potential only to be accurate up to a constant factor. How this weighting can be optimized is an open question. Hyper-parameter optimization methods may work well enough to provide good results, but most likely involve substantial computational cost.

#### Separable FoE priors

With the right preprocessing, FoE priors can be efficiently implemented as convolutions. For this reason I decided to experiment with *depthwise separable convolutions*, a method which had proven effective at reducing computational costs in CNNs. Originally based on work by Sifre and Mallat [47] and more recently employed by Chollet [6], this method replaces a regular convolution with two separate operations. First a channelwise convolution takes place with a small number $M$ of filters per channel, then the resulting feature map is convolved with a number $K$ of column-filters of height and width 1, which compute weighted sums of the channelwise results and output the final feature map, as shown in figure B.1 in the appendix. Mathematically speaking, for a given input patch $\mathbf{x}$ of shape $[H, W, C]$ and filter matrix $\mathbf{W}$ of shape $[H, W, C, K]$, a regular convolution operation simply flattens the first three dimensions and computes $\mathbf{W}^{\top}\mathbf{x}$. A separable convolution operation replaces $\mathbf{W}$ with a depth-wise filter $\mathbf{d}$ of shape $[M, H, W, C]$ and a point-wise filter $\mathbf{p}$ of shape $[M, K, C]$, defining the operation as follows:

$$\text{sepconv}_{\mathbf{p},\mathbf{d}}(\mathbf{x}) = \sum_{i}^{M} \mathbf{p}_i (\sum_{h}^{H} \sum_{w}^{W} \mathbf{x}_{h,w} \odot \mathbf{d}_{i,h,w}) \tag{3.24}$$

Contrary to the previous variants, the separable prior utilizes structure information within a given patch, namely the distinction between different channels. As PCA-whitening would eliminate this structure, ZCA-whitening is used instead. However, this also means that dimensionality reduction is not available in this step. While this approach has been implemented and generally works on image level, it fails to provide any gains in efficiency. Training the model with score matching still requires explicit computation of the implicitly defined filters. Attempts to formulate the training objective in a way that eliminates the quadratic scaling in this computation have been unsuccessful so far, but the possibility that such an expression exists has not been excluded either.

## 3.6 Adversarial examples

### 3.6.1 Threat models

The literature on adversarial examples distinguishes between several different kinds of attack scenarios, also referred to as threat models. *White-box* attacks assume that the attacker has full access to gradient information generated by the attacked model, which makes them the hardest attacks to defend against. *Black-box* attacks, which assume that information about the model is only in part accessible are less

powerful as a result. Attacks can have various objectives with the most common ones aiming to change only the class label computed as most probable by the classifier. These can be *targeted*, in which case the objective is to change the classification to a given target label, or *untargeted*, in which the new label is arbitrary. The following experiments are exclusively concerned with untargeted white-box attacks, as defenses which prove effective in the scenario will generalize to others. The experiments test both *oblivious* attacks, which aim only at the classifier, and *adaptive* attacks, which take the defense into account.

### 3.6.2 Adversarial attacks

The objective of an adversarial attack on a classifier $f$, as it is usually framed, is to find a minimal perturbation $\boldsymbol{r}$ to an image $\mathbf{x}$ with assigned label $f(\mathbf{x}) = l$, such that the modified image receives a different target label $f(\mathbf{x} + \boldsymbol{r}) = l'$. Two different attacks were used in this project and are explained in the following.

#### The fast gradient sign method

Goodfellow et al. [13] introduces the fast gradient sign method (FGSM) as an efficient alternative to the L-BFGS based attacks initially put forward by Szegedy et al. [50]. This attack has been widely used in the adversarial examples literature, but has become less relevant with the development of more powerful attacks. With $J(\boldsymbol{\theta}, \mathbf{x}, l)$ as the cost function of the classifier with respect to weights $\boldsymbol{\theta}$, image $\mathbf{x}$ and label $l$, a small value $\epsilon$ is chosen to generate the perturbation $\boldsymbol{r}$:

$$\boldsymbol{r}^* = \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, l)) \tag{3.25}$$

By taking a single gradient step toward the target class, this attack relies heavily on the local linearity of the classifier with respect to $\mathbf{x}$. Goodfellow makes the argument that local linearity may actually be the one of the reasons, why models are so consistently vulnerable to adversarial perturbation.

#### The Deepfool Method

The deepfool attack method proposed by Moosavi-Dezfooli et al. [30] presents an iterative untargeted method which greedily searches for the closest classification boundary to another class. Given a classifier which computes the current class as $\hat{k}(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$, the method computes linear distance estimates for all classes $k \neq \hat{k}(\boldsymbol{k}_0)$ on each iteration, based on the difference in class probabilities $f'_k$ and and the difference between their gradients $\boldsymbol{w}'_k$.

$$f'_k = f_k(\mathbf{x}_i) - f_{\hat{k}(\mathbf{x}_0)}(\mathbf{x}_i) \tag{3.26}$$

$$\boldsymbol{w}'_k = \nabla f_k(\mathbf{x}_i) - \nabla f_{\hat{k}(\mathbf{x}_0)}(\mathbf{x}_i) \tag{3.27}$$

On each iteration $i$, the closest class $\hat{l}_i$ is chosen using a linear estimate and the input is updated to move to lie on the estimated boundary. This process is iterated until $\mathbf{x}_i$ is misclassified. The update to $\mathbf{x}_i$ is multiplied by a small factor $\eta = 1.02$, so the decision boundary is not just approached but crossed. To reduce computational costs, the set of considered target classes can be sub-sampled at the cost of returning sub-optimal solutions.

$$\hat{l}_i = \arg\min_{k \neq \hat{k}(\mathbf{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2} \tag{3.28}$$

$$\boldsymbol{r}_i = \frac{|f'_{\hat{l}_i}|}{\|\boldsymbol{w}'_{\hat{l}_i}\|_2^2} \boldsymbol{w}'_{\hat{l}_i} \tag{3.29}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \eta\, \boldsymbol{r}_i \tag{3.30}$$

### 3.6.3 Defending against adversarial attacks

Countermeasures against adversarial examples can take different forms. The common two goals of a defense is to either detect adversarial inputs or to neutralize the adversarial and restore the original class label. This section introduces a defense from the latter category and the measures to quantify its success.

#### Robustness measures

The quality of a defense is typically measured in two ways. The first test looks at oblivious adversaries, which fool a model, when the defense is not implemented, and measures, how many of these are detected or restored, when adding the defense. Ideally, this result is combined with a test on natural images to quantify the type 1 error (a false positive, i.e. changing the label of natural inputs) caused by the defense, so it can be weighed against the type 2 error of falling for an adversarial attack. Many defenses yield good results under this measure with detection and restoration rates over 80%, while negatively affecting less than 10% of natural images [15, 16, 29, 25].

The second measure is significantly more challenging and arguably more important. Here, adversarials are generated, targeting both the classifier and the defense. Fooling the model in this setting is in principle always possible and is trivially achieved by selecting a perturbation, which maps the source image to one from the target class. The success of a defense is typically measured by the increase in perturbation necessary to fool the defense, compared to the unprotected model. As this evaluation is conducted using specific attacks, it can also happen, that these attacks fail to find adversarials altogether. This is commonly reported as a success of the defense, but can be very specific to the type of attack and how it was executed. As attack types become more diverse, this kind of result may become less relevant.

#### The FoE regularizer

Adversarial attacks take a point from the data distribution and aim to find the closest point which facilitates misclassification. The Fields of Experts regularizer approximately reverses this attack by mapping the adversary onto the closest part of the data distribution. Given the small adversarial perturbation, this region likely also contains the source image. In turn, assuming local consistency on the data manifold, points from this region will also map to the same label. This is achieved by iteratively increasing the probability of the adversary under the learned prior through gradient ascent steps. The strength of the regularizer depends on the chosen learning rate and number of steps. If the regularizer is used until convergence, the adversary is mapped to a mode of the learned data distribution, but this can be detrimental, depending on what the learned distribution looks like. The strength of the regularizer thus depends on the learning rate and number of steps taken, for which the optimal choices can be determined empirically, given a set of adversaries to restore, the fooled classifier and an importance weighting of type 1 and type 2 errors.

# Chapter 4

# Inverting Neural Networks

## 4.1 Research questions

This section explores, whether Fields of Experts can be employed as more expressive priors for network inversion. Testing this empirically first requires the reproduction of reference models, then training and optimization of the model itself, and finally the selection of error metrics. As network inversion is not exclusive to reconstructions at image-level, a secondary objective is the finding of efficient feature map priors. These objectives can be summarized in the following two research questions:

**Q**$_{1.1}$: Does the FoE image prior improve reconstruction compared to the reference methods by Mahendran & Vedaldi and Dosovitskiy & Brox?

**Q**$_{1.2}$: Can feature map priors be used to efficiently enable network inversion beyond image level?

## 4.2 Image reconstruction

### 4.2.1 Experiment setup

All models were trained on the validation set of the 2012 ImageNet recognition challenge. The first 2 occurrences of each of the 1000 classes were set aside as a test set, with the remaining 48000 images used for training. The images were cropped to the largest central square and resized to the input height and width of AlexNet (227 x 227) and VGG-16 (224 x 224). VGG-16 is only used to compare FoE priors with Mahendran & Vedaldi's method, so unless explicitly stated, all experiments use AlexNet.

