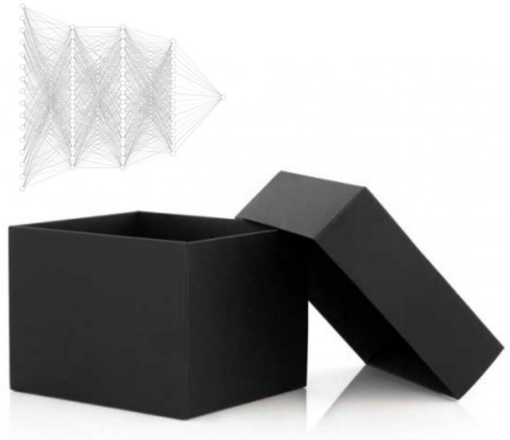
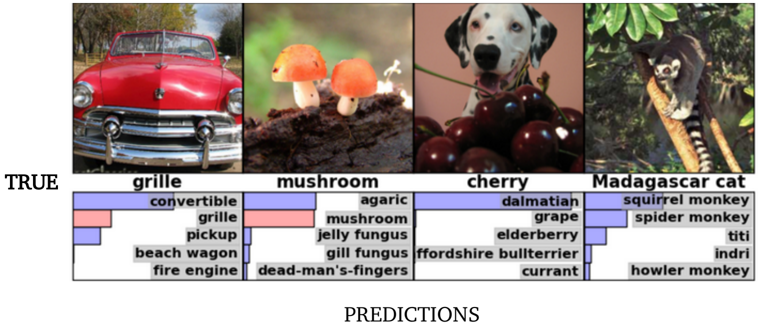
Interpretability and Deep Learning

– Feature Visualization
– Prediction Analysis

Jonas Glombitza

RWTH Aachen

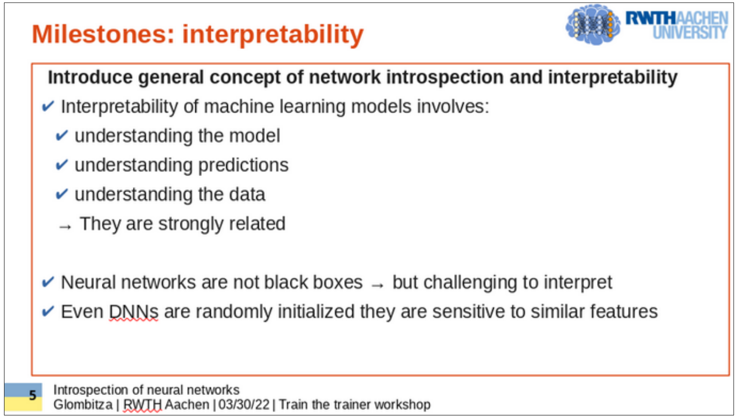




Structure

- Example lecture
  - introduction to interpretability
    (field is large and developing fast)

- Milestone slides:
  - pedagogical reasoning (and important points)



Feel free to ask questions during the seminar!
Just raise your hand...

Interpretability

Deep models have thousands of parameters → open black box

- "What is the model learning?"
  - ➔ learn from trained model
- "Can we trust the model? Does the model work as expected?"
  - ➔ model verification
    → systematic studies (strongly application dependent)

➔ **Interpretability**

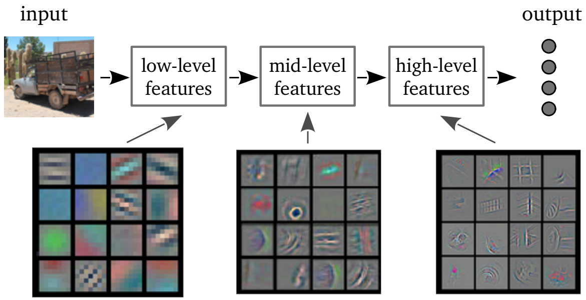
ArXiv/1602.04938

(a) Husky classified as wolf (b) Explanation

“Why is the model predicting a certain class / value?”

Predictions

Interpretability



Model

Data

“How is the model working / are features formed?”
“How do DNNs see the world?

“Which part of the data is most useful?”

Milestones: interpretability

Introduce general concept of network introspection and interpretability

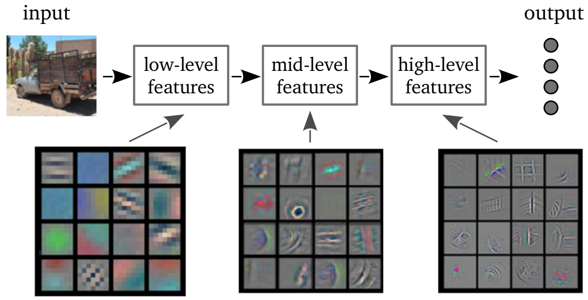
- ✔ Interpretability of machine learning models involves:
  - ✔ understanding the model
  - ✔ understanding predictions
  - ✔ understanding the data

  → They are strongly related

- ✔ Neural networks are not black boxes → but challenging to interpret
  - ✔ propagate signals backwards, they are differentiable (no sampling needed!)
- ✔ Even DNNs are randomly initialized they are sensitive to similar features

Feature Visualization

Model Interpretability

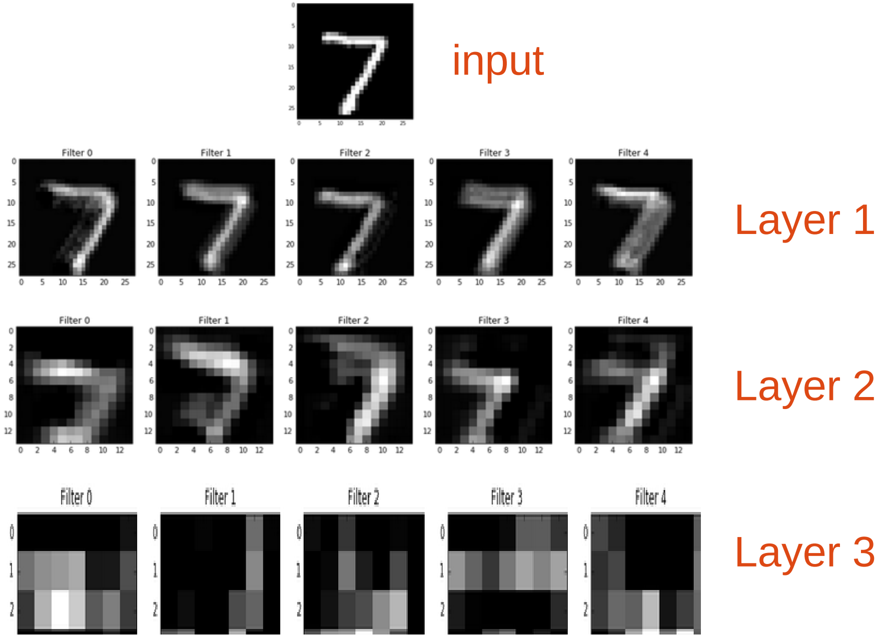


“How is the model working / are features formed?”
“How do DNNs see the world?

Visualization of an MNIST CNN

Visualization of activations

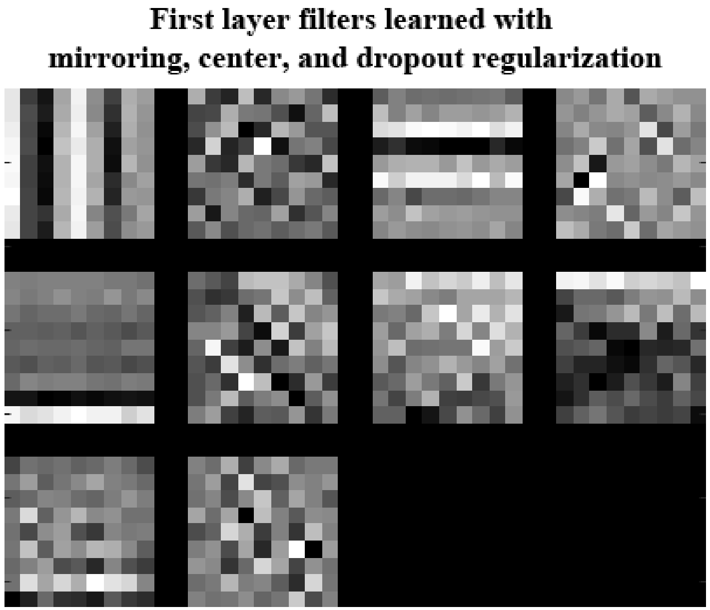
- Propagation of input through model
- Later activations hard to interpret



Arthur Juliani - Visualizing Neural Network Layer Activation (Tensorflow Tutorial), Medium

Visualization of first layer filters

- Edge detection
- Focus on structures in the center



James Hays - http://cs.brown.edu/courses/cs143/2017_Spring/proj6a/

Transposed Convolution ('Deconvolution')

What input patterns caused a given activation?

