



**FAU**

FRIEDRICH-ALEXANDER-  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
SCHOOL OF ENGINEERING

# Visualization

**A. Maier, K. Breininger, L. Mill, N. Ravikumar, T. Würfl**

Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

June 06, 2018



# Outline

## Motivation

## Network Architecture Visualization

## Visualization of Training

## Visualization of Parameters

Simple Parameter Visualization

Gradient-Based Visualization

Parameter Visualization via Optimization



**FAU**

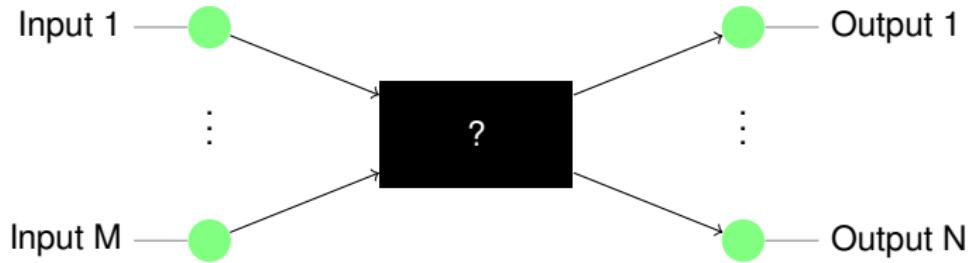
FRIEDRICH-ALEXANDER-  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
SCHOOL OF ENGINEERING

# Motivation



## Motivation - Why do we need visualization?

- Neural networks often treated as black box:



- Today: How can we understand and communicate the inner-workings of a network?

## Motivation

### An incomplete list why understanding and visualization matters:

- Communicate architectures between researchers
- Identify issues during training (not converging, dying ReLUs)
- Identify faulty training/test data
- Understanding how, why and what networks learn

## Motivation

### An incomplete list why understanding and visualization matters:

- Communicate architectures between researchers
- Identify issues during training (not converging, dying ReLUs)
- Identify faulty training/test data
- Understanding how, why and what networks learn

### Three (main) types of visualization:

- Architecture
- Training
- Learned parameters/weights
  - Visualize **representation** of data in the network



**FAU**

FRIEDRICH-ALEXANDER-  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
SCHOOL OF ENGINEERING

# Network Architecture Visualization



# Network Architecture Visualization

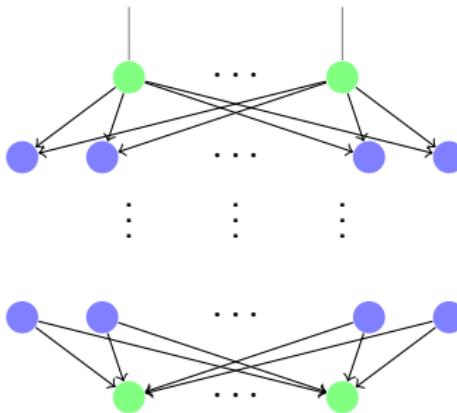
- Important to communicate architectures effectively
- Priors imposed by the architecture may be most important factor for good performance
- Mostly graph-based structures with different granularity
- Compare Lecture 6: Neural Network Architectures

# Network Architecture Visualization

Three categories:

- Node-link diagrams: Neurons = nodes, (weighted) connections = edges
- Block diagrams: Layer = solid block, single connection between layers
- “Others”

# Network Architecture Visualization - The node-link diagram



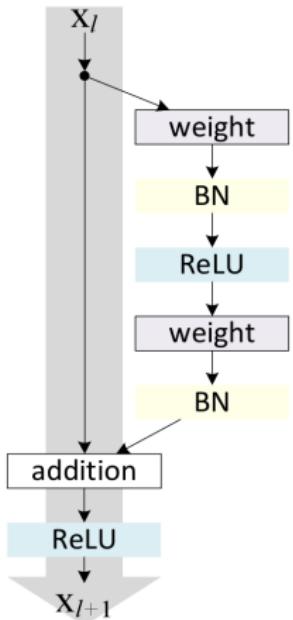
- Detailed representation, focus on connectivity
- Only for small (sub-)networks, building blocks
- Variants, e.g., with explicit weights, recurrent connections, ...

# Network Architecture Visualization

Three categories:

- Node-link diagrams: Neurons = nodes, weighted connections = edges
- Block diagrams: Layer = solid block, single connection between layers
- “Others”

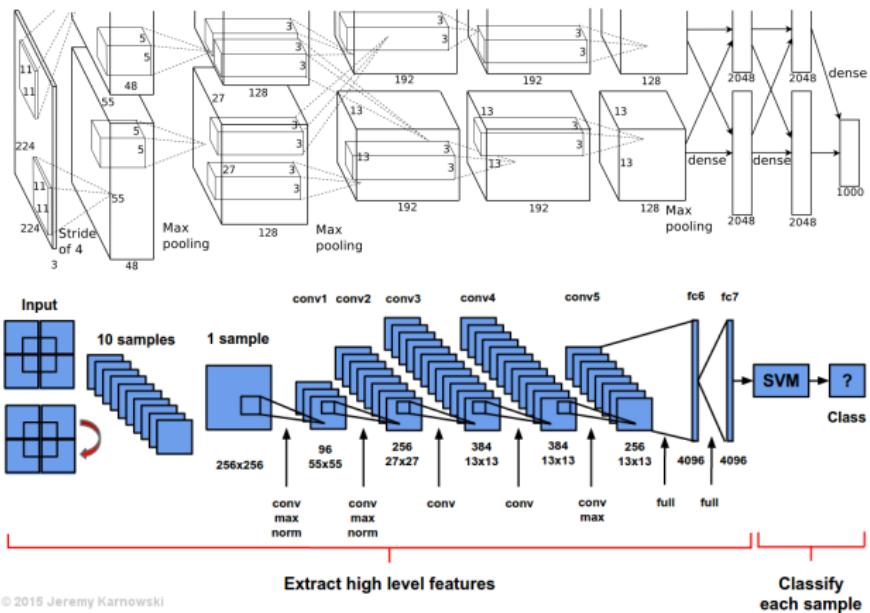
# Network Architecture Visualization - Block diagrams