### 4.2.2 Modular inverting networks

The structure of AlexNet is first divided into inverting modules, which are shown in figure 4.1. Modules are selected to begin and end at linear layers so they match the other methods discussed below. Another structure based on ReLU layers has lead to similar results, indicating that this choice is not critical. The modules are first trained individually for 15000 iterations using the Adam optimizer [22] with a learning rate of 3e−4. Following this, the pre-trained modules stacked and trained to optimize only the final reconstruction loss for another 5000 iterations with a learning rate of 3e−5. As shown in figure 4.2, this improves the reconstruction quality, as it prevents some errors from accumulating across modules.

| | Module 1 | Module 2 | Module 3 | Module 4 | Module 5 | Module 6 | Module 7 | Module 8 | Module 9 |
|---|---|---|---|---|---|---|---|---|---|
| Input | conv1/lin | lrn1 | pool1 | conv2/lin | lrn2 | pool2 | conv3/lin | conv4/lin | conv5/lin |
| Target | image | conv1/lin | lrn1 | pool1 | conv2/lin | lrn2 | pool2 | conv3/lin | conv4/lin |
| Filter size | $11 \times 11 \times 96$ | $8 \times 8 \times 96$ | $3 \times 3 \times 96$ | $8 \times 8 \times 256$ | $8 \times 8 \times 256$ | $3 \times 3 \times 256$ | $8 \times 8 \times 384$ | $8 \times 8 \times 384$ | $8 \times 8 \times 384$ |
| Strides | 4 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 |

**Figure 4.1:** Structure of modules for inverting AlexNet

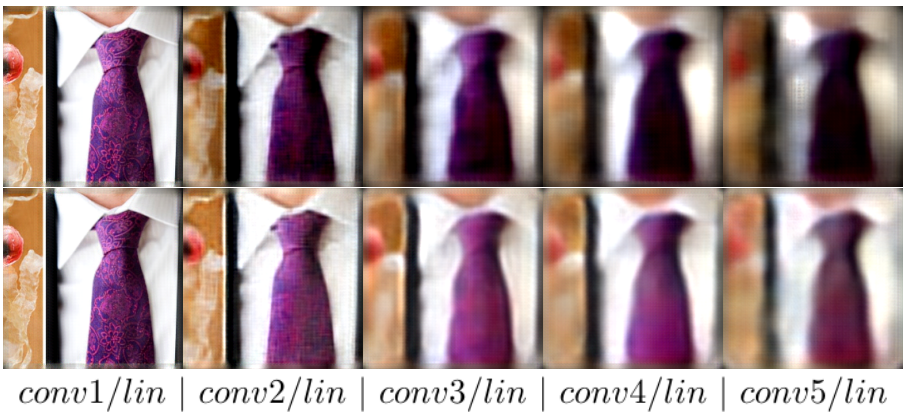



**Figure 4.2:** Reconstruction from feature maps with stacked modules (top) and after additional joint training (bottom)

### 4.2.3 Reproduction of Mahendran and Vedaldi's method

The reconstruction method by Mahendran and Vedaldi [28] has been reproduced using the same regularizer weights and jitter settings as described in their paper. Instead of their variant of AdaGrad, I used Adam for 3500 iterations, stopping jittering after 2750. As recommended, the learning rate is calculated as $\eta B^2/\alpha$ for $B = 80$ and $\alpha = 6$, whith $\eta$ starting at $10^{-2}$ and lowering to $3 \times 10^{-3}$ after 1000 and $10^{-3}$ after 2000 iterations. Figure 4.3 shows a reproduction of the reconstructions they display in their paper. Slight variations can be explained by different initializations and the use of an alternate version of AlexNet[1], as well as the change in optimizers.

**Figure 4.3:** Reconstruction of all AlexNet feature maps for the reference image used in [28] (Fig.9) to ensure an adequate reproduction. (Unlike the original it also includes the layers pool5, fc8/relu and softmax.) Row 1: *conv1/lin* to *pool2*, row 2: *conv3/lin* to *fc6/lin*, row 3: *fc6/relu* to *softmax*

### 4.2.4 Fields of Experts priors

#### Implementation notes

The filter weights $\boldsymbol{\alpha}$ are constrained to positive values by defining them through a softplus function $\boldsymbol{\alpha} = \exp(1 + \log(\mathbf{a}))$. The filters $\mathbf{w}_i$ are mapped to unit norm by defining them through an unconstrained variable as well $\mathbf{w}_i = \frac{\mathbf{w'}_i}{\|\mathbf{w'}_i\|}$. Whitening matrix $\mathbf{A}$ and filter bank $\mathbf{W}$ are combined to efficiently compute the components of all image patches as one convolution over the input. Centering each patch individually would be more accurate, but is computationally costly during runtime, as each patch

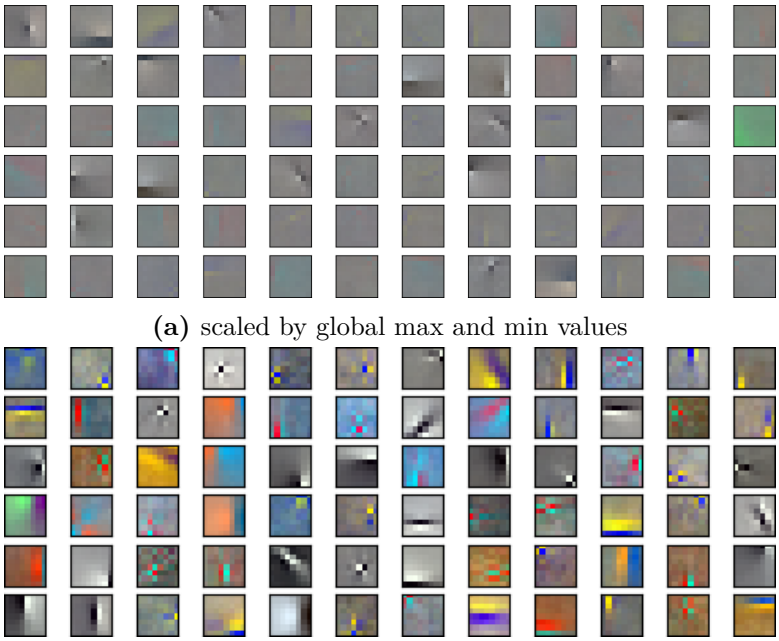
[1] CaffeNet used to have pooling layers placed before local response normalization, as can be seen in their ordering of the reconstructions. This was later changed to the order of AlexNet and found here

would have to be computed separately. using the training set mean and standard deviation, the whole input can be normed at once without division into patches. rescaling per channel is not necessary on image patches (but does no harm either), as the channels have sufficiently similar variance. On feature maps, however, the channels can vary in standard deviation by a factor of 10 or more.

### Training

The patch prior is trained on 100,000 randomly cropped patches from the validation set. For preprocessing, the channel-wise mean and standard deviation are computed and the patches are centered and rescaled to zero mean and unit variance in each channel. The patches are then whitened using PCA-whitening and the whitening matrix is stored along with the training set means and variances for use at runtime. For the image Prior, patches of 8 by 8 pixels have lead to good results. The chosen number of filters is 512, which makes the prior 2.7 times overcomplete a filter size of 192. Both $\mathbf{a}$ and $\mathbf{W}'$ are initialized using Gaussian noise with standard deviations of 1 and $1 \times 10^{-3}$ respectively.

The model is trained for 25000 iterations using Adam and a learning rate of 1 for the first 10000 iterations and a decaying rate afterwards, which starts at 0.1 and alternately decreases by factors of 0.3 and 1/3 every 2000 iterations down to $3 \times 10^{-5}$. The trained filters encode sparse wavelet-like features of the patches, as shown in figure 4.4.



**(a)** scaled by global max and min values

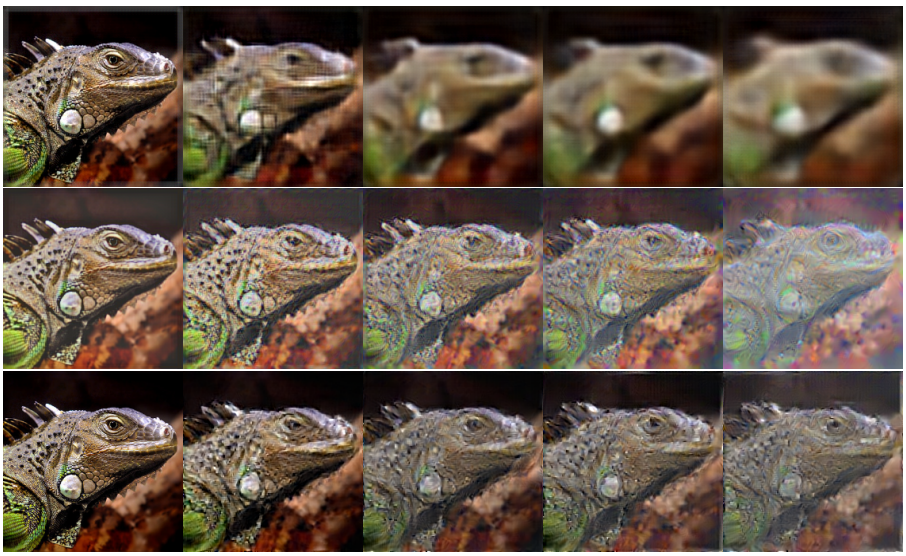
**(b)** scaled per filter

**Figure 4.4:** Subsets of the 512 filters from an over-complete PoT prior

Several other settings were tried, but led to no clear improvements. Patches were centered by the training set mean instead of the true mean at training and test time without negative effects. Using a logistic distribution instead of a Student-t distribution in the patch prior, as found in [19] made no discernible difference in the results.

### Inversion

The optimal weighting of the FoE prior relative to the mean squared error loss is determined through grid search. minimizing the mean squared error of the reconstruction, which has shown to correlate with visual reconstruction quality in this case. The tested values ranged from $1 \times 10^{-7}$ to $1 \times 10^{-2}$ separated roughly by a factor of 3, as performance proved to be not very sensitive to the exact weighting. The chosen weights are determined for all linear layers in this way and interpolated for the others. For the early layers up to *conv4/lin*, the weightings between $1 \times 10^{-4}$ and $3 \times 10^{-3}$ tend to produce very similar scores and $1 \times 10^{-4}$ is consistently the best weighting after that. The



*conv1/lin* | *conv2/lin* | *conv3/lin* | *conv4/lin* | *conv5/lin*

**Figure 4.5:** Comparison of reconstructions of the linear convolutional feature maps from 1 (left) to 5 (right) with all three methods: CNN (top), M&V (middle), FoE (bottom).

first *conv1/lin* layer is a special case, requiring almost no regularization to produce a near perfect reconstruction at a weight of $1 \times 10^{-6}$.

The optimization is performed using Adam at a learning rate of 0.6. Because optimization with this method is considerably slower than Mahendran and Vedaldi's method, the randomly initialized pre-image is first optimized using only the matching error for 5000 iterations. Then the prior loss is added for another 5000 iterations, using jitter as Mahendran and Vedaldi for the first 3200.

### 4.2.5 Results under error metrics

All three methods are qualitatively compared on a small set of ten example images. Figure 4.5 shows a direct comparison between the obtained reconstructions and the full set of reconstructions can be found in the appendix. Reconstruction quality independent of a clear objective is hard to define and quantitative measures are controversial as a result. To give an intuition of the effect the different methods have, I measure two scores: The mean squared error and the VGG-score proposed in [54]. Given that the CNN was explicitly trained to minimize the Mean squared error, it is no surprise that it outperforms the optimization based method on this metric, as seen in figure 4.6. Nonetheless, the FoE prior gives a significant improvement over the M&V reconstructions.