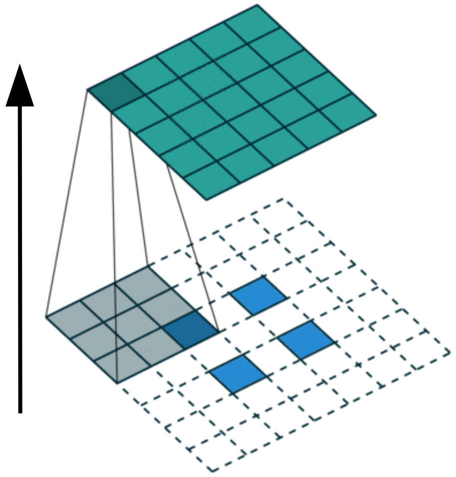
- Visualize intermediate feature layers

Idea: Map activations back to input space

- Use transposed convolution layer to "invert" convolutions (approximately)
- Mapping from feature space → input space

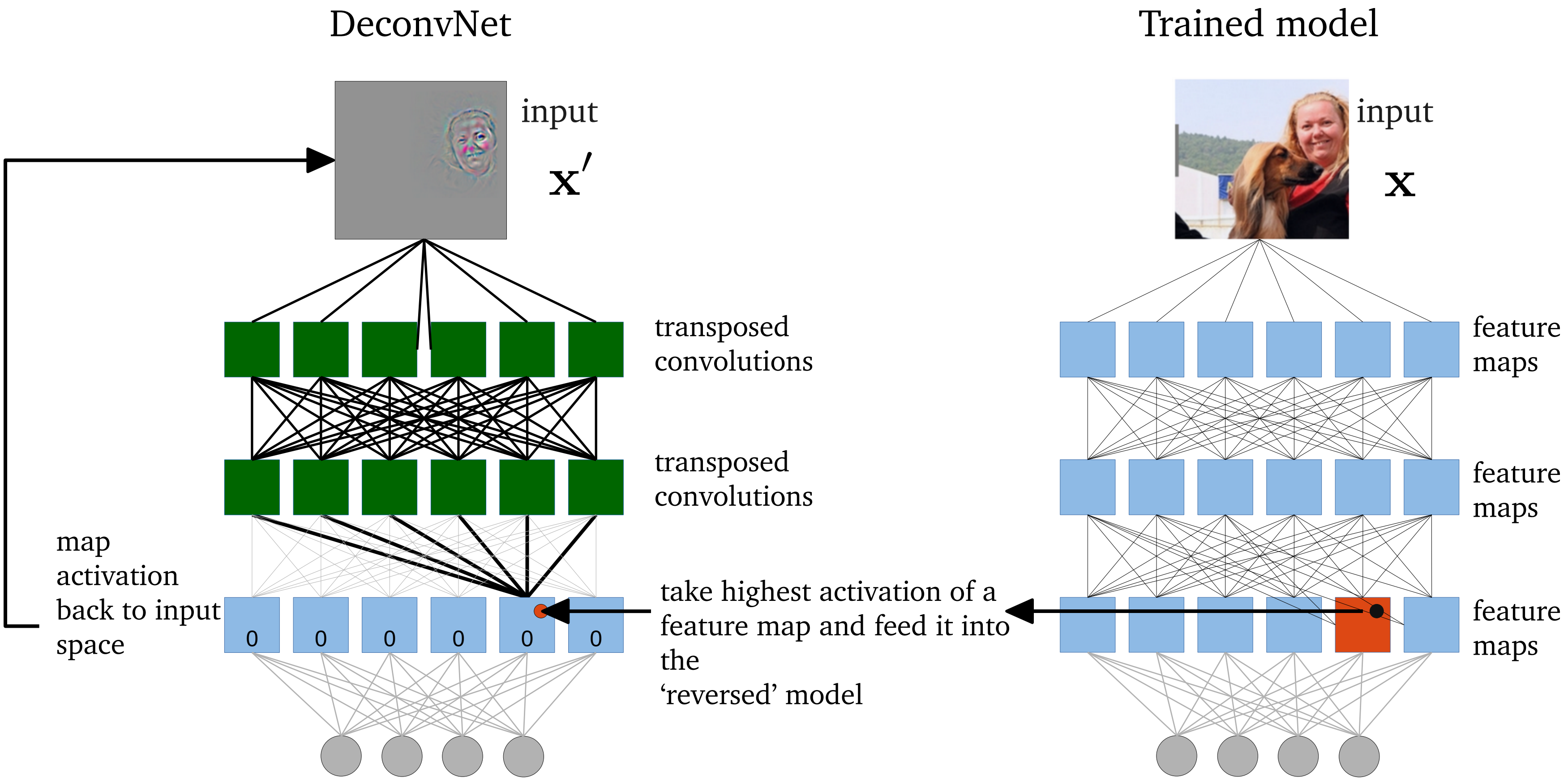
- ✓ Use highest activation in specific feature map
- ✓ Transpose weight matrix of trained model to invert mapping
- ✓ Use ReLU after filtering

Example
Transposed convolution, fractionally strided convolution or 'deconvolution'
no padding, stride 2, kernel 3 x 3



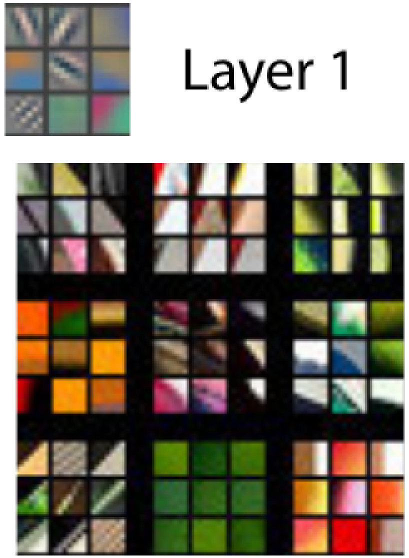
Paul-Louis Pröve,
Towards Data Science

Deconvolutional Network (DeconvNet)