- Blocks: mathematical operations/layers
- Arrows: **direction of flow** of the data
- Blocks can have different granularity – often hierarchical descriptions
- Textual description of hyperparameters (filter size, # of filters, ...) common

Block diagram of a ResNet-module.

## Network Architecture Visualization - Block Diagrams (cont.)



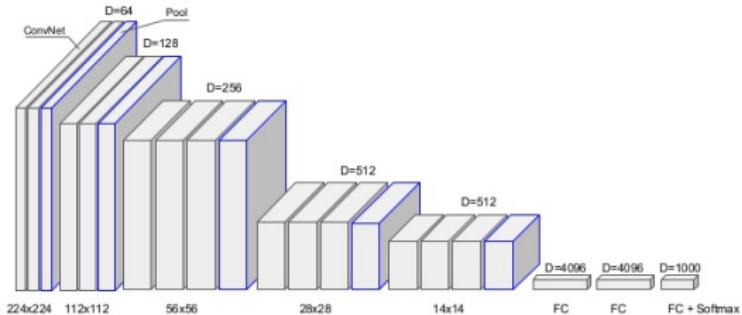
© 2015 Jeremy Karnowski

Two different visualizations for AlexNet

Source: <https://jeremykarnowski.wordpress.com/2015/07/15/alexnet-visualization/>, [6]

## Network Architecture Visualization - Block diagrams (cont.)

- Most common representation, but many variants  
→ 3D/pseudo-3D layers, receptive fields, colors, ...
- Recommendation: Pick one that clearly represents what you want to show  
→ Good combination of text & figure is key!
- Most DL libraries have tools to display defined computational graphs  
→ Good for debugging, usually not good for reports



Block diagram of VGG16

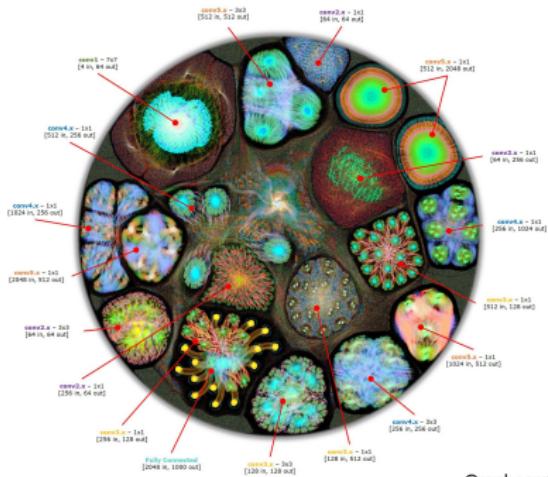
# Network Architecture Visualization

Visualization strategies break down into three categories:

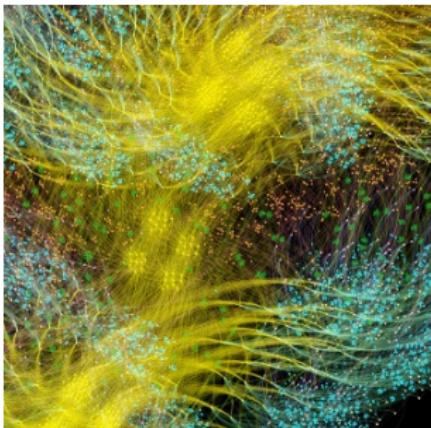
- Node-link diagrams: Neurons = nodes, weighted connections = edges
- Block diagrams: Layer = solid block, single connection between layers
- “Others”

## Network Architecture Visualization - Others

- Deep Visualization [14]: Combination of architecture & parameter visualization (see next sections).
- Graphcore Poplar™ – fancy graph visualization:



Graphcore



Resnet 50 visualized with Graphcore Poplar.