As can be seen in figure 4.7, the FoE prior performs better than the other two methods under the VGG-score metric, which compares the presence of higher order features in images, independent of location. The sample size is admittedly very small, but this effect is consistent. Out of 50 reconstructions in total (10 images, 5 layers) there is only one case where M&V achieves a lower score and no case for the CNN.

| MSE | conv1/lin | conv2/lin | conv3/lin | conv4/lin | conv5/lin |
|---|---|---|---|---|---|
| CNN | 188.45 | **457.82** | **845.15** | **1022.58** | **1419.92** |
| M&V | 868.60 | 1391.62 | 1714.48 | 2197.03 | 3085.13 |
| FoE | **2.786** | 543.16 | 923.23 | 1334.41 | 2317.36 |

**Figure 4.6:** Comparison of mean squared reconstruction errors averaged over the 10 example images.

| VGG-score | conv1/lin | conv2/lin | conv3/lin | conv4/lin | conv5/lin |
|---|---|---|---|---|---|
| CNN | 0.061 | 0.517 | 0.664 | 0.694 | 0.730 |
| M&V | 0.180 | 0.445 | 0.530 | 0.647 | 0.729 |
| FoE | **9.5e-06** | **0.216** | **0.371** | **0.439** | **0.599** |

**Figure 4.7:** Comparison of VGG-scores [54] of the reconstructions averaged over the 10 example images.

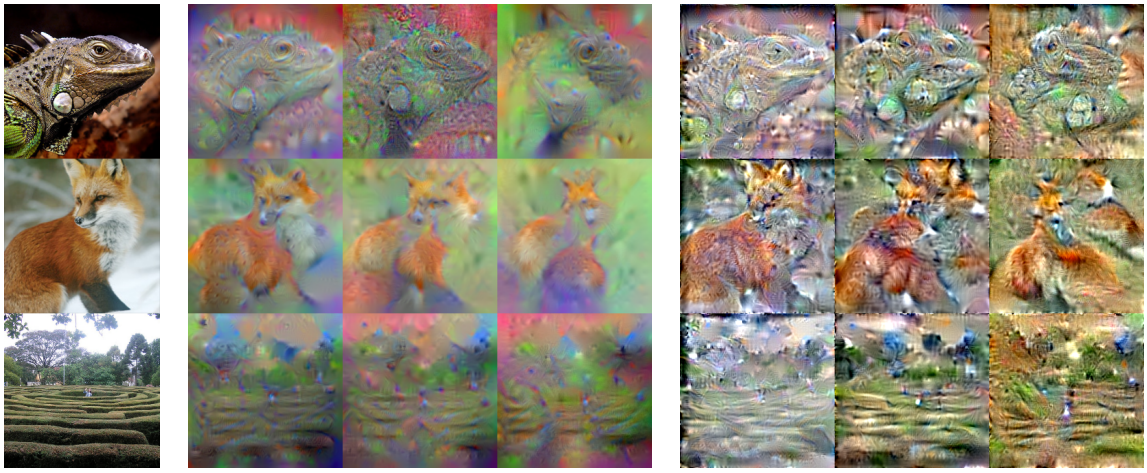


**Figure 4.8:** Reconstructions from fully connected layers 6, 7 and 8. Left: reference images, middle: Mahendran and Vedaldi, right: FoE priors (rescaled for higher contrast). Unnatural colors found by M&V are in part fixed with a better prior.

### 4.2.6 Comparison between AlexNet and VGG-16

In order to show results on a classifier with slightly different properties, optimization-based reconstructions were created for VGG-16, which can be found in the appendix. Because of the differences in architecture, there is no direct correspondence between layers, but both models can be mapped out with roughly evenly spaced layers, as shown in figure 4.9.

Up to the fifth convolutional block, VGG-16 produces almost indistinguishable reconstructions, which can be attributed to its large feature map sizes and number of channels up to that point, which require little compression. A less clear-cut difference between the classifiers can be observed in the kinds of reconstruction errors that appear. Using VGG-16, geometric shapes and patterns appear at a higher resolution than with AlexNet and boundaries between objects are clearer, while color information appears to remain more accurate in AlexNet. Both effects can be found in the Mahendran & Vedaldi reconstructions as well, suggesting they are not a result of the prior.

An odd phenomenon which may however well be a property of the prior, are the color inaccuracies in the first dense layer of both architectures, which are mostly white in AlexNet and violet in the case of VGG-16. Taking samples from additional models such as the remaining VGG networks should make it possible to attribute these phenomena to properties of the architecture, but this is left for a future project.

## 4.3 Feature map reconstruction

In order to test some basic properties of feature map priors, a number of priors are trained on the *conv1/lin* layer with 96 channels. A regular 8 by 8 patch of the feature map has 6144 parameters, so one prior is trained on the features after dimensionality reduction, which keeps only the 3000 dimensions of greatest variance during PCA-whitening. This retains 98% of explained variance and the prior uses 6000 filters. As can be seen in figure 4.10, the learned filters correspond to the convolutional filters of the network. In addition, a prior with 2000 filters is trained on 3x3 image patches and an ensemble of 96 channelwise priors is trained on 8x8 patches with 150 filters each.

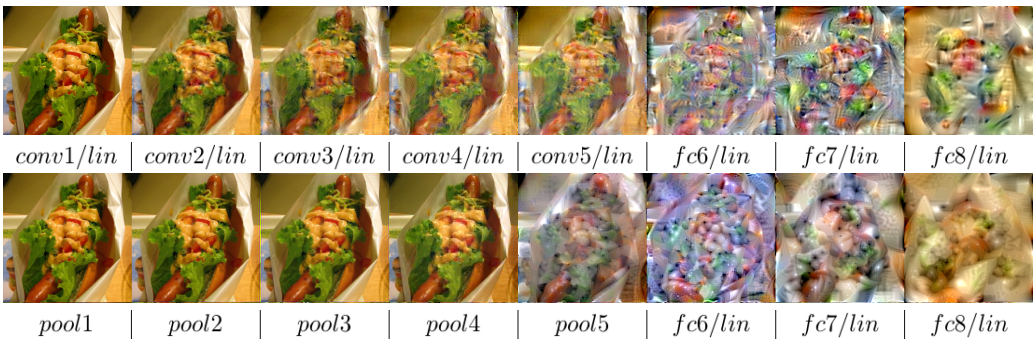
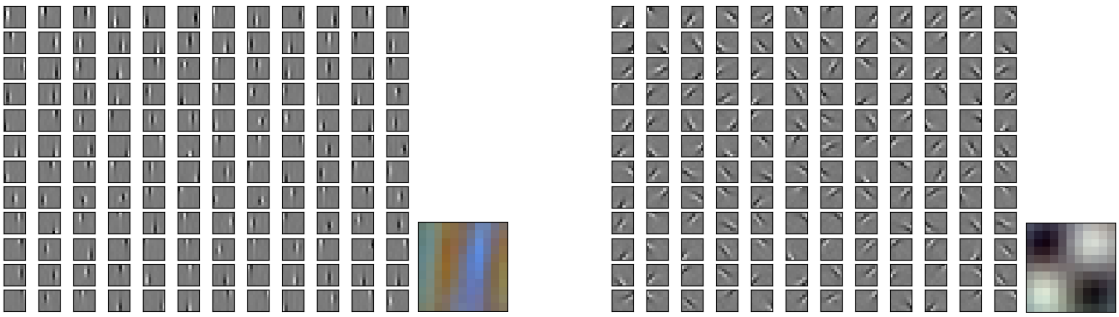



**Figure 4.9:** Comparison of reconstructions of feature maps from AlexNet (top) and VGG-16 (bottom) with FoE priors. (rescaled for higher constrast)



**(a)** Channel 1/96 **(b)** Channel 2/96

**Figure 4.10:** Visualization of partial filters from the 8x8 FoE-prior in layer *conv1/lin*. The filter regions corresponding to a particular channel in the layer are chosen and ordered by their norm. The 144 partial filters with highest norm are shown, along with the corresponding convolutional filter in AlexNet. The filters are sparse, which means that most of the 6000 filters have a norm close to zero in the parts associated with any given channel.

### 4.3.1 Inverting to feature maps

Due to the computational cost of training feature map priors, inversion to feature maps is explored in one simple setup. A selection of priors is used to invert a set of feature maps from the *conv2/lin* layer to *conv1/lin*, with weightings optimized as in the image reconstruction case. Figure 4.12 shows relative reconstruction and matching errors achieved during optimization. The same order of reconstruction quality can be observed by visual comparison of some of the inverted feature maps, as shown in figure 4.11, giving some justification for tuning the prior weighting by MSE. The visualizations also highlight, how simple methods like the total variation prior fail in this setting. Compared to image reconstruction, this inversion poses a harder problem, as the increased target dimensionality implies fewer constraints on the optimization. A side effect of this can be observed in the explicitly optimized MSE of the *conv2/lin* layer, shown in figure 4.12b. The unconstrained optimization progresses slowly and is significantly sped up through the addition of a prior, despite it being a competing objective. In this case, a prior guides the optimization to a region with good optima.

### 4.3.2 Network implicit priors

When inverting towards a *pre-image* at the input, each feature map of that network is constrained by the function that generates it from the pre-image, the latter of which is typically of much lower dimensionality than the feature map itself. Coupling this constraint with an image prior tends to lead to better feature map reconstruction than can be achieved with a feature map prior. I suspect that the strength of these network-implicit priors is responsible for the lack of improvement observed when trying to combine simple (given the computational constraints) feature map priors with an image