Visualization using DeconvNet

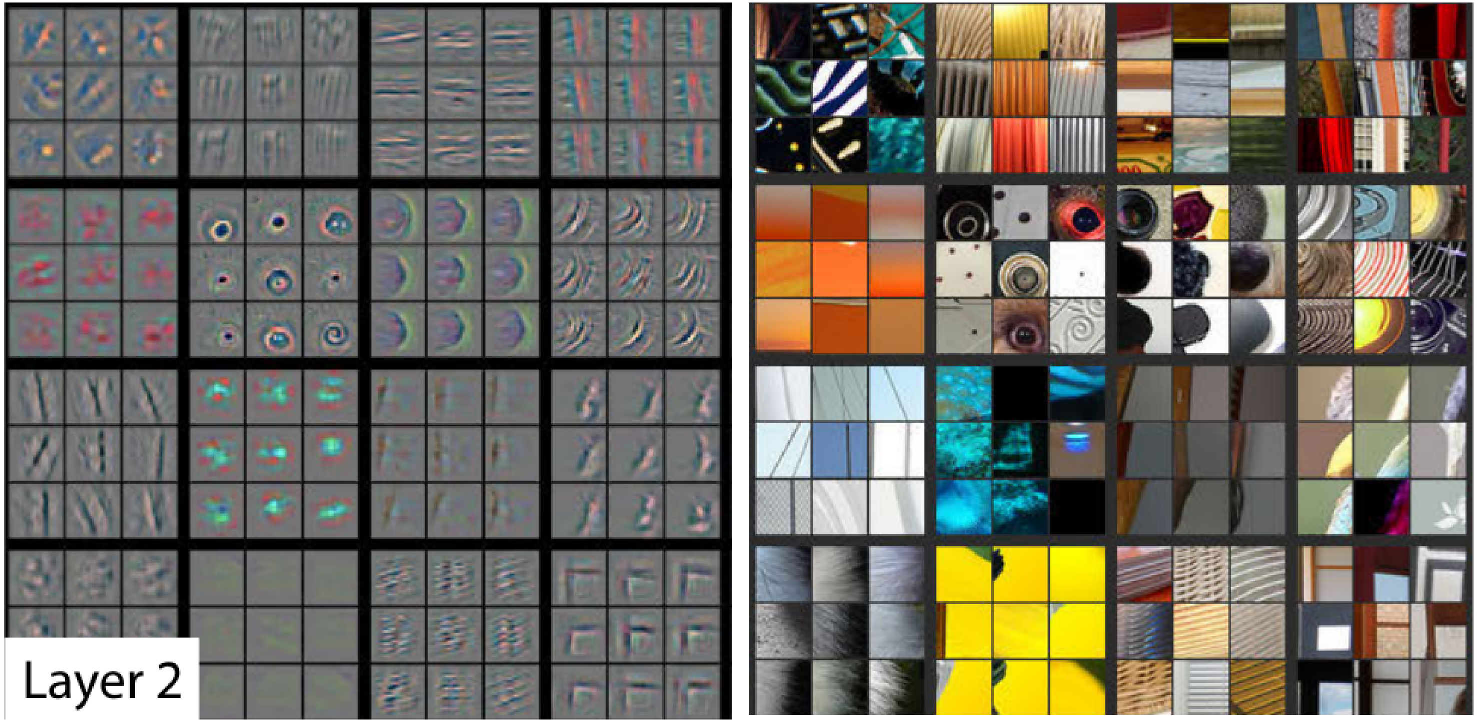
Visualized feature



Cut-outs of samples which create high activation in the specific feature map

Visualized feature

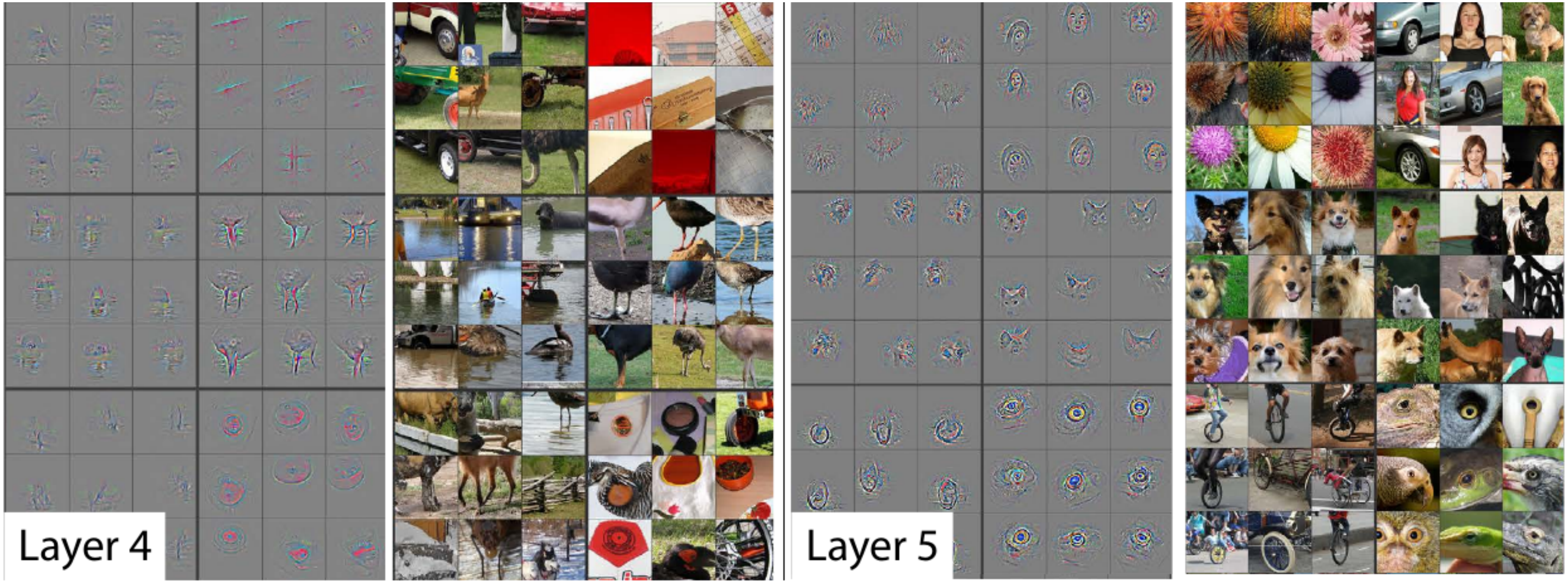
Zeiler, Fergus: Visualizing and Understanding Convolutional Networks



Cut-outs of samples which create high activation in the specific feature map

Visualization using DeconvNet

Visualized feature | Sample cut-out | Visualized feature | Sample cut-out



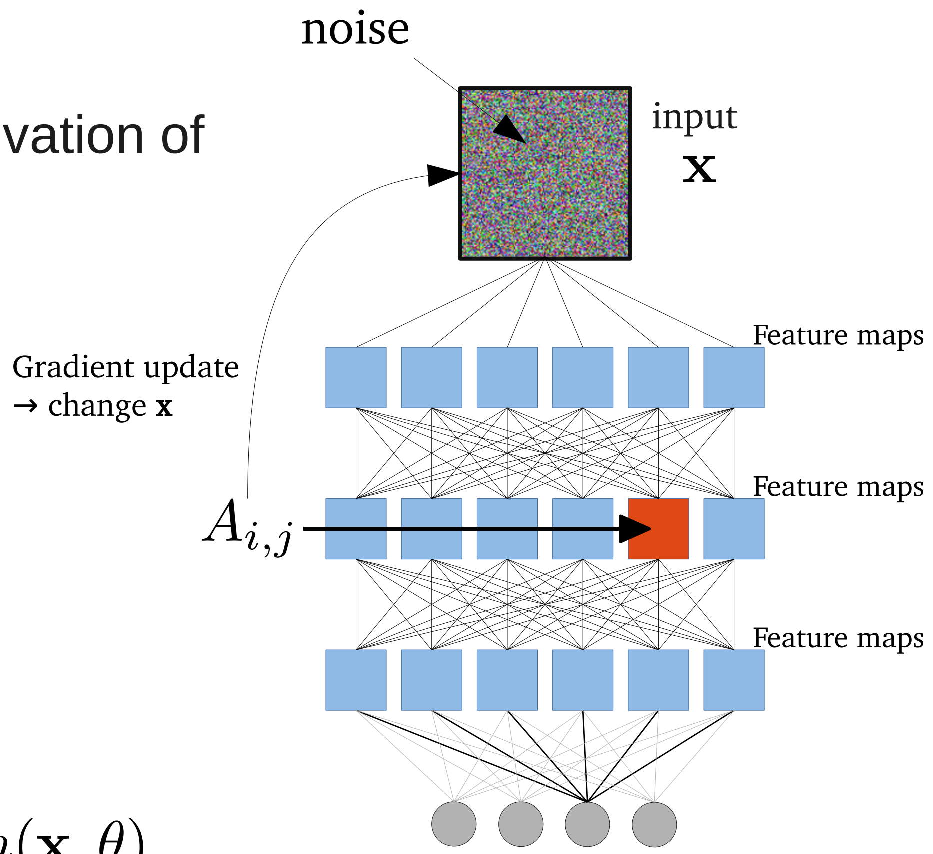
Zeiler, Fergus: Visualizing and Understanding Convolutional Networks

- Layer representation show feature hierarchy → features become more complex
  - Feature semantic becomes more specific (separation more class specific)

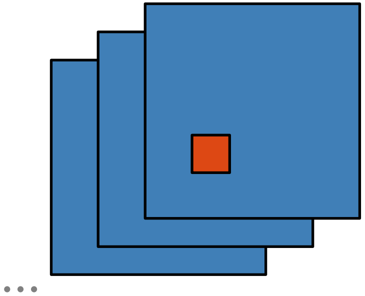
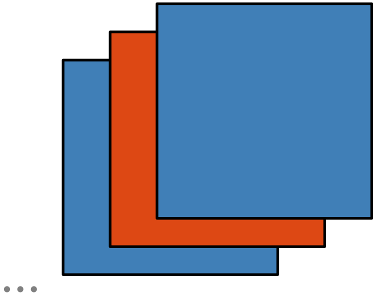
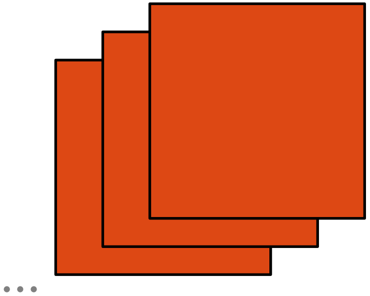
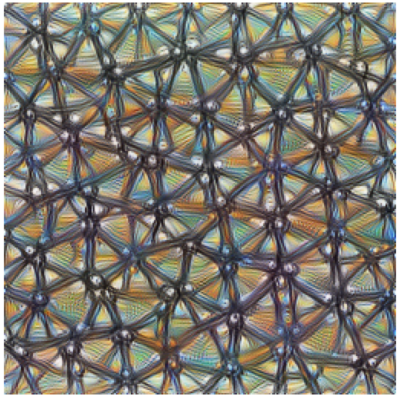
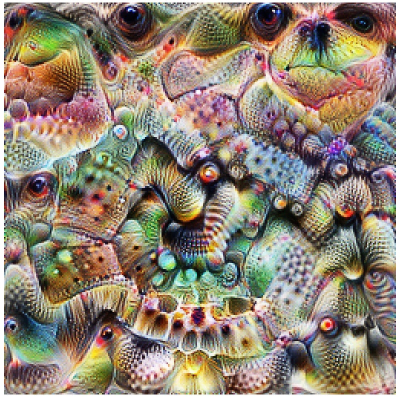
Activation maximization

Idea:

- Construct pattern which maximizes the activation of a specific feature map
- Model f_θ pre-trained, weights θ fixed
- Find $\tilde{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}} \; h(\mathbf{x}, \theta)$
- $h(\mathbf{x}, \theta) = \sum_{i,j} A_{i,j}(\mathbf{x}, \theta) + b$
- Start from noise
  - ➔ perform gradient **ascent** $\mathbf{x}' \to \mathbf{x} + \alpha \frac{dh(\mathbf{x}, \theta)}{d\mathbf{x}}$

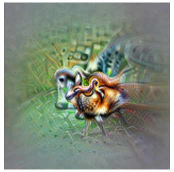


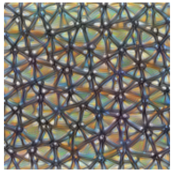
Activation Maximization

	neuron	channel	layer (deep dream)
objective			
obtained visualizations			

Summary feature visualization

- Approach to gain understanding of **trained** model → introspection of features
- Tracking signal propagation through model not comprehensible for humans
  - ➔ more sophisticated approaches needed for interpretation
- Local understanding of the model:
  - introspect model around given sample (DeconvNet)
- Towards global model understanding:
  - search/generate pattern that maximize feature response using differentiable of neural networks





- Visualization confirms hypothesis of feature hierarchy, DNNs learn:
  - decomposition of input space into modular hierarchical structure
  - probabilistic mapping between high-level features

Milestones: model visualization

Understanding the building blocks of CNNs → 'What is a feature?'