Source: <https://www.graphcore.ai/posts/what-does-machine-learning-look-like>



**FAU**

FRIEDRICH-ALEXANDER-  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
SCHOOL OF ENGINEERING

# Visualization of Training



## Visualization of Training

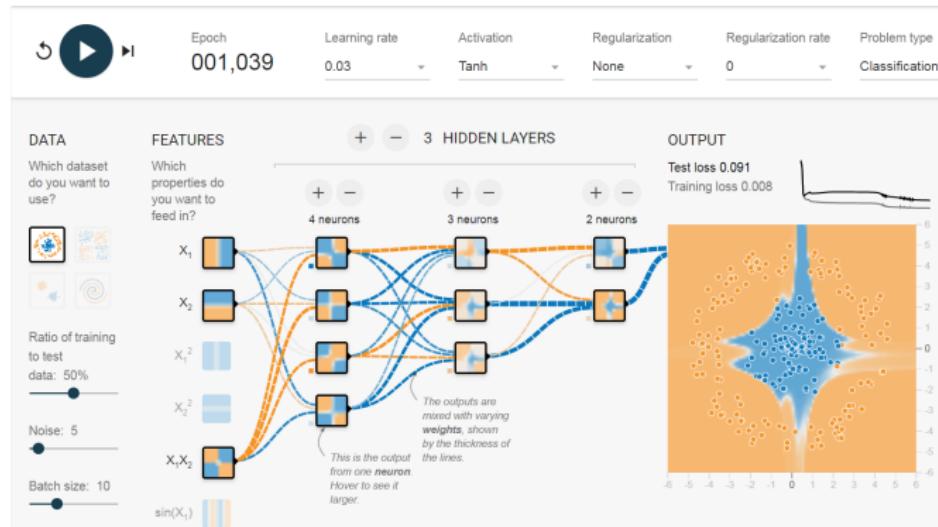
Lots of interesting information available during training:

- Input data (images, text, ...)
- Parameters (weights, biases, ...)
- Hidden layer data (activations, hidden states, ...)
- Output data (classification, loss curves, ...)

Tracking information helps!

→ Debugging, improve model design, ... (see also Lecture 5 - Common Practices)

# Visualization of Training

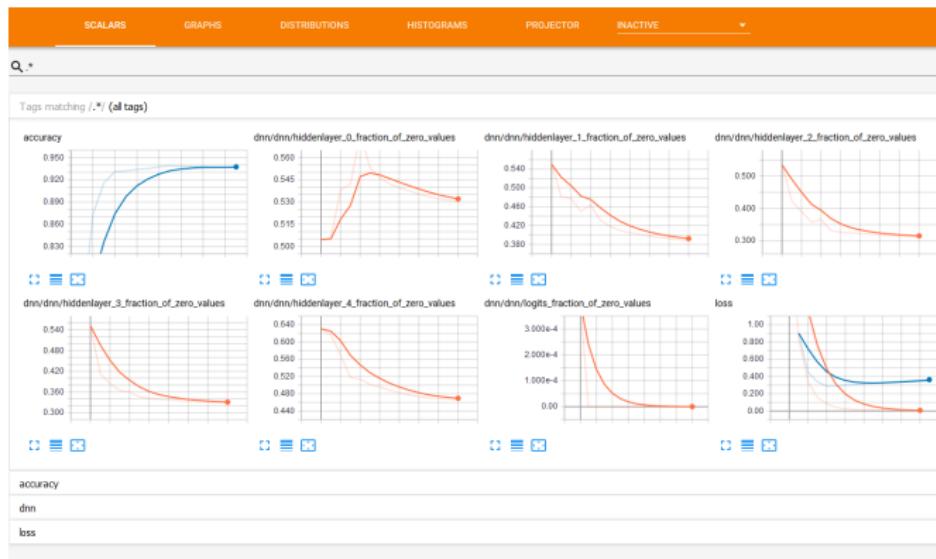


Tensorflow playground

Only 2D-toy examples, but nice concepts, e.g., interactive loss curve, visualization of decision boundary, importance of weights

Source: <http://playground.tensorflow.org>

# Visualization of Training



Tensorboard example (TensorFlow)

Most DL libraries provide tools to record and monitor training - **use them!**



**FAU**

FRIEDRICH-ALEXANDER-  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
SCHOOL OF ENGINEERING

# Visualization of Parameters



## Motivation

- Networks learn **representation** of the training data → Question: What happens with the data in our network?

## Motivation

- Networks learn **representation** of the training data → Question: What happens with the data in our network?
- Additional question: Why should we care?

## Motivation

- Networks learn **representation** of the training data → Question: What happens with the data in our network?
- Additional question: Why should we care?
- Answer: To investigate unexpected/unintuitive behavior:
  - Adversarial examples

## Motivation

- Networks learn **representation** of the training data → Question: What happens with the data in our network?
- Additional question: Why should we care?
  - Answer: To investigate unexpected/unintuitive behavior:
    - Adversarial examples
    - Network performs well in the lab, but fails in the wild

## Motivation

- Networks learn **representation** of the training data → Question: What happens with the data in our network?
- Additional question: Why should we care?
  - Answer: To investigate unexpected/unintuitive behavior:
    - Adversarial examples
    - Network performs well in the lab, but fails in the wild
    - Potential causes: Focus on “wrong” features, different noise properties, ...