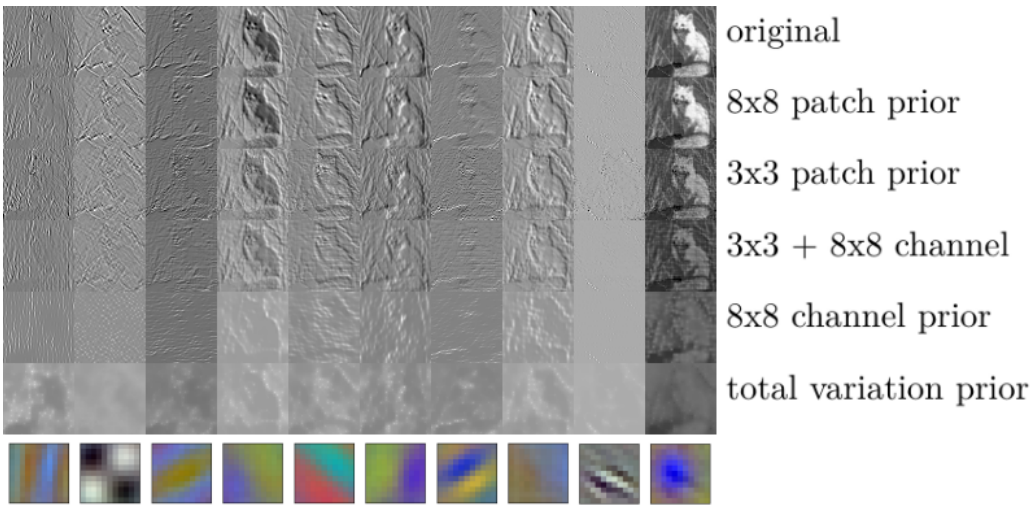
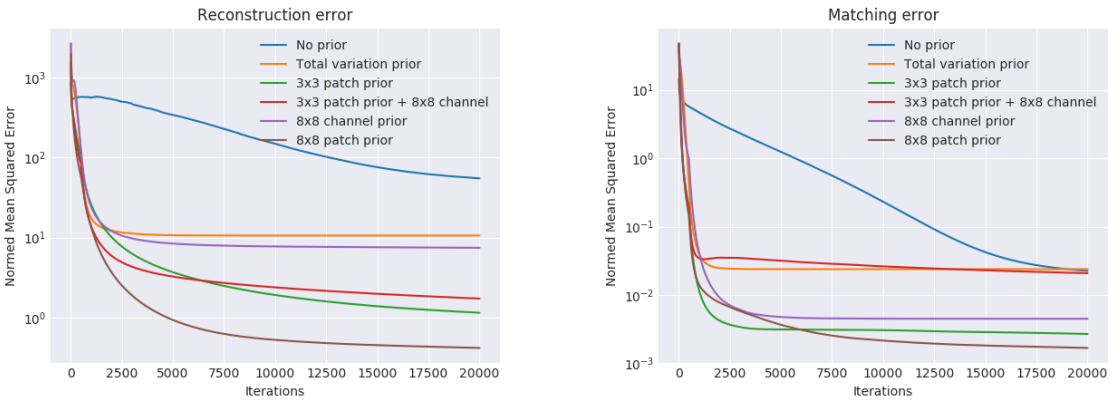



**Figure 4.11:** First 10 *conv1/lin* feature maps, when optimizing MSE in *conv2/lin* under the different priors ordered by reconstruction loss, as shown in fig. 4.12a. Normalization is done by column, so priors are directly comparable. The bottom row shows the corresponding CNN filters.




**(a)** Reconstruction error **(b)** Matching error

**Figure 4.12**

prior for visualization. A prior that only learns the feature map regularities which are encoded in the network does not add any value at test time. That the structure of CNNs already encodes a useful image prior is supported by work of Ulyanov et al. [53], who use untrained CNNs as image priors for tasks such as inpainting and superresolution.

## 4.4 Discussion

In their article on feature visualization, Olah et al. [37] point out that while learned priors provide stronger regularization than heuristics such as frequency penalization and transformation robustness, they also come with a significant drawback. When choosing an expressive prior for reconstruction, it can become unclear, which parts of the reconstruction are attributable to the prior, and which parts to the classifier. The most photo-realistic network visualizations and inversions to this point have been achieved through the use of generative adversarial networks [7, 35], which are powerful generative models, but whose theoretical properties are currently not well understood and whose learned representations are as inaccessible as those of any other deep neural network. If the goal of network inversion is to learn something about the network, trained priors must be used with care, as it is generally difficult to use one opaque model to gain insight into a second one.

Fields of Experts offer a useful middle-ground to this trade-off, because their learned features are

shallow and conditionally independent, which makes them easy to interpret. At the same time they are more powerful than handcrafted heuristics, as the experiments have shown, answering the first research question put forward in this chapter ($\mathbf{Q}_{1.1}$).

With regard to the second research question ($\mathbf{Q}_{1.2}$), the results are less clear. The experiment in section 4.3.1 shows that it is in principle possible to reconstruct feature map via inversion, but also that the problem requires considerably more effort and no efficient models have been found yet.

# Chapter 5

# Denoising Adversarial Examples

## 5.1 Research questions

This thesis proposes that Fields of Experts priors can be employed as a principled defense against adversarial examples. To support this claim empirically, the defense is tested with both oblivious and adaptive attacks and compared to a simple smoothing defense. Following this, two promising avenues to improve the defense are tested, by adding dropout and using feature map information. This can be summarized in the following four research questions:

$\mathbf{Q}_{2.1}$: Does the defense restore oblivious adversaries?

$\mathbf{Q}_{2.2}$: Is the regularizer robust to adaptive attacks?

$\mathbf{Q}_{2.3}$: Can dropout make the regularizer more robust to adaptive attacks?

$\mathbf{Q}_{2.4}$: Do attacks transfer between feature maps?

## 5.2 Experiments

### 5.2.1 Experiment setup

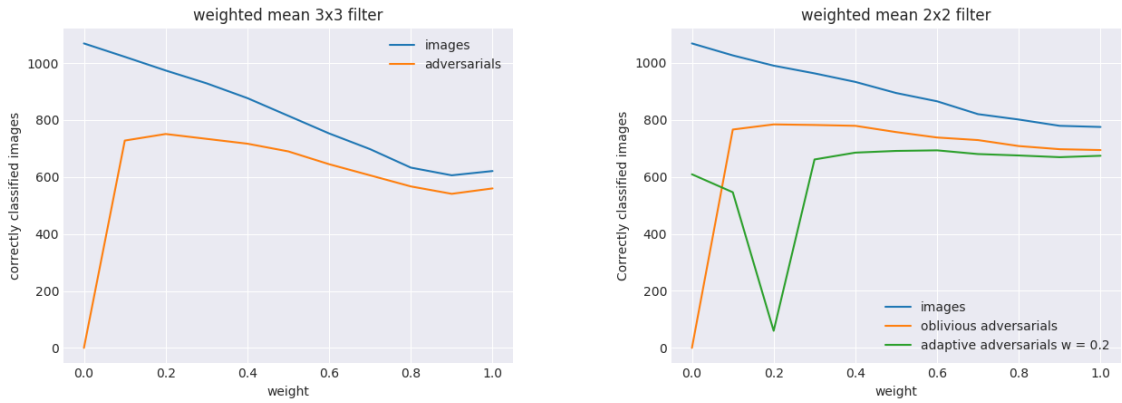
For sake of comparability, the Foolbox package by Rauber et al. [42] (version 0.9.6) is used for all adversarial attacks listed in this section. Unless stated otherwise, experiments were conducted using the Deepfool attack by Moosavi-Dezfooli et al. [30] with the default settings. This choice was made following Metzen et al. [29], who found that Deepfool generated stronger adversaries than fast and iterative gradient sign methods, and based on the current scores on the Robust Vision Benchmark which indicate Deepfool consistently as the strongest tested attack. The recently published attacks by Carlini and Wagner [3] are not yet available in the Foolbox and were left out for this reason.

The defenses were evaluated on a subset of the ImageNet Validation set. Of the 2000 images that were set aside as a test set, when training the prior, 1073 images are classified correctly by the AlexNet model. These images, which are in the training set of neither the classifier nor any priors, are the basis for all following experiments.

### 5.2.2 Mean filter baseline

As a baseline model, I follow the results by Li et al. [25] who have shown that smoothing adversarial examples with a simple mean filter can restore original class labels. After convolving the image with a 3 by 3 mean filter, they report a Top-5 accuracy of 73% on AlexNet for adversaries generated with an L-BFGS attack. This translates to 84.4%, when considering only correctly classified images. When evaluating the same method on the chosen set of correctly classified images under Top-1 accuracy, I obtain a considerably lower accuracy of 52.4%, as shown in figure 5.1a at weight = 1. Using a smaller 2 by 2 filter improves accuracy to 64.9%. Further, the linear interpolation $f(\mathbf{x}, w) = w\mathbf{x} + (1-w)g(\mathbf{x})$ between original image $\mathbf{x}$ and smoothed image $g(\mathbf{x})$ shows, that mean smoothing is stronger than

necessary. Reducing the strength of the smoothing to 20% yields a Top-1 accuracy of 73.3%. As shown in figure 5.1, these steps also improve the retained accuracy on non-adversarial images to 92.7%, which lies at a low 58.1% for the 3 by 3 filter.




**(a)** 3 by 3 filter **(b)** 2 by 2 filter

**Figure 5.1:** Denoising performance of a weighted mean filter

### 5.2.3 Denoising oblivious attacks

Adversarial images are smoothed with an FoE prior by updating the image to maximize their probability under the prior using the Adam optimizer. The regularization strength depends on the learning rate and the number of iterations used. The same image prior which was used for inversion in section 4.2.4 is applied here. A learning rate of 0.6 is chosen in order to keep the number of iterations low, after tests revealed no significant change in performance for smaller rates. Figure 5.2 shows performance over a small number of iterations. At 4 iterations, this model restores 84.9% of adversaries to the correct class label while retaining 96.7% accuracy on non-adversarial images. This constitutes an improvement of 11.6% in adversary restoration and 4.0% in natural image retention over the best mean filter results and is used as the default setting for the defense.

Most state-of-the-art attacks such as the above used Deepfool method minimize the amount of adversarial noise used, which makes it easier to smooth out oblivious attacks. When the regularization is applied to adversaries created by the fast gradient sign method (FGSM) by Goodfellow et al. [13], which causes larger perturbations, restoration performance is significantly worse. As seen in figure 5.3, the FoE regularizer still outperforms the mean filter, but stronger regularization is necessary, which also affects the rate of correctly classified regular images. Given that attacks with large adversarial noise are generally easier to detect by other means and recent developments have focused on minimizing the amount of perturbation, this finding is however only of limited importance.