- ✔ No visualization of model, but to what it is sensitive to
- ✔ Deep learning is form of representation learning
  → nodes, feature maps, and layers dispose distinct abstraction level
  → visualization of CNNs confirms hierarchy of features
- ✔ DNNs don't think! If CNNs work similarly to human visual cortex is under debate
- ✔ Challenge: interplay of nodes, feature maps, layers
- ✔ Several methods available (global and local):
  - ✔ propagation-based (DeconvNet)
  - ✔ gradient-based (activation maximization) / gradient estimate **w.r.t. input**

Analysis of predictions & feature attribution

arXiv:1602.04938

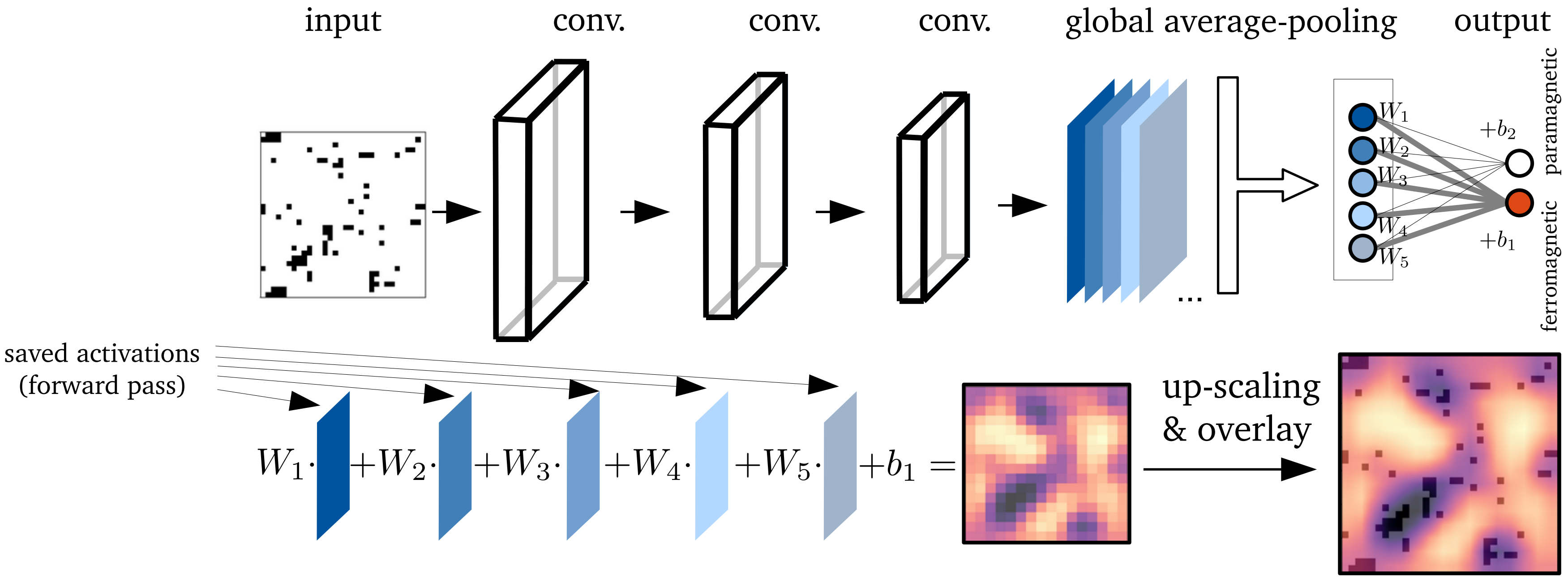
(a) Husky classified as wolf (b) Explanation

“Why is my model predicting a certain class / value?”

“What influences the model’s reasoning most?”

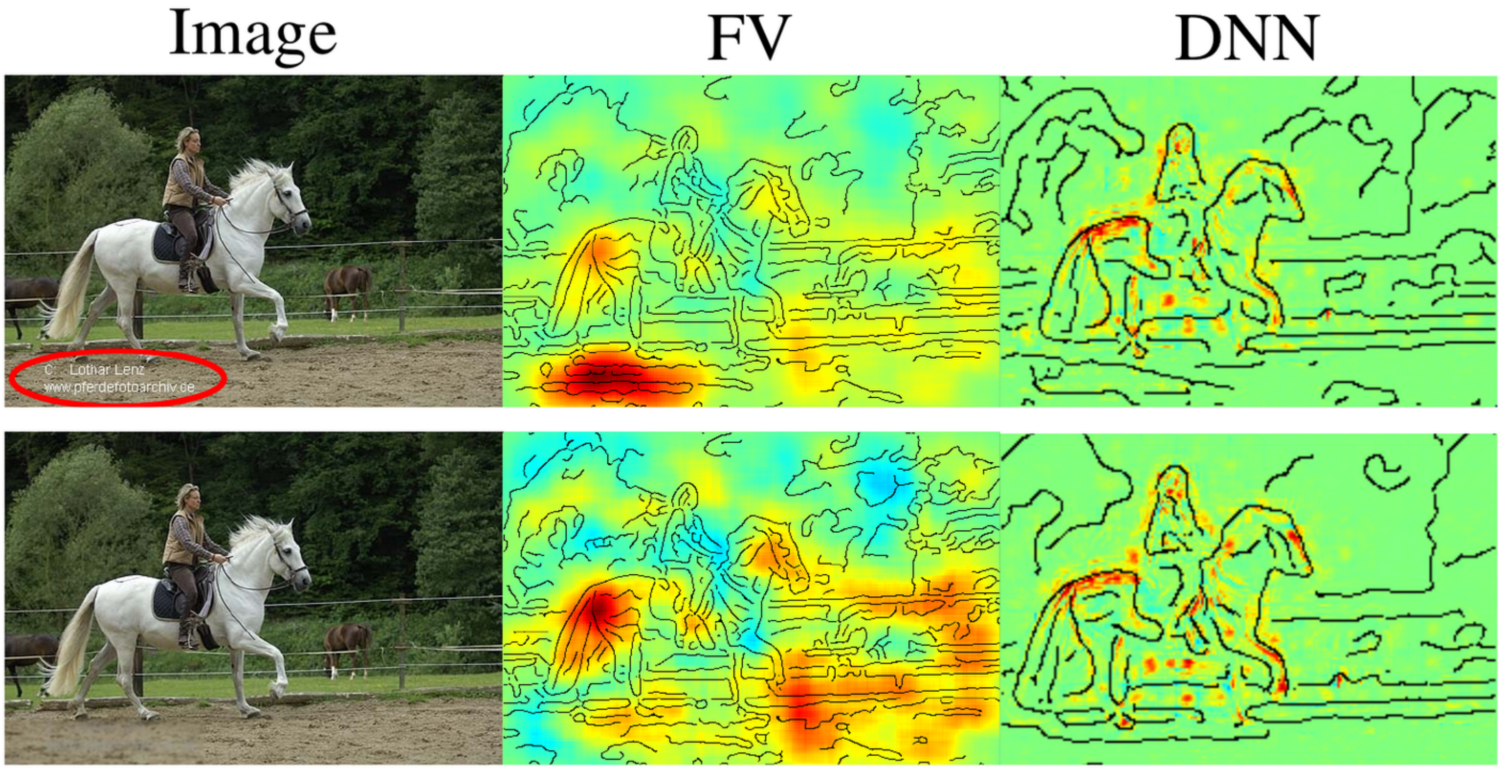
Predictions

Discriminative Localization



- Class activation map (CAM) indicates how the output of the last CNN layer is used for classification
- Generated by scaling of feature maps and up-sampling (interpolation)
  - Limited to particular architecture (GAP, single fully-connected layer)

Semantic Misinterpretation



Bach et. Al. - Analyzing Classifiers: Fisher Vectors and Deep Neural Networks, arXiv:1512.00172

arXiv:1602.04938

(a) Husky classified as wolf (b) Explanation

How important is the context?