## Motivation

- Networks learn **representation** of the training data → Question: What happens with the data in our network?
- Additional question: Why should we care?
  - Answer: To investigate unexpected/unintuitive behavior:
    - Adversarial examples
    - Network performs well in the lab, but fails in the wild
    - Potential causes: Focus on “wrong” features, different noise properties, ...
    - (Anecdotal) example: Identification of tanks in photos [2], [4]

## Motivation - Confounds



- Task: Identify whether image shows a tank
- Problem: All tank images recorded on cloudy days, all non-tank images on sunny days

Source: [2]

## Motivation - Confounds

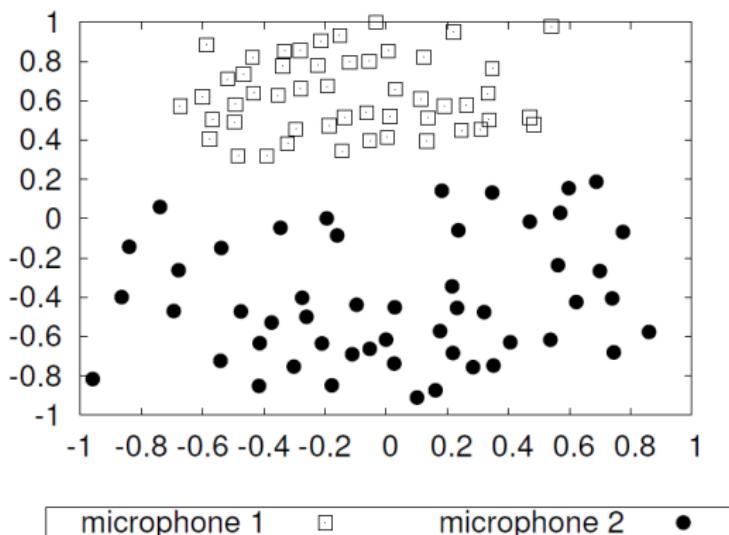


- Task: Identify whether image shows a tank
- Problem: All tank images recorded on cloudy days, all non-tank images on sunny days
- Network learned the **correlated feature** weather, **not** to identify the tank!
- **Important:** Not a fault in the learning algorithm, but in the data!

Source: [2]

## Motivation - Confounds (cont.)

Example: Speech recordings with two microphones [8]



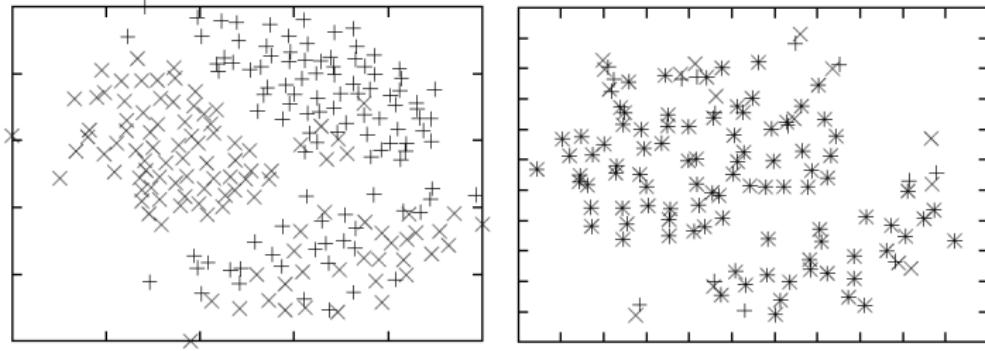
Recordings of the same speaker group with two different microphones

→ ML will focus on the most discriminative features!

Source: Maier et al. [8]

## Motivation - Confounds

- If confounder is **known**, we may be able to correct for it:
- Maier et al. [8]: QMOS – Use knowledge of same participants

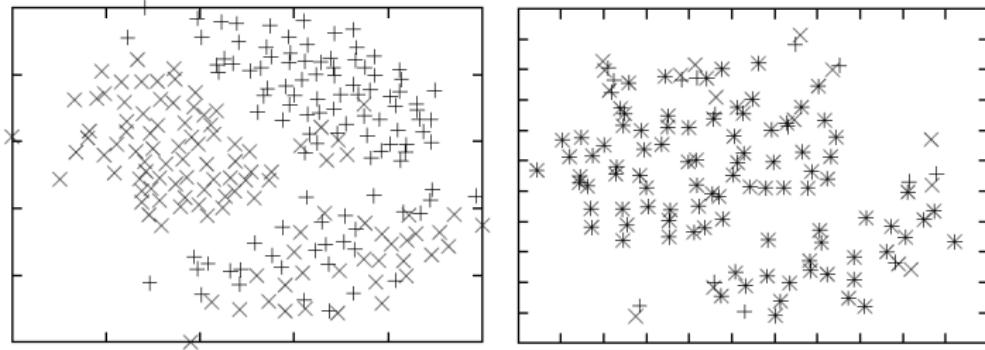


Speaker visualization before (left) and after (right) correction for different microphones ('x' and '+').

Source: Maier et al. [8]

## Motivation - Confounds

- If confounder is **known**, we may be able to correct for it:
- Maier et al. [8]: QMOS – Use knowledge of same participants



Speaker visualization before (left) and after (right) correction for different microphones ('x' and '+').

- Other confounds: **Sensor**, lighting, age/sex of participants, temperature, ...
- Best strategy: Be aware and avoid!