### 5.2.4 Adaptive attacks

An adaptive attack takes aims to fool both the classifier as well as any defenses that have been put in place against oblivious adversaries. In overcoming the Fields of Experts regularizer defense, there are two straightforward approaches that can be taken. A secondary optimization objective can be integrated in the attack, which maximizes the likelihood of the adversary under the FoE prior. When the adversary is at an optimum for the prior, it will not change under regularization. While this type of attack is generally possible with optimization-based attacks, such as those by Carlini and Wagner [3], the Foolbox package is not flexible enough to allow this type of modification. Instead, I choose the second option, which views the regularizer as a part of the classifier and propagates gradients through both classifier and regularizer to create the adversary. In practice, this is done in TensorFlow by rolling out the iterative process and computing the gradient ascent steps as a sequence of feed-forward

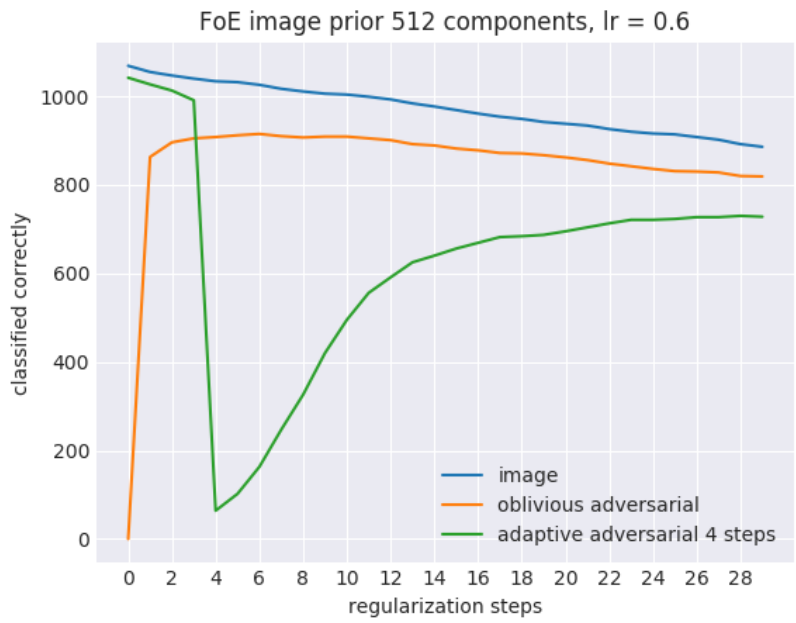
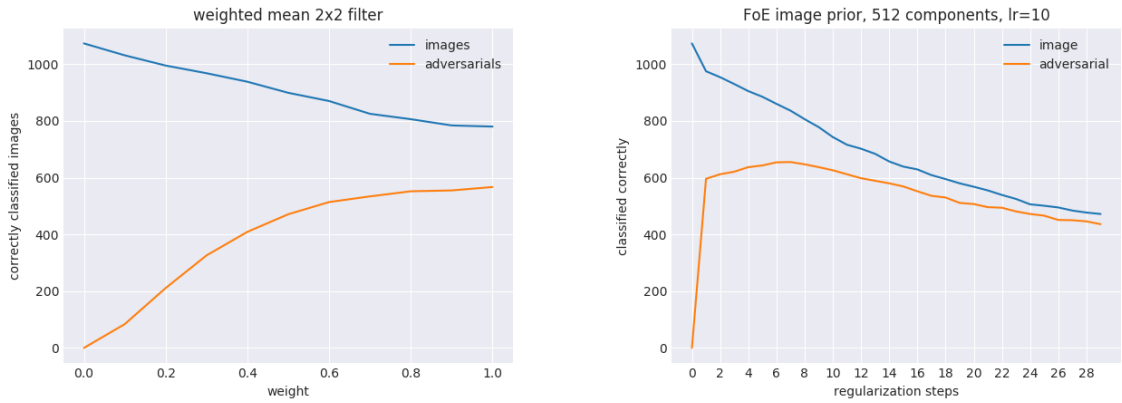



**Figure 5.2:** FoE image regularizer effects over several iterations on natural images and adversaries



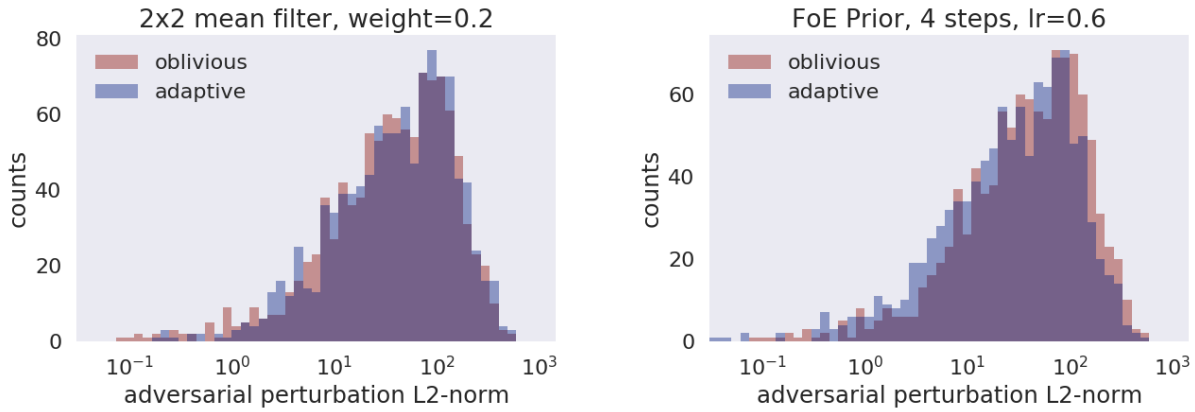

**(a)** weighted 2x2 mean filter **(b)** FoE image regularizer

**Figure 5.3:** Denoising performance on FGSM adversaries

operations[1].

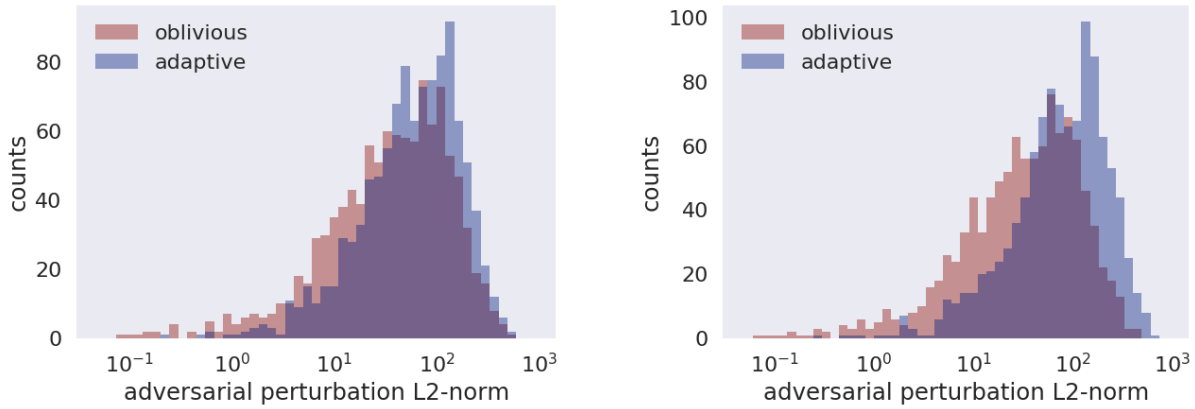
The robustness of a defense against adaptive attacks is typically measured by the increase in perturbation under $L_2$-norm that is necessary to overcome it. The distribution of norms for both methods is shown in figure 5.4. While the mean filter causes a small increase from 70.5 to 75.0, the FoE regularizer surprisingly leads to a mean decrease in perturbation down to 56.5. A different picture arises, when the norms between images and adversaries after FoE regularization are considered in figure 5.5a. The mean squared difference between the original image and oblivious adversaries decreases to 67.3. However, the norm of adaptive ones increases to 89.3, entering the classifier with larger perturbation. Figures 5.2 and 5.1b show that adaptive attacks on particular settings of the regularizer are quite sensitive to its weighting or number of iterations respectively. FoE priors exhibit an asymmetry here, where adversaries transfer to some extent to additional iterations, but largely fail when fewer iterations take place.

[1]Implementation note: Tensorflow's Adam optimizer uses the optimized version suggested by Kingma and Ba [22], so the parameter $\varepsilon$ is actually $\hat{\varepsilon}$ but still receives a default value of 1e-8. This difference is irrelevant for normal use cases, but leads to very different outcomes after only a few iterations.




**(a)** 2 by 2 mean filter **(b)** FoE prior

**Figure 5.4:** Norms of adversarial noise used in oblivious and adversarial attacks. While a trend in (a) is hard to make out, (b) shows that adaptive attacks on the FoE regularizer require smaller perturbations than oblivious ones.




**(a)** 4 iterations (adaptive target) **(b)** 15 iterations

**Figure 5.5:** Comparison of the $L_2$-norms between adversaries and images after regularization. While regularization decreases the mean norm for oblivious adversaries from 71.2 to 67.3 (a) to 59.2 (b), it increases the norm for adversaries from 56.5 to 89.3 (a) and 113.8 (b).

### 5.2.5 Dropout regularizer

In [4] Carlini and Wagner report that dropout based defenses are among the few types of defenses, which achieved a significant increase in perturbation, when attacked with their method. Using dropout at test time in a classifier typically causes only small losses in accuracy on natural images and has the benefit of not offering the attacker a clear gradient, as the exact computation of the network changes on every run. To see, whether dropout can be used in the FoE regularizer for the same benefit, I train a more than twice overcomplete image prior with 1024 filters. Only at test time, 50% of the filters are dropped, with different ones on each iteration. Despite the dropout, the regularizer shows similar performance on oblivious adversaries. When an adaptive attack is directed at the prior with deactivated dropout, the generated adversaries do not translate to the prior with dropout turned on, as figure 5.6 shows.

### 5.2.6 Feature map transferability

As a second modification I tested, whether adaptive adversaries transfer to regularizers at different feature maps. Instead of targeting the image directly, a prior can be used to constrain any feature

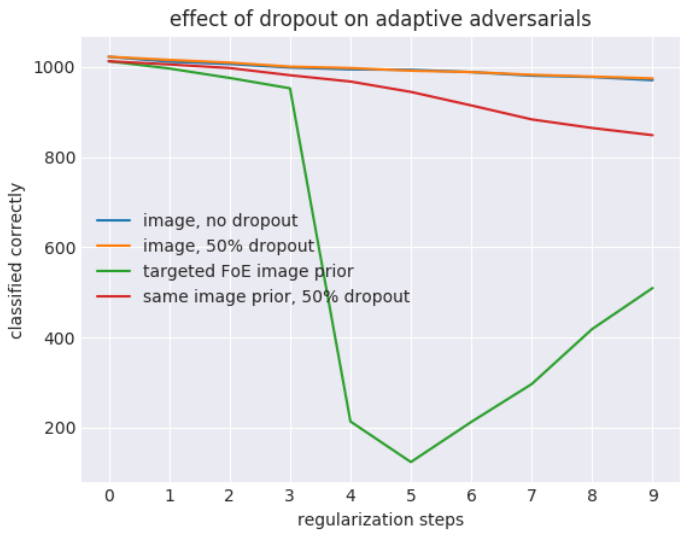



**Figure 5.6:** Transferability test of adaptive adversaries to regularizer with 50% dropout. The changed regularizer is not affected by the adversaries.

map representation of the input image. The regularization updates are backpropagated into image space and given that the higher order representation is regularized to lie on or close to the data(-representation) manifold, classification remains accurate, regardless of what the image looks like after regularization. The closest feature map to the input in AlexNet is the *conv1/lin* layer and in figure 5.7 it can be seen, that adaptive adversaries targeting an image-based regularizer have no effect on a regularizer based on the *conv1/lin* feature map. It is therefore possible to combine regularizers from multiple layers into one defense, either as an ensemble, or in a probabilistic strategy, in order to make adversarial attacks more difficult.