Saliency Maps

Idea:

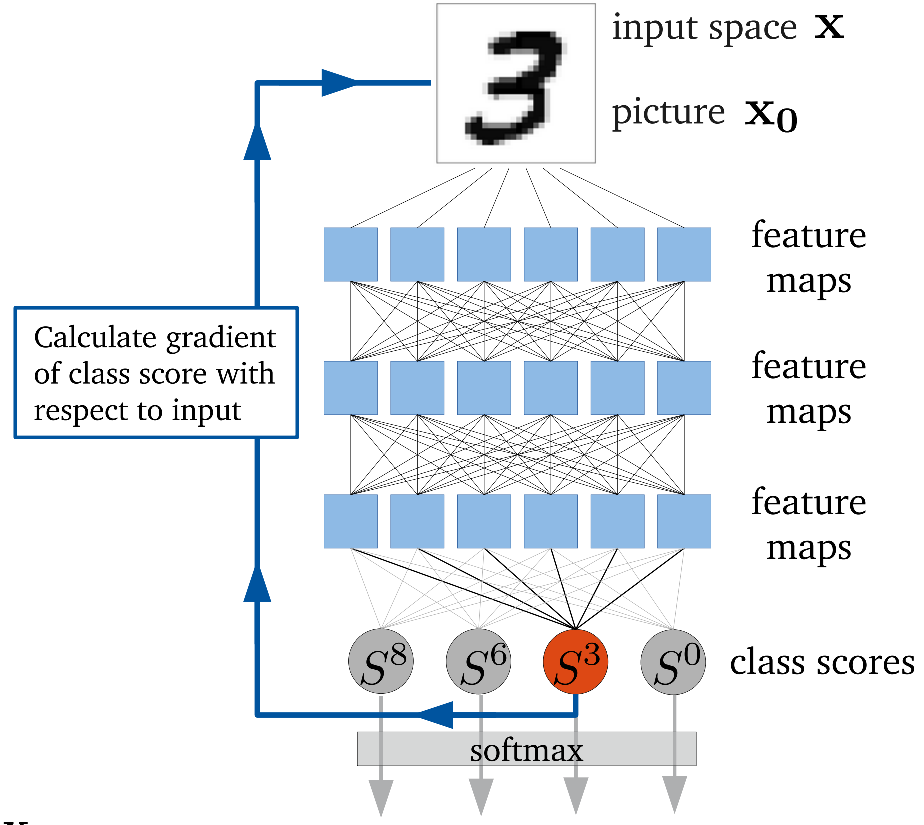
- *"What influences the class score at most?"*
- ➔ Important pixels have large gradients
- ➢ Fix network parameters
- ➢ Rank pixel importance of input space

- DNN $f(\mathbf{x})$ outputs score $S_c(\mathbf{x})$ for image $\mathbf{x}$
- Compute 1st order Taylor expansion

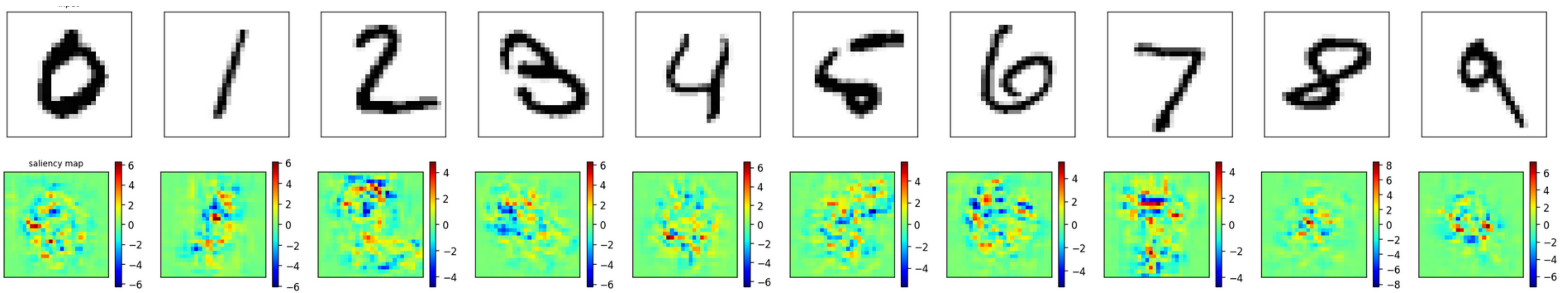
$$f(\mathbf{x}) = S_c(\mathbf{x}) \approx \mathbf{w}^T\mathbf{x} + \mathbf{b}$$

- Resulting map of gradients: $\mathbf{w} = \left.\frac{\partial S_c}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x_0}}$

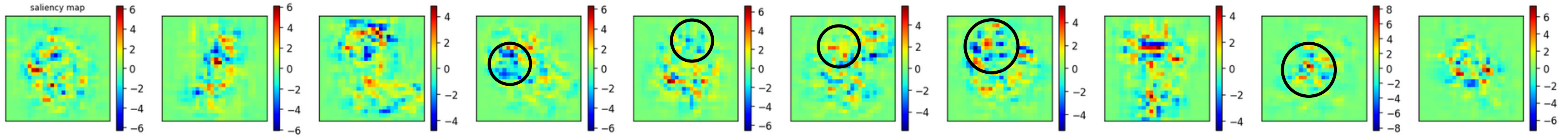
Map has dimension of input image



Saliency Maps MNIST



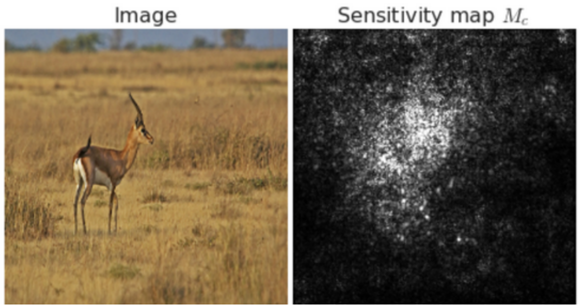
- Negative gradient: intensity increase of respective pixel → reduce class score
- Positive gradient: intensity increase of respective pixel → raise class score



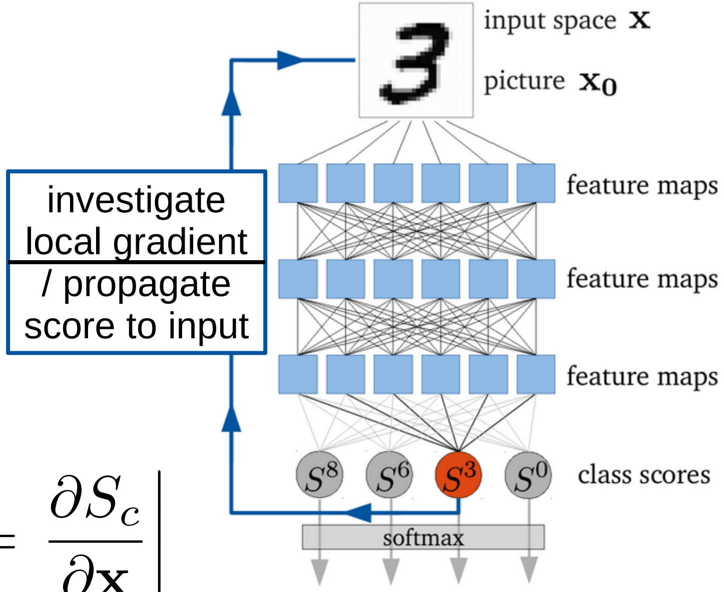
Prediction analysis

Sensitivity analyses (saliency maps ArXiv/1312.6034)

- Study to what the DNN is **sensitive** to
- Fast and versatile approach
- Limited expressiveness
  - does not explain prediction itself but rather how it will it may change
  - locally estimated gradients are noisy



arXiv:1706.03825



$$\mathbf{w} = \left.\frac{\partial S_c}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}$$

https://doi.org/10.1142/12294

- Related (advanced) approaches: SmoothGrad, Integrated gradients

Prediction analysis

Instead of studiying sensitivity, investigate how predictions are formed

Attribution analyses

- Fulfill **completeness:**

prediction

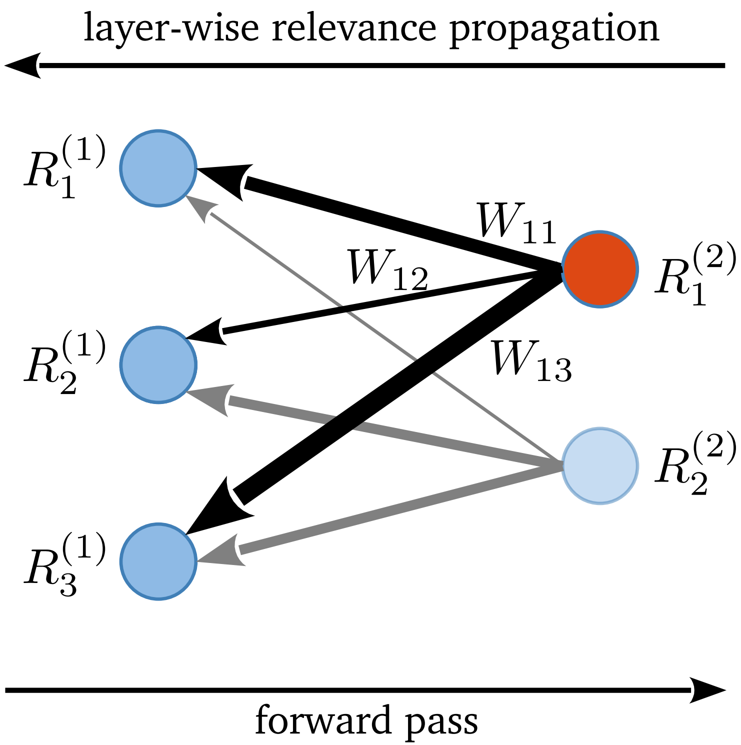
$$\sum_i R_i^c = S^c(\mathbf{x_0})$$

sum runs over all inputs   relevance/attribution

- Sum over all input relevances = prediction
  - ➔ ranks input by it attribution to the prediction
- Common methods:
  - Layer-wise relevance propagation, IntegratedGradients, DeepLIFT