Source: Maier et al. [8]

## Motivation - Unintuitive Behaviour of Neural Networks



This is a panda, 57.7% certainty



This is a gibbon, 99.3% certainty

- Output of the same network
- No “visible” difference
- **Adversarial Example:** differ only by **specifically optimized**, added “noise”

Source: <https://blog.openai.com/adversarial-example-research/>

## Motivation - Adversarial Examples and Optical Illusions

Human perception is not flawless:



Waterfall, 1961, M.C. Escher

## Motivation - Adversarial Examples and Optical Illusions

Human perception is not flawless:



Waterfall, 1961, M.C. Escher



Pareidolia in Neptune's Grotto, Italy

## Motivation - Adversarial Examples

- Can be generated to cause a specific mistake
- Example #1: Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition (Sharif et al. [10])

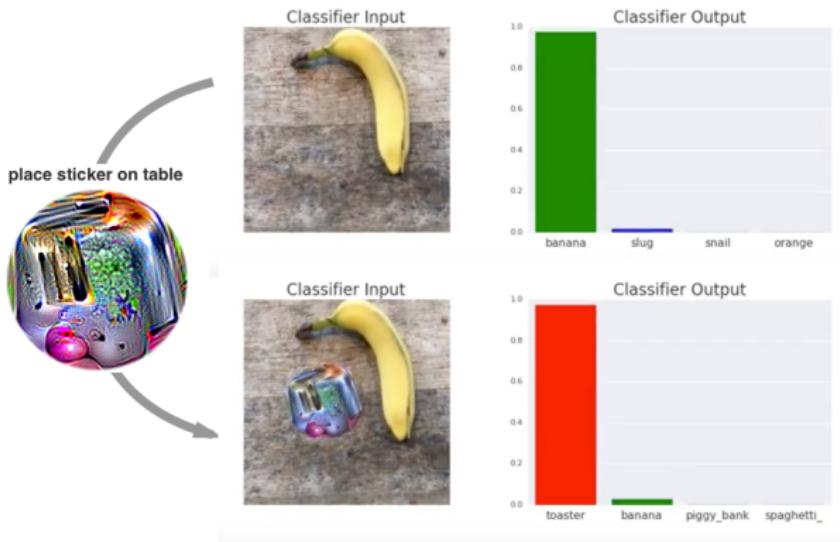


Reese Witherspoon impersonating Russel Crowe with a set of glasses

Source: Sharif et al. [10]

## Motivation - Adversarial Examples

- Example #2: Adversarial Stickers [1]



Small printed sticker is added to a scene. Link to video

Source: Brown et al. [1]

## Motivation - Summary

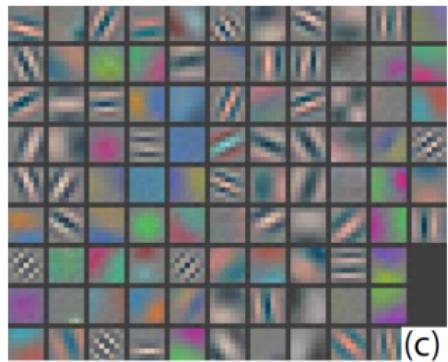
Main goal: Understand how the network represents data

- Identify confounds
- Explain why a network works (or why not)
- Increase confidence in predictions
- Understand (or at least investigate) limitations (→ adversarial examples)

# Simple Parameter Visualization

## Direct Visualization of Learned Kernels

- Idea: Plot learned filter weights directly
- Easy to implement, easy to interpret for the first layer(s)
  - Mostly edge and Gabor filters
  - Example MNIST: stroke detection
  - Very noisy first layer filters: something wrong with the setup

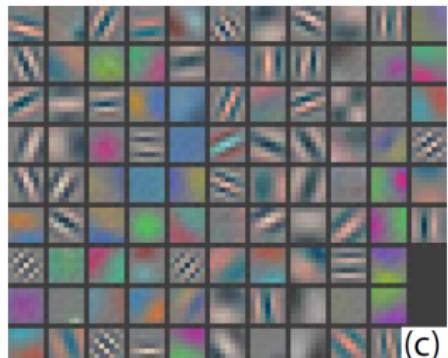


First layer filters in AlexNet ( $11 \times 11$  conv)

Source: Image from [15]

## Direct Visualization of Learned Kernels

- Idea: Plot learned filter weights directly
- Easy to implement, easy to interpret for the first layer(s)
  - Mostly edge and Gabor filters
  - Example MNIST: stroke detection
  - Very noisy first layer filters: something wrong with the setup
- Apart from that, mostly uninteresting
- No easy interpretation for higher layers, esp. with small kernel sizes, e.g.,  $3 \times 3$



First layer filters in AlexNet ( $11 \times 11$  conv)

Source: Image from [15]

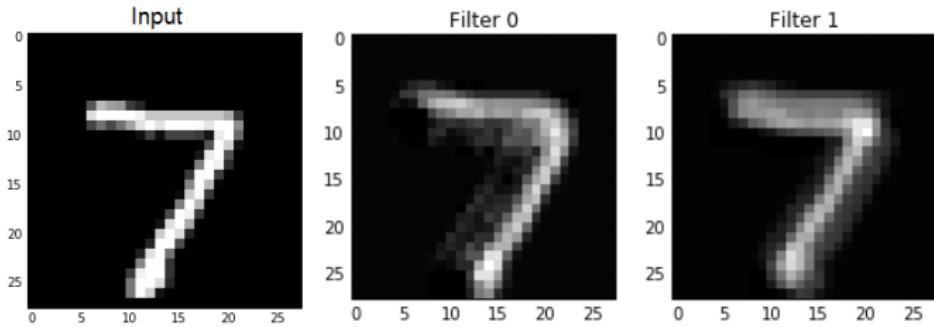
## Visualization of Activations

- Problem: Kernels difficult to interpret  
→ Idea: Instead visualize activations generated by kernels
- Strong response: feature is present, weak response: feature is absent
- Possible for any layer/neuron in the network, with different resolutions