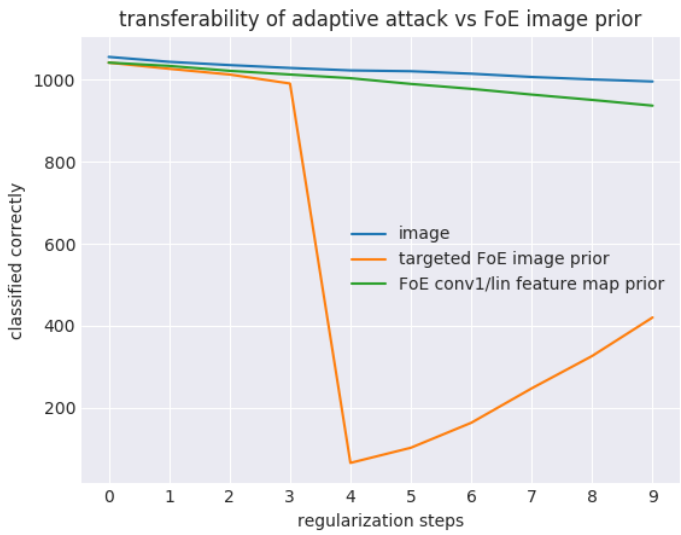



**Figure 5.7:** Transferability test of adaptive adversaries to regularizer on the *conv1/lin* layer. The changed regularizer is not affected by the adversaries.

## 5.3 Discussion

The experimental results provided in this section, are not sufficient to claim that the proposed FoE-based defense is competitive with the rapidly changing state of the art in this field. This would

require testing with a broader range of attacks and state of the art image classifiers, and, in all likelihood, further refinement of the method as well. The goal of this chapter has been, to establish that the regularizer possesses a number of desirable qualities, such as its attack-agnostic training and the relative independence from the type of classifier it is attached to. As shown, the latter property allows one to use augmentations such as dropout without changes to the classifier and the theoretical considerations do work in practice. In terms of the research questions formulated at the beginning of the chapter, this means that, yes, the defense restores oblivious adversaries ($\mathbf{Q}_{2.1}$), Dropout increases robustness to adaptive attacks ($\mathbf{Q}_{2.3}$) and feature map regularizers are not susceptible to adversaries targeting an image level defense ($\mathbf{Q}_{2.4}$).

The finding that the regularizer is vulnerable to adaptive attacks ($\mathbf{Q}_{2.2}$) is worth discussing independently, as it relates to previous work on denoising autoencoders (DAE) by Gu et al. [15]. The DAE architecture is related to the FoE model through a finding by Alain and Bengio [2] that DAEs approximate the score of the data distribution, computing and mapping the input to a local mean. This corresponds to the gradient ascent steps performed by the FoE regularizer, which, as explained in section 3.5.2, was also trained to approximate the score of the data distribution. The vulnerability for FoE regularizers to adaptive adversarial attacks matches a finding of theirs when they tested DAEs as a defense against adversarials and observed a drop in necessary perturbation for adaptive attacks. They suggest that this may be due to the fact that the DAE is trained without the knowledge of the classification objective and may have more "blind-spots" with respect to the final objective, but do not ultimately provide an explanation of the phenomenon.

Looking at the difference in regularized norms shown in figure 5.5 adds another perspective to this. It also explains the asymmetric sensitivity of adaptive attacks to the number of regularization steps observed in figure 5.2. Adaptive attacks use the regularizer to diverge from the regular image. If this process is cut short, it does not cross the decision boundary to the other class. What is unexpected, however, is that upon extending the regularization process, the adversaries slowly revert back to the original class label, although the adversary diverges further from the image. This indicates that the adversarial finds a misclassified *pocket* close to the data manifold rather than leaving the manifold altogether.

# Chapter 6

# Conclusion

This thesis has shown how a relatively simple image prior can be utilized to give a discriminatively trained classifier some of the benefits of a generative model. Through constraining its input space to an estimate of the data distribution, it was possible to generate recognizable reconstructions from compressed feature map representations and restore a high percentage of adversarial examples.

Although the results provided in this thesis can stand on their own, there are a number of concrete experiments, which exceeded the scope of this project, but would help provide a more complete picture. The portion of these tests, which is considered necessary before one can fully assess, how the presented results will generalize are discussed in the section on limitations. In the section after that, I provide a broader outlook on promising extensions and other application areas.

## 6.1 Limitations

### Reference feature map prior

The exploration of feature map priors presented here is far from complete. In order to gain an upper bound estimate of what the FoE prior could model, it would have helped to train an unmodified and overcomplete feature map prior for reference, despite not being feasible in practice. Given the project setup and available computing resources, this was not an option. An extended study of FoE feature map priors and their modifications should start here.

### Compression in modern architectures

As seen in the juxtaposition with VGG-16, AlexNet compresses its input a lot more than newer models. This made it a natural choice for this project, as the comparatively smaller model meant less computational cost and early compression had the advantage of actually giving the prior something to do. This does, however, raise the question, how much of the observed improvement over existing approaches translates to modern architectures.

### Limited attack models

Using the Foolbox package to create adversarial examples has substantially reduced the effort to conduct the experiments presented in the previous chapter. However, the framework does come with a number of constraints. Defenses with probabilistic components such as dropout can be countered by attacking an ensemble of deterministic samples of the defense. This setup is not available in the Foolbox. The same is true for attacks with a secondary objective, such as maximizing probability under the prior. If other probabilistic defenses are any indication, it can be expected, that the current dropout defense is overcome with an ensemble approach. Whether attacks with secondary objective will be successful is less clear.

Another shortcoming of the Foolbox is that it does not include the attacks by Carlini and Wagner, as they have been shown to beat several defenses, which were effective against other attacks. In particular, they have been shown to fool detectors and efficiently conduct adaptive attacks, with

smaller perturbations than other attacks. As the FoE regularizer defense is not a detector and already relies on small perturbations, no significant difference in performance is expected here. The attack will, however, be relevant when exploring the previously mentioned scenarios.

#### Missing benchmarks

A stark limitation inherent in the current research culture around adversarial examples is the lack of comparability. As already mentioned in the literature overview, testing methodologies still vary by many factors, but is currently getting more organized, as the research area becomes more established. When some consensus about testing protocols is reached, it may be worth the effort to extend the investigation of the FoE regularizer accordingly, beyond what has been presented here.

## 6.2 Directions for future work

#### Efficient feature map priors

A central but non-trivial task is the search for feature map priors that are efficiently trainable. Finding such, would allow additional constraints on the input space, which will benefit the methods discussed here. The two proposed variants of factorized filters and mixed clique potentials give an intuition of how this might be done, but requires further work to overcome their respective problems.

#### Transformation robustness

Other benefits to network inversion can likely be gained by integrating the transformation robustness regularization recently used by Olah et al. [37], which includes several steps of jittering, rotation, and scaling. This regularization is orthogonal to the use of an image prior and can be used in place of the comparatively simple jittering which is currently employed.

#### Learning rate randomization

The FoE regularizer possesses a number of properties, which may be utilized to complicate attacks against it. The experiments have shown that adaptive attacks are highly sensitive to the learning rate and number of iterations, while restoration performance is not. Randomizing these values complicates attacks and, like dropout, requires attacking a sampled ensemble.

#### Stronger priors for adversarial restoration

While Fields of Experts are a good choice in network inversion, thanks to their transparent design, the defense against adversarial attacks does not benefit directly from this property and other, more powerful priors are worth exploring. While many currently competitive defenses are motivated by approximating some kind of image prior, few commit explicitly to modeling the data distribution. I believe that taking this step offers a conceptual clarity and independence of specific attack scenarios, which can be valuable in analyzing the phenomenon of adversarial examples in the long run. Even when such models fail, they may provide an indication, what assumptions, for instance the assumption that adversarial examples do not lie on the data distribution, were violated.

# Bibliography


[1] Anonymous (2018). i-RevNet: Deep Invertible Networks. International Conference on Learning Representations. https://openreview.net/forum?id=HJsjkMb0Z

[2] Alain, G., & Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. The Journal of Machine Learning Research, 15(1), 3563-3593.

[3] Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In Security and Privacy (SP), 2017 IEEE Symposium on (pp. 39-57). IEEE.

[4] Carlini, N., & Wagner, D. (2017). Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. arXiv preprint arXiv:1705.07263.

[5] Chen, P. Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C. J. (2017). ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. arXiv preprint arXiv:1708.03999.

[6] Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint arXiv:1610.02357.

[7] Dosovitskiy, A., & Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. In Advances in Neural Information Processing Systems (pp. 658-666).

[8] Dosovitskiy, A., & Brox, T. (2016). Inverting visual representations with convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4829-4837).

[9] Dziugaite, G. K., Ghahramani, Z., & Roy, D. M. (2016). A study of the effect of JPG compression on adversarial images. arXiv preprint arXiv:1608.00853.

[10] Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. University of Montreal, 1341, 3.

[11] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

[12] Gomez, A. N., Ren, M., Urtasun, R., & Grosse, R. B. (2017). The reversible residual network: Backpropagation without storing activations. arXiv preprint arXiv:1707.04585.

[13] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[15] Gu, S., & Rigazio, L. (2014). Towards deep neural network architectures robust to adversarial examples. arXiv preprint arXiv:1412.5068.

[16] Guo, C., Rana, M., Cisse, M., & van der Maaten, L. (2017). Countering Adversarial Images using Input Transformations. arXiv preprint arXiv:1711.00117.

[17] Hammersley, J. M., & Clifford, P. (1971). Markov fields on finite graphs and lattices.

[18] He, W., Wei, J., Chen, X., Carlini, N., & Song, D. (2017). Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. arXiv preprint arXiv:1706.04701.

[19] Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(Apr), 695-709.

[20] Kessy, A., Lewin, A., & Strimmer, K. (2017). Optimal whitening and decorrelation. The American Statistician, (just-accepted).

[21] Kindermans, P. J., Schütt, K. T., Alber, M., Müller, K. R., & Dähne, S. (2017). PatternNet and PatternLRP–Improving the interpretability of neural networks. arXiv preprint arXiv:1705.05598.

[22] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[23] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[24] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[25] Li, X., & Li, F. (2016). Adversarial examples detection in deep networks with convolutional filter statistics. arXiv preprint arXiv:1612.07767.

[26] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[27] Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5188-5196).

[28] Mahendran, A., & Vedaldi, A. (2016). Visualizing deep convolutional neural networks using natural pre-images. International Journal of Computer Vision, 120(3), 233-255.

[29] Metzen, J. H., Genewein, T., Fischer, V., & Bischoff, B. (2017). On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267.

[30] Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2574-2582).

[31] Mordvintsev, A., Olah, C., & Tyka, M. (2015). Inceptionism: Going deeper into neural networks. Google Research Blog. Retrieved June, 17, 15.