Layer-wise relevance propagation (LRP)

- Re-distribute activation to input
- Designed for DNNs with ReLU activation



- ε-LRP: propagation rule for ReLU networks

$$R_i^{(l-1)} = \sum_j \frac{a_i W_{ij}}{\epsilon + (b_j + \sum_{i'} a_{i'} W_{i'j})} R_j^{(l)}$$

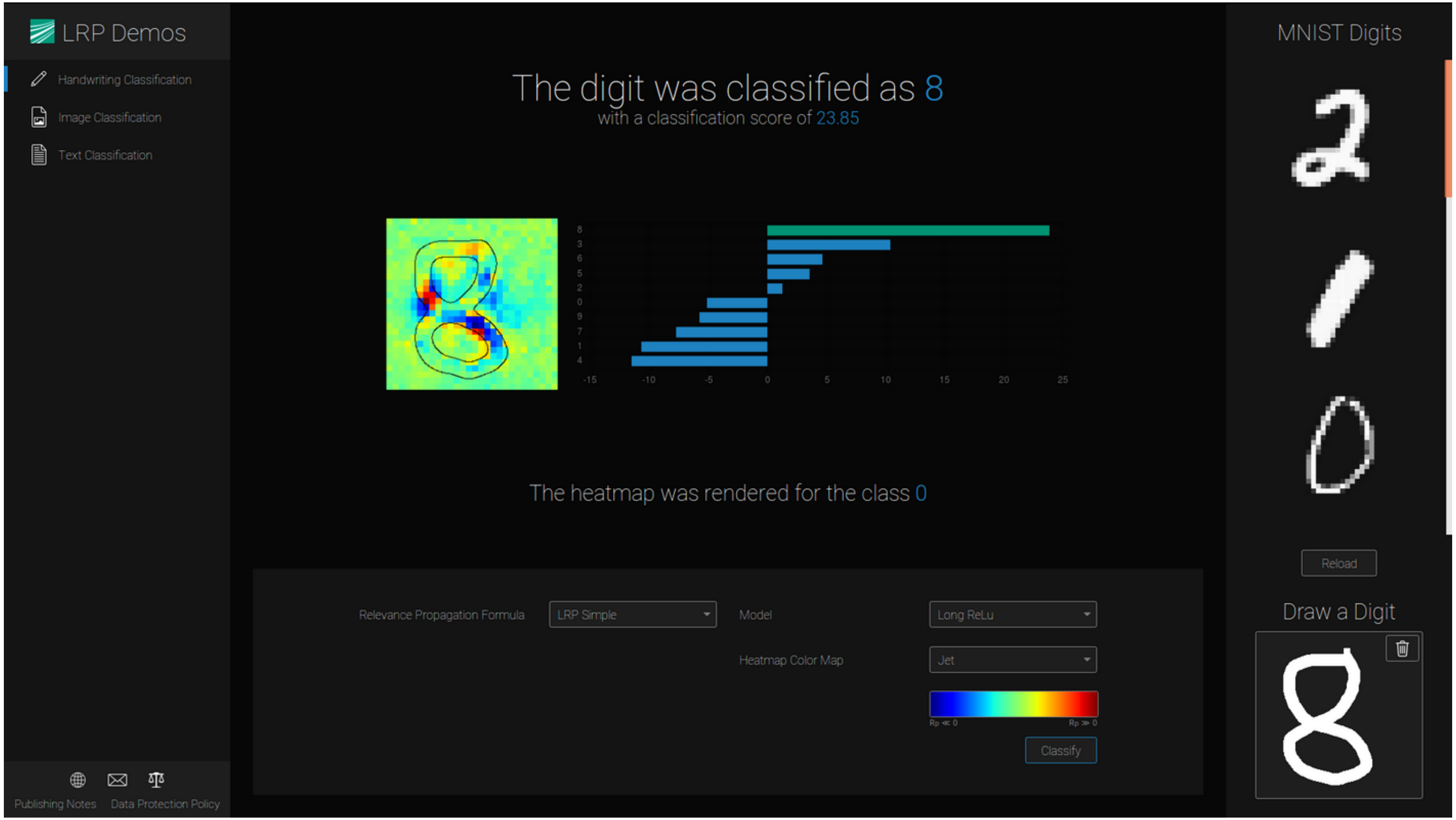
obtained relevance

bias

activations determined in forward pass

- ε controls re-distribution (numerical stability)
  - small: high sensitivity, tend do be noisy
  - larger: less noisy, sparser, absorb weak relevances

DEMO - Handwriting



https://lrpserver.hhi.fraunhofer.de/handwriting-classification

Summary Prediction Analysis

Interpretation of model predictions – *“What causes the certain prediction?*

Sensitivity analysis – *“To which input my prediction is most sensitive?”*

- Investigate sensitivity of the model locally around given input
  e.g., saliency maps (gradient-based)
- Describe sensitivity and not predictions itself

Attribution analyses – *“Which input contributed how much to the output?”*

- Completeness criterion (attributions sum up to prediction)
- Study input relevances to the prediction
  e.g., LRP, IntergratedGradients, DeepLift, Discriminative Localization

Perturbation-based

- Perform perturbations of the input
- Costly, meaningful baseline important

Milestones: prediction analysis

Understanding the predictions of deep neural networks

- ✔ Verification of model, understanding the algorithm
- ✔ Learn about the data (important inputs, selection of features, segmentation)
- ✔ Interpretation approaches
  - ✔ sensitivity analyses (to which input my prediction is most sensitive)
  - ✔ attribution analysis (which input contributes how much to the prediction) fulfill completeness, more sophisticated approaches
- ✔ Popular classes of techniques:
  - ✔ propagation-based, perturbation-based, gradient-based

Questions

- DeconvNet: "*Why are the images becoming larger when going deeper?*"
  - Increasing receptive field of view

- Discriminative localization: "*Why the network is limited to a single FC layer?*"
  - Inversion of non-linearity (to increase network capability by adding a layer) needed to be considered

- Code: "*What causes the black visualized features maps?*"
  - ReLU non-linearity, dying ReLU (no gradient for negative values)

- Activation Maximization: "*Why is maximization more common then minimization?*"
  - For DNNs with ReLUs, negative activations are cut away

Summary: understanding deep networks

Feature visualization

- understanding the model & building blocks – *“What is learned by the network?”*

Prediction analysis

- interpret a prediction – *“Why a specific pattern caused a certain reconstruction”*

> Introspection techniques are similar and can generally applied vice versa
> (applied at output vs. applied at feature level)

Fast growing field of research

- ➔ study your network using a collection of techniques
- understand your model, debug your architectures

- ➢ Software libraries: iNNvestigate, DeepExplain, Captum

!Warning: Be cautious to disentangle observations and human implications!