Source: <https://medium.com/@awjuliani/visualizing-neural-network-layer-activation-tensorflow-tutorial-d45f8bf7bbc4>

## Visualization of Activations

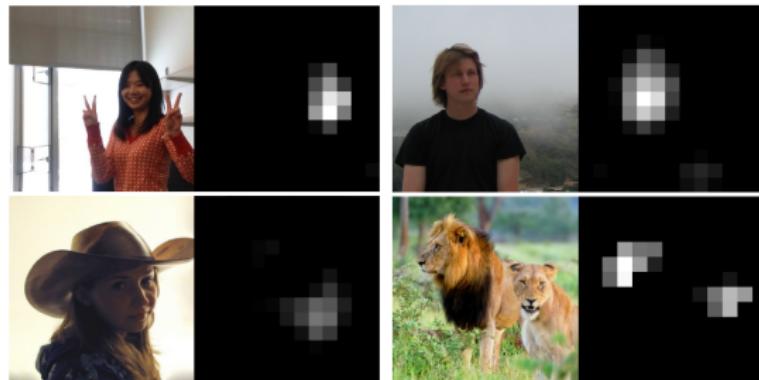
- Problem: Kernels difficult to interpret  
→ Idea: Instead visualize activations generated by kernels
- Strong response: feature is present, weak response: feature is absent
- Possible for any layer/neuron in the network, with different resolutions
- For first-layer neurons, activations look like normal filter responses:



Source: <https://medium.com/@awjuliani/visualizing-neural-network-layer-activation-tensorflow-tutorial-d45f8bf7bbc4>

## Visualization of Activations (cont.)

- For higher-level neurons, the activation maps are usually more coarse (remember: pooling layers)
- Channels may correspond to specific features, e.g., faces:



- Deep Visualization Toolbox [14]: Code online available
- Drawback: No insight into what exactly caused the response, coarse representation

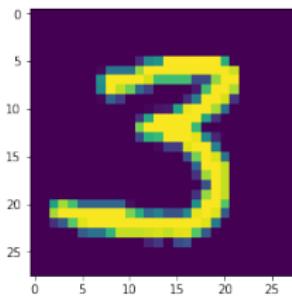
Source: Yosinski et al. [14]

## Investigating Features via Occlusion

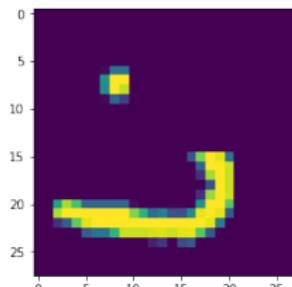
- Idea: Move a masking patch around the input image
- If occlusions cause a significant drop in prediction confidence → area important for classification (Zeiler et al. [15]).

## Investigating Features via Occlusion

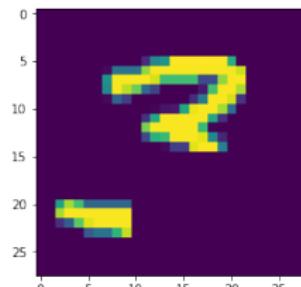
- Idea: Move a masking patch around the input image
- If occlusions cause a significant drop in prediction confidence → area important for classification (Zeiler et al. [15]).