[32] Mordvintsev, A., Olah, C., & Tyka, M. (2015). Deep Dream - a code example for visualizing Neural Networks. Google Research Blog. Retrieved Juli, 1, 15.

[33] Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 427-436).

[34] Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In Advances in Neural Information Processing Systems (pp. 3387-3395).

[35] Nguyen, A., Yosinski, J., Bengio, Y., Dosovitskiy, A., & Clune, J. (2016). Plug & play generative networks: Conditional iterative generation of images in latent space. arXiv preprint arXiv:1612.00005.

[36] Odena, A. Dumoulin, V. & Olah, C. (2016) Deconvolution and Checkerboard Artifacts. Distill. http://doi.org/10.23915/distill.00003

[37] Olah, C. Mordvintsev, A. & Schubert, L. (2017) Feature Visualization. Distill. http://doi.org/10.23915/distill.00007

[38] Papernot, N., McDaniel, P., & Goodfellow, I. (2016). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277.

[39] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016, March). The limitations of deep learning in adversarial settings. In Security and Privacy (EuroS&P), 2016 IEEE European Symposium on (pp. 372-387). IEEE.

[40] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2016). Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint arXiv:1602.02697.

[41] Papernot, N., Goodfellow, I., Sheatsley, R., Feinman, R., & McDaniel, P. (2016). cleverhans v1. 0.0: an adversarial machine learning library. arXiv preprint arXiv:1610.00768.

[42] Rauber, J., Brendel, W., & Bethge, M. (2017). Foolbox v0. 8.0: A Python toolbox to benchmark the robustness of machine learning models. arXiv preprint arXiv:1707.04131.

[43] Roth, S., & Black, M. J. (2005, June). Fields of experts: A framework for learning image priors. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 2, pp. 860-867). IEEE.

[44] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.

[45] Sabour, S., Cao, Y., Faghri, F., & Fleet, D. J. (2015). Adversarial manipulation of deep representations. arXiv preprint arXiv:1511.05122.

[46] Shwartz-Ziv, R., & Tishby, N. (2017). Opening the Black Box of Deep Neural Networks via Information. arXiv preprint arXiv:1703.00810.

[47] Sifre, L., & Mallat, S. (2014). Rigid-motion scattering for image classification (Doctoral dissertation, Ph. D. thesis).

[48] Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

[49] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[50] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

[51] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[52] Teh, Y. W., Welling, M., Osindero, S., & Hinton, G. E. (2003). Energy-based models for sparse overcomplete representations. Journal of Machine Learning Research, 4(Dec), 1235-1260.

[53] Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2017). Deep Image Prior. arXiv preprint arXiv:1711.10925.

[54] Ustyuzhaninov, I., Brendel, W., Gatys, L., & Bethge, M. (2016). What does it take to generate natural textures?.

[55] Welling, M., Osindero, S., & Hinton, G. E. (2003). Learning sparse topographic representations with products of student-t distributions. In Advances in neural information processing systems (pp. 1383-1390).

[56] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.

[57] Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

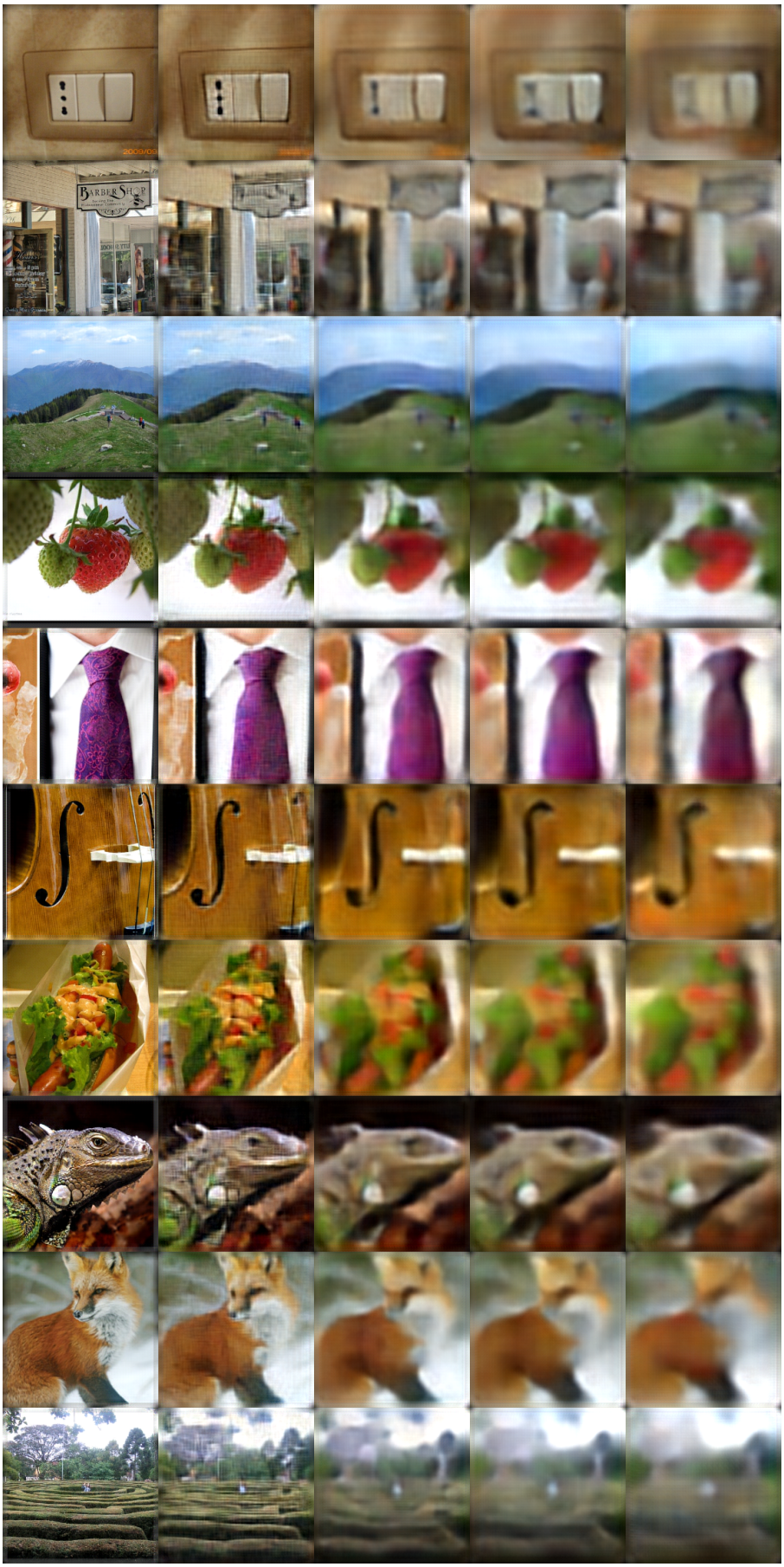
[58] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

[59] Zintgraf, L. M., Cohen, T. S., Adel, T., & Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. arXiv preprint arXiv:1702.04595.