Milestones (general)

- ✔ Neural networks are not black boxes → but challenging to interpret
- ✔ Interpretability involves data, model, predictions

- ✔ Diverse methods for network introspection exist (examine various aspects)
- ✔ Introduced techniques in feature and prediction analyses are strongly related
  - ✔ for model visualization (application at feature level: node, feature map, layer)
  - ✔ for prediction analyses (application at output / class score (before softmax))

- ✔ Introspection possible for various architecture (CNNs particular simple)
- ✔ Interpretation involves humans (possible bias)

Structure of the lecture

General: Top down approach (features → output)

- start with visualization of features, then investigate predictions
- include 3 multimedia breaks (one after each block) to let the audience wake up

(in principle easy to switch order by interpreting output as feature)

- Simple to more complex
  - Visualization: plot filters → DeconvNet → activation maximization
  - Predictions: discriminative localization → saliency maps → LRP
    sensitivity → attribution
- Tutorial:
  - one example for each part
  - hard to find easy examples (implementation is relatively complex)

Multimedia resources

- Feature Visualization
  - MNIST foward CNN: https://www.cs.ryerson.ca/~aharley/vis/conv/flat.html
  - Visualization of Features: https://distill.pub/2017/feature-visualization/
  - Model Collection with Visualization: https://microscope.openai.com/models

- Prediction Analyses
  - LRP MNIST: https://lrpserver.hhi.fraunhofer.de/handwriting-classification
  - Baselines IntergratedGradients: https://distill.pub/2020/attribution-baselines/

Tutorial

- Open tutorial page
  https://github.com/jglombitza/Introspection_tutorial
- open Colab link and login with your Google Account

- **Exercise 1:** model introspection
  - model visualization using activation maximization

  Open in Colab

- **Exercise 2:** introspection of predictions
  - implement discriminative localization

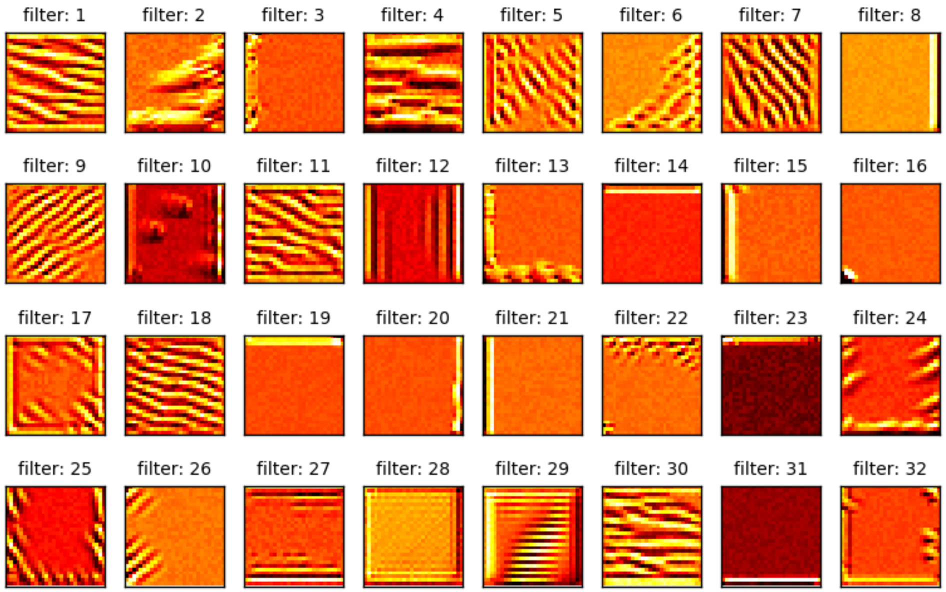
  Open in Colab

Task 1 – Code

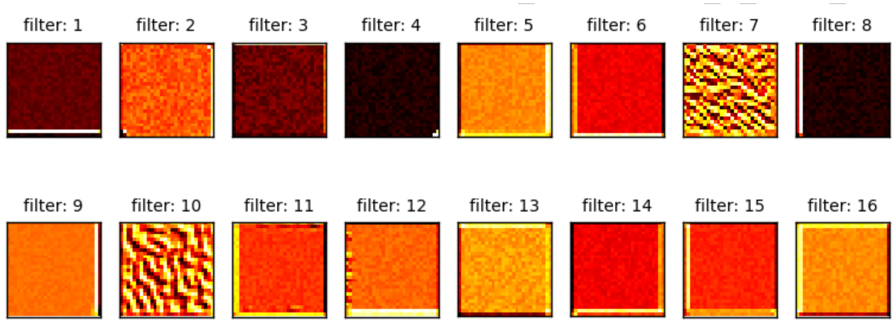
```
model = models.load_model("./my_mnist_model.h5")
layer_names = ['conv2d_1', 'conv2d_2', 'conv2d_3', 'conv2d_4']

for layer_name in layer_names:
    layer_output = layer_dict[layer_name].output
    sub_model = models.Model([model.inputs], [layer_dict[layer_name].output])

    for filter_index in range(layer_output.shape[-1]):
        input_img = keras.backend.variable(np.random.uniform(0,1, (1, 28, 28, 1)))

        for i in range(gradient_updates):

            with tf.GradientTape() as gtape:
                layer_out = sub_model(input_img)
                loss = keras.backend.mean(layer_out[..., filter_index])
                grads = gtape.gradient(loss, input_img)
                input_img.assign_add(step_size * normalize(grads))
        visualized_filter = deprocess_image(input_img.numpy()) # cast to numpy array
```

Task 1 – Results

Layer 1

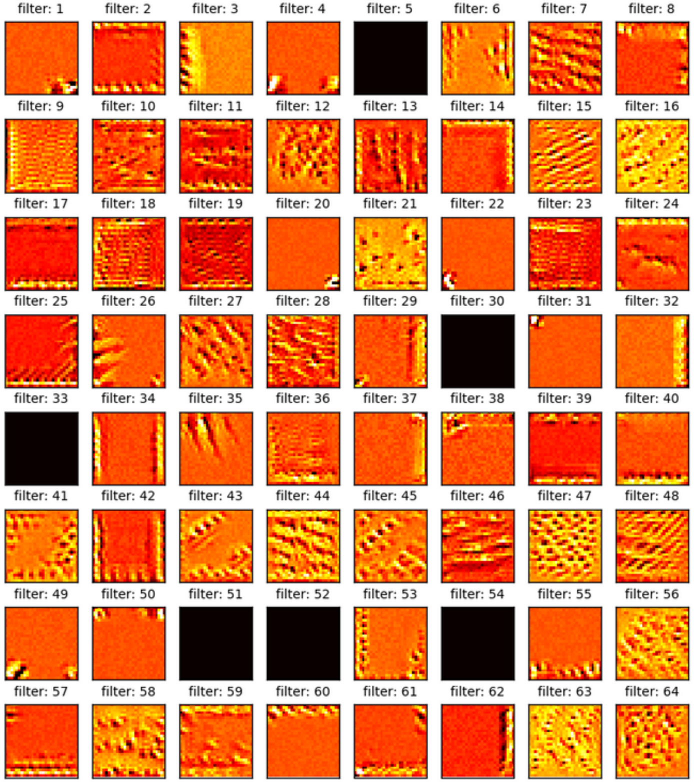


Layer 2



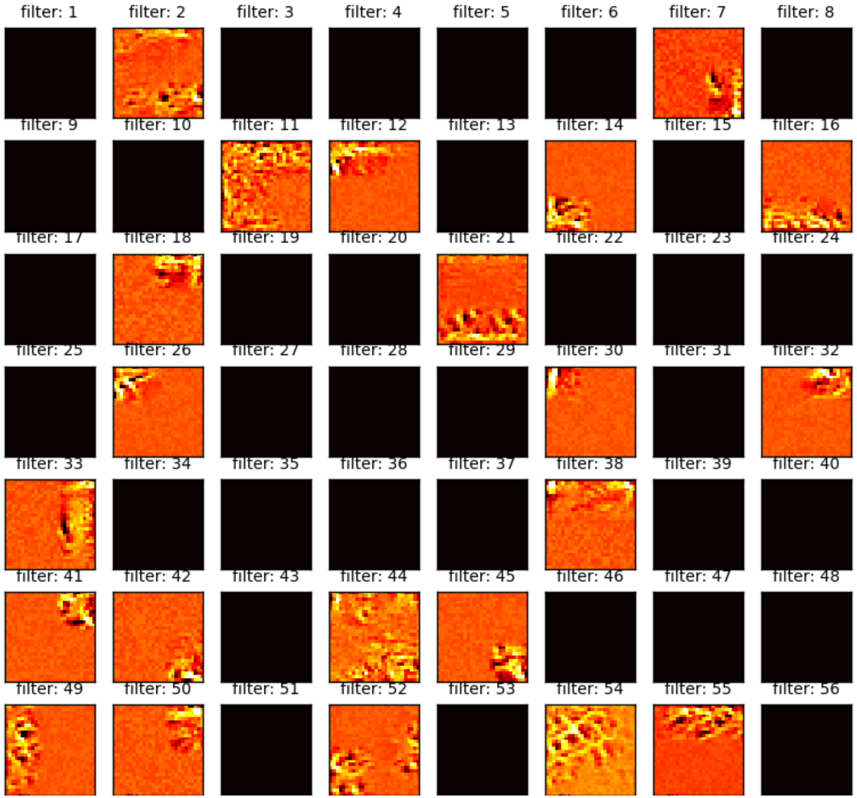
Task 1 – Results

Layer 3



Layer 4

dead nodes caused by ReLU



Task 2 – Code

- Which spatial regions lead to the network's decision?
  I. Use padding to maintain spatial information
  II. Use global pooling to collapse the spatial dimensions → probabilistic mapping
- ➢ Look how the last convolutional layer output is used for the decision

```
model = models.Sequential([InputLayer(input_shape=(32, 32, 1)),
    layers.Conv2D(8, (3, 3), padding='same', activation='relu'),   # (32, 32, 8)
    layers.MaxPooling2D((2, 2)),                                    # (16, 16, 8)
    layers.Conv2D(16, (3, 3), padding='same', activation='relu'), # (16, 16, 16)
    layers.Conv2D(32, (3, 3), padding='same', activation='relu'), # (16, 16, 32)
    layers.Dropout(0.25),
    layers.GlobalAveragePooling2D(),                                # (1, 1, 32)
    layers.Dense(2, activation='softmax')])
```

```
F = ...  # output of last conv layer
W, b = model.layers[7].get_weights()  # weights of final dense layer
M = np.einsum('ixyz,zc->ixyc', F, W) + b  # class activation maps
```