3 with 97.4% confidence



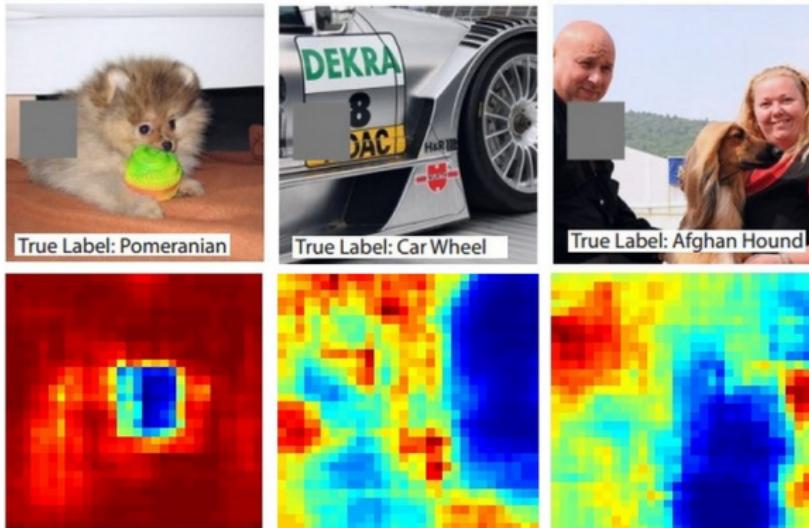
3 with 83% confidence



3 with 94% confidence

→ Can identify confounds, e.g., wrong focus

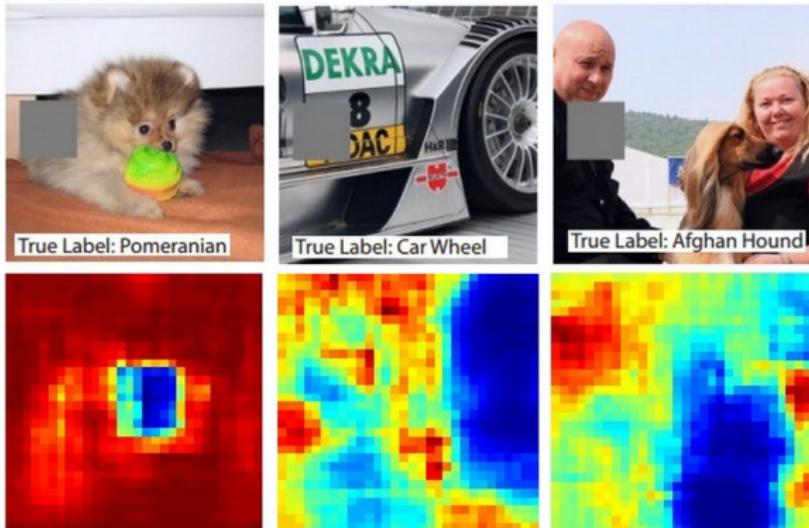
## Investigating Features via Occlusion (cont.)



- Shift mask over input generates **probability heatmap** for a class

Source: Zeiler et al. [15]

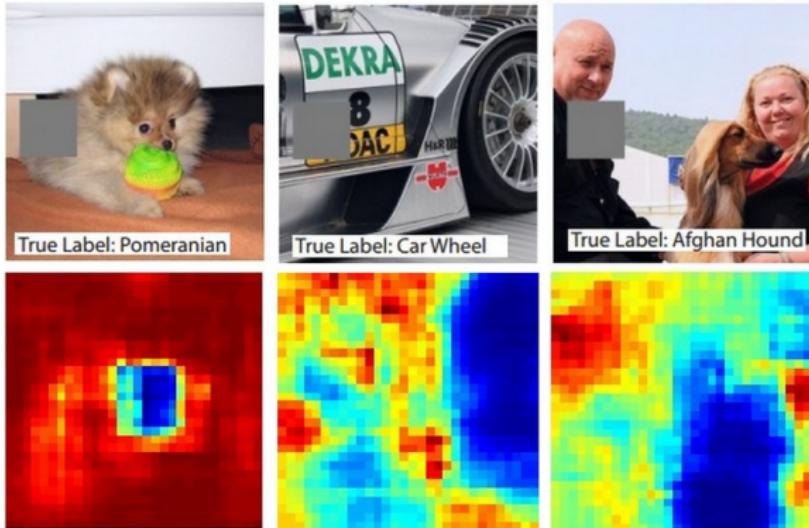
## Investigating Features via Occlusion (cont.)



- Shift mask over input generates **probability heatmap** for a class
- Pomeranian: Only occlusion of dog's face causes a drop in performance

Source: Zeiler et al. [15]

## Investigating Features via Occlusion (cont.)



- Shift mask over input generates **probability heatmap** for a class
- Pomeranian: Only occlusion of dog's face causes a drop in performance
- Car wheel: Occlusion of advertisements - correlated feature learned?

Source: Zeiler et al. [15]

## Investigating Features via Maximally Activating Images

- So far: Looking at activations for single images → cumbersome
- More general question: Which inputs cause high activations?

Source: Girshick et al. [3]

## Investigating Features via Maximally Activating Images

- So far: Looking at activations for single images → cumbersome
- More general question: Which inputs cause high activations?
- Idea: Find input that activates a specific neuron the most → “maximally activating image” [3]

Source: Girshick et al. [3]

## Investigating Features via Maximally Activating Images

- So far: Looking at activations for single images → cumbersome
- More general question: Which inputs cause high activations?
- Idea: Find input that activates a specific neuron the most → “maximally activating image” [3]
- Example: Dog face? Dark spots?



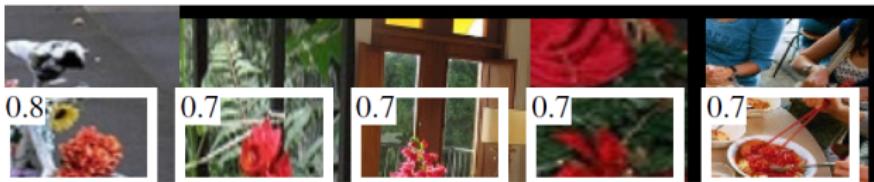
Source: Girshick et al. [3]

## Investigating Features via Maximally Activating Images (cont.)

- Benefits: Easy to implement, “false friends” are comparatively easy to find
- Drawbacks: Neurons don’t necessarily have semantic meaning by themselves, rather basis vectors” of a representation

## Investigating Features via Maximally Activating Images (cont.)

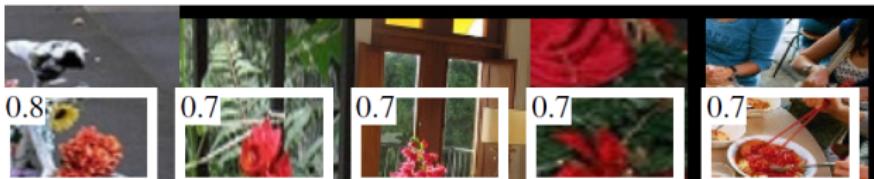
- Benefits: Easy to implement, “false friends” are comparatively easy to find
- Drawbacks: Neurons don’t necessarily have semantic meaning by themselves, rather basis vectors” of a representation



Red flowers (and tomato sauce?)