# Appendix A

# Additional visualizations

**Figure A.1:** The original set of images used in Chapter 4.



*conv1/lin* | *conv2/lin* | *conv3/lin* | *conv4/lin* | *conv5/lin*

**Figure A.2:** AlexNet: CNN inversion results for all linear convolutional feature maps

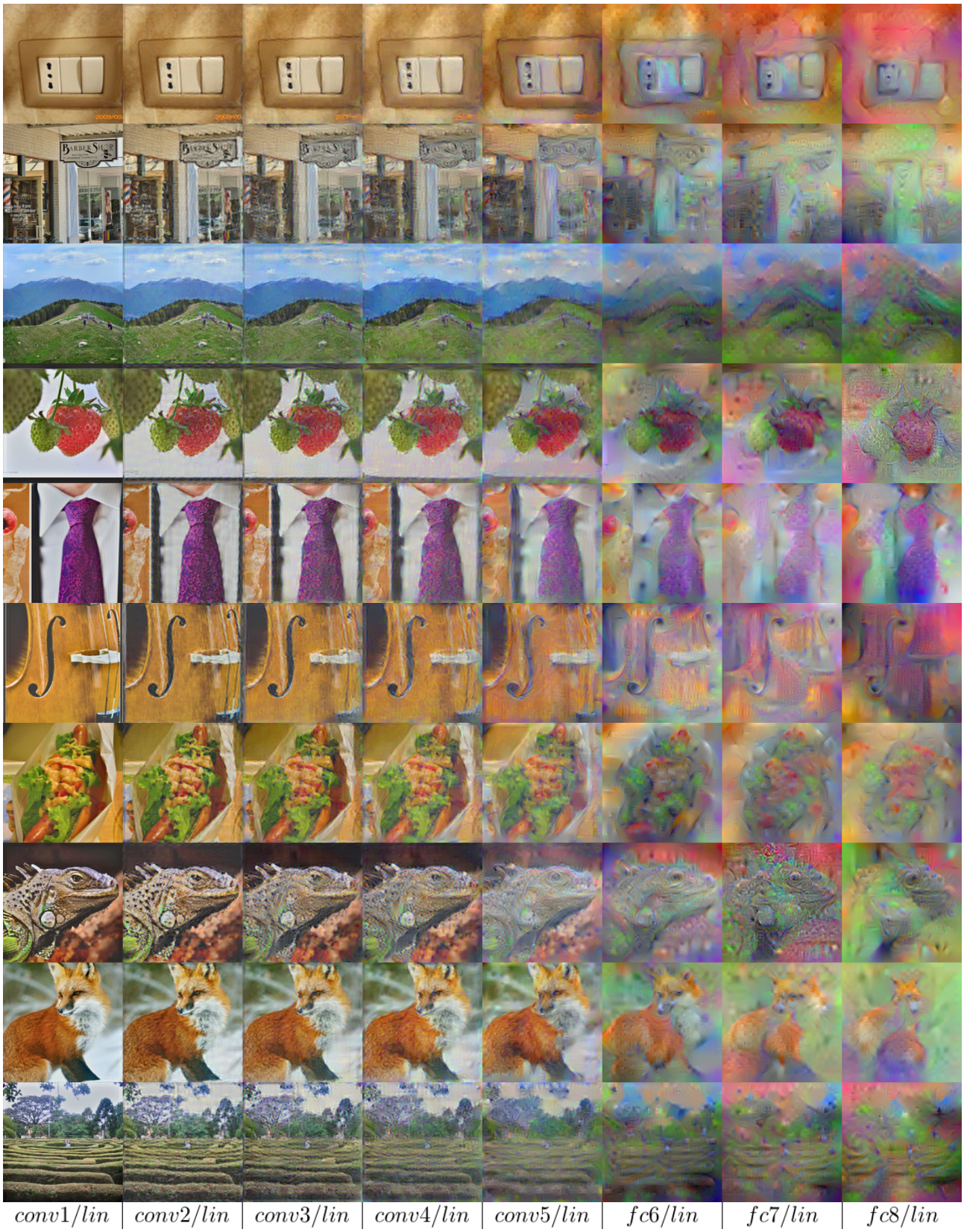



**Figure A.3:** AlexNet: M&V inversion results for all linear feature maps

**Figure A.4:** AlexNet: FoE inversion results for all linear feature maps

**Figure A.5:** AlexNet: FoE inversion results for all linear feature maps rescaled for higher contrast

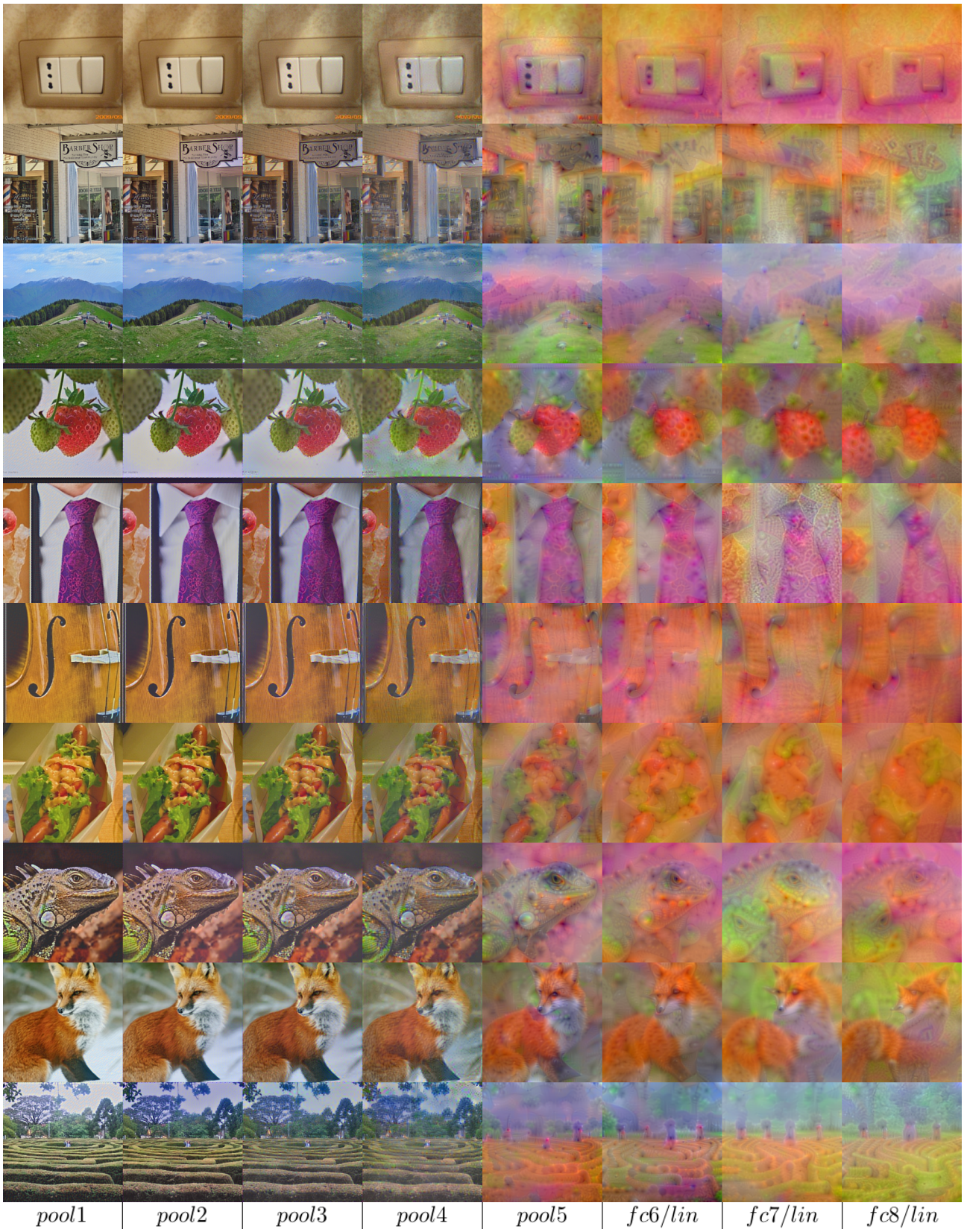



**Figure A.6:** VGG-16: M&V inversion results for all pooling and dense linear feature maps

**Figure A.7:** VGG-16: FoE inversion results for all pooling and dense linear feature maps

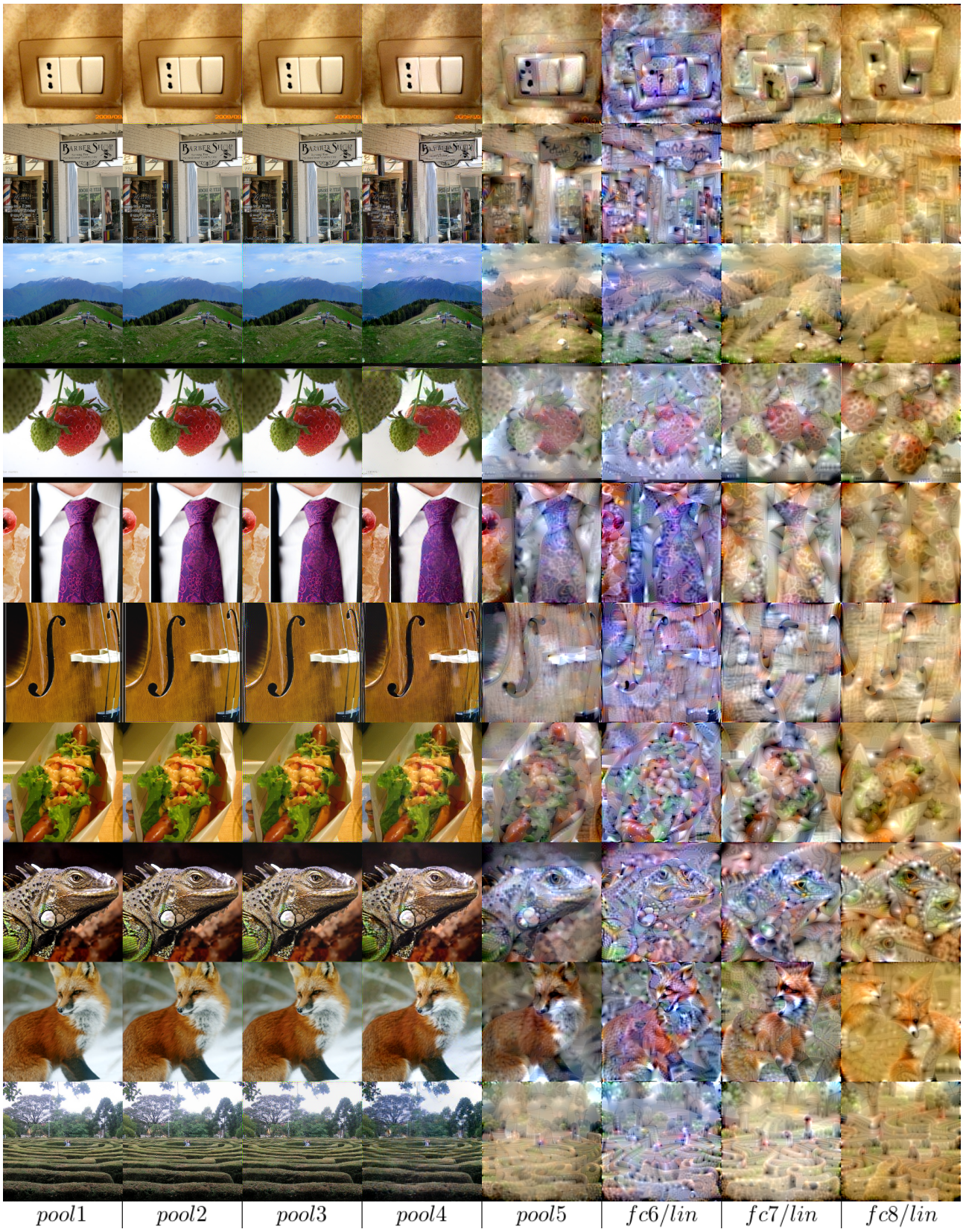



**Figure A.8:** VGG-16: FoE inversion results for all pooling and dense linear feature maps rescaled for higher contrast

# Appendix B

# Separable convolutions

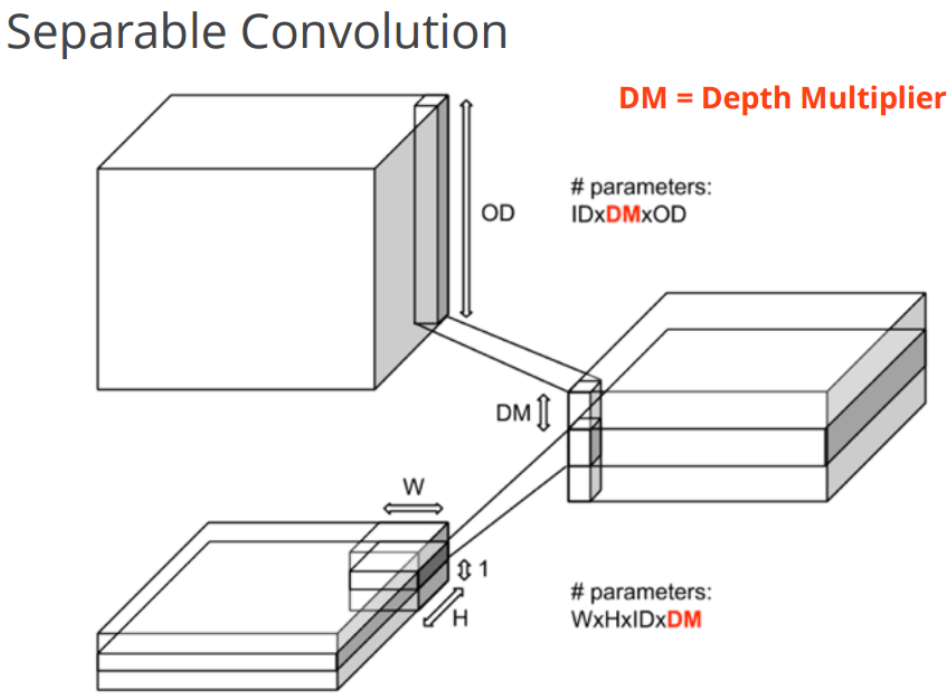



**Figure B.1:** Illustration of depthwise separable convolutions. bottom: input feature map, right: result of depthwise convolution, top: result of columnwise convolution (found online, original source could not be determined)