– BACKUP –

Jonas Glombitza

RWTH Aachen

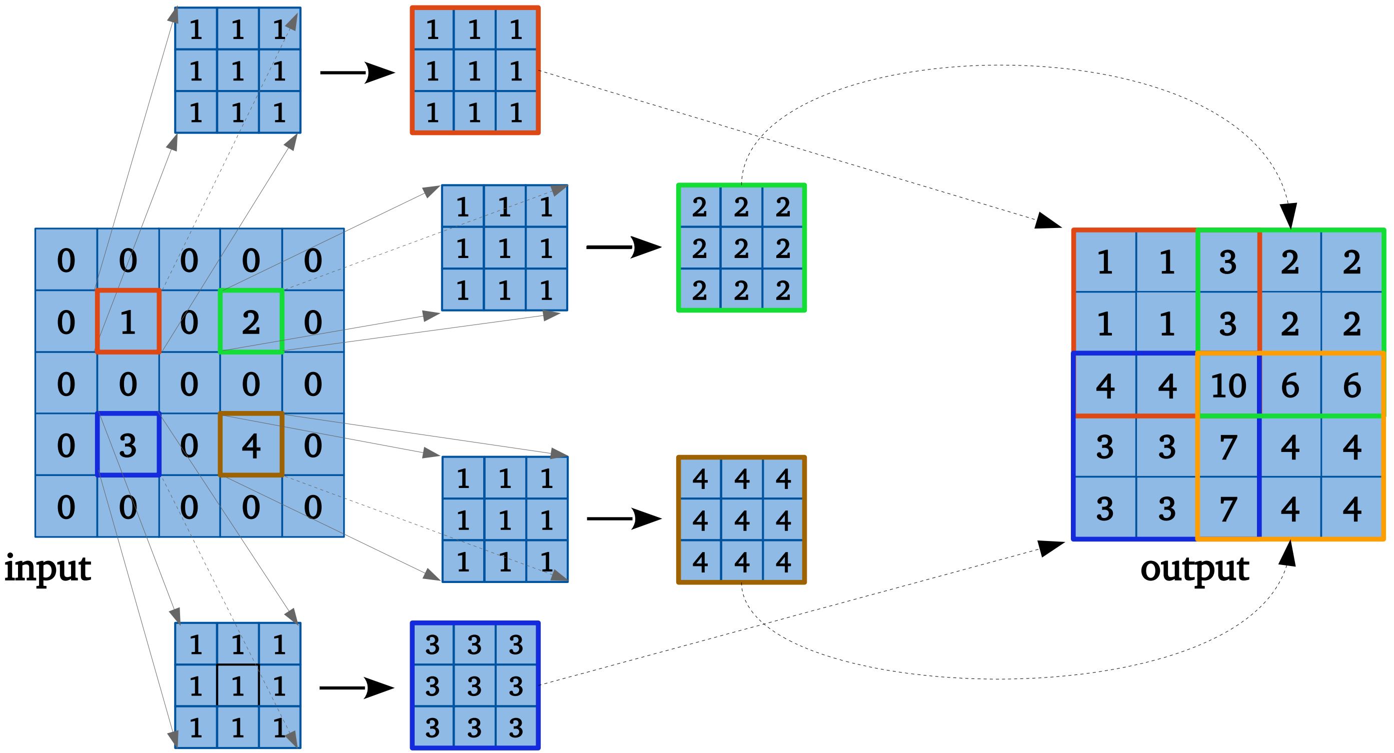
Transposed convolution
strides = (2, 2)
no zero padding

input image

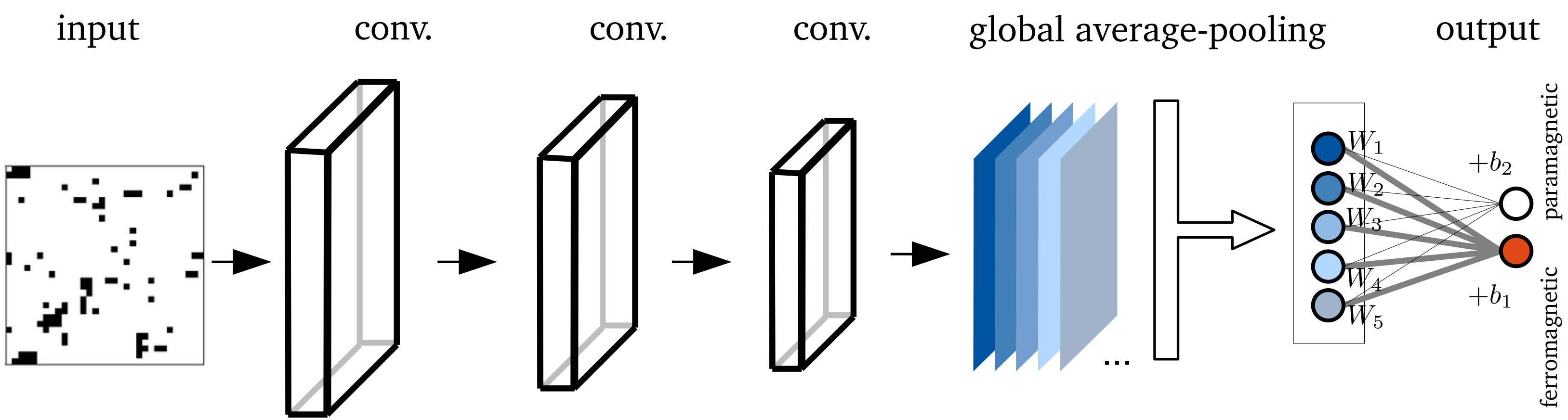
1	2
3	4

filter

1	1	1
1	1	1
1	1	1



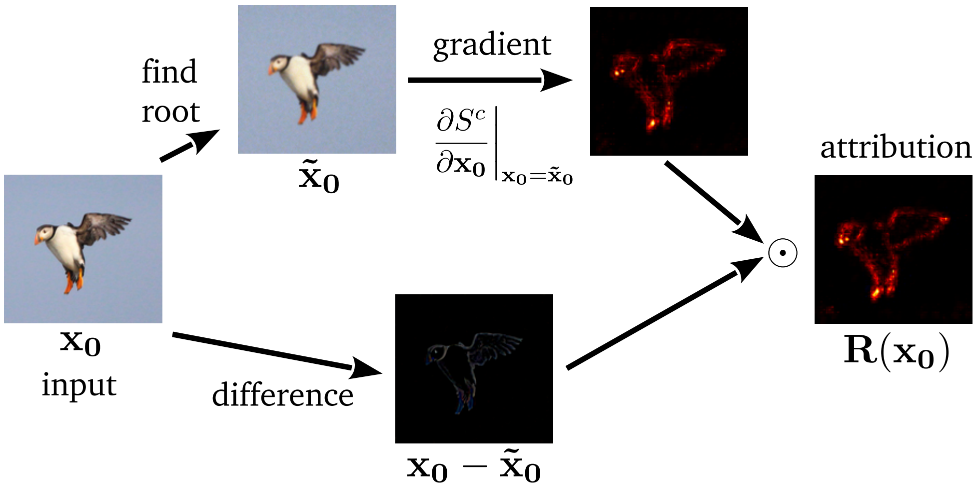
GradCAM and CAM



- Discriminative localization with CAMs requires global average pooling layer
  - enables to stack feature maps and scale with weights of the final layer
  - breaks for more complex architectures (e.g., by adding a fully-connected layer)

→ Fuse technique with gradient-based sensitivity analyses
  - propagated gradients to first CNN layers and built GradCAM
  - → technique more flexible

Deep Taylor Decomposition / Integrated Gradients



$$S^c(\mathbf{x}_0) = \underbrace{S^c(\tilde{\mathbf{x}}_0)}_{0} + \underbrace{\left(\left.\frac{\partial S^c}{\partial \mathbf{x}_0}\right|_{\mathbf{x}_0=\tilde{\mathbf{x}}_0} \right) \cdot (\mathbf{x}_0 - \tilde{\mathbf{x}}_0)}_{\mathbf{R}} + \mathcal{O}(\ldots).$$

$$\mathbf{R}^c = (\mathbf{x} - \mathbf{x}_0) \odot \int_0^1 \left.\frac{\partial S^c(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right|_{\tilde{\mathbf{x}}=\mathbf{x}'+\alpha\cdot(\mathbf{x}_0-\mathbf{x}')} d\alpha,$$