## Investigating Features via Maximally Activating Images (cont.)

- Benefits: Easy to implement, “false friends” are comparatively easy to find
- Drawbacks: Neurons don’t necessarily have semantic meaning by themselves, rather basis vectors” of a representation



Red flowers (and tomato sauce?)



Specular highlights

Source: Girshick et al. [3]

## t-SNE visualization of CNN codes

- Idea: Understand which images the network regards as “similar”
- Question: How to define and show similarity?

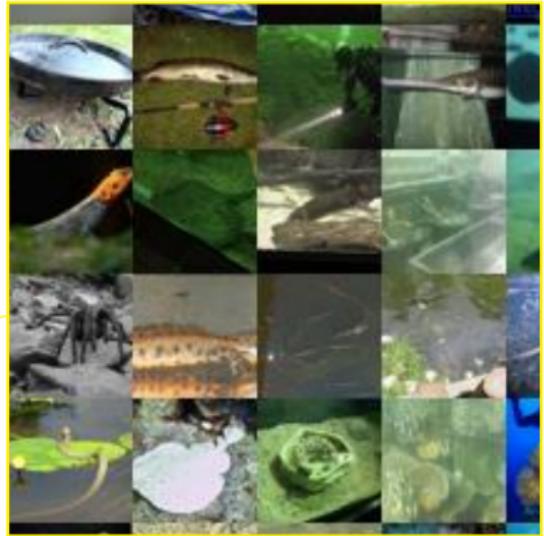
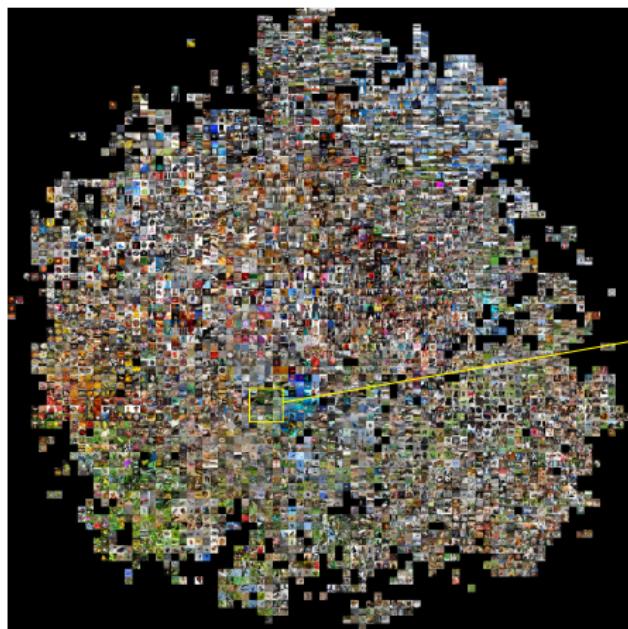
## t-SNE visualization of CNN codes

- Idea: Understand which images the network regards as “similar”
- Question: How to define and show similarity?
- Karpathy [5]: Compute activations of the last layer and group inputs with “similar” activations
- t-SNE: t-Distributed Stochastic Neighbor Embedding
  - Performs dimensionality reduction for high-dimensional datasets
  - Result: 2-D embedding that respects high-dimensional distances of activations
- 2-D map of images

## t-SNE visualization of CNN codes

- Idea: Understand which images the network regards as “similar”
- Question: How to define and show similarity?
- Karpathy [5]: Compute activations of the last layer and group inputs with “similar” activations
- t-SNE: t-Distributed Stochastic Neighbor Embedding
  - Performs dimensionality reduction for high-dimensional datasets
  - Result: 2-D embedding that respects high-dimensional distances of activations
  - 2-D map of images
- Can help to assess whether the network grasps the correct concept of “similarity”
- Drawback: 2-D embedding of **very** high dimensional space, difficult to interpret

## t-SNE visualization of CNN codes (cont.)



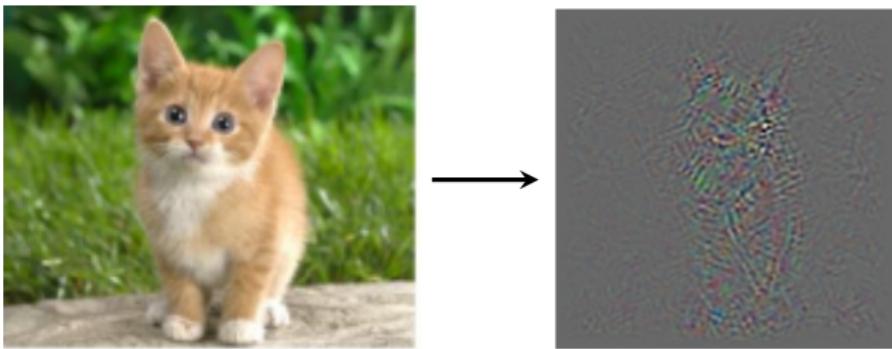
Macroscopic patterns (colored regions/backgrounds - confounds?), local similarity

Source: <http://cs.stanford.edu/people/karpathy/cnnembed/>

# Gradient-Based Visualization

## Backpropagation for Visualization

- Question: Which pixels are most significant to a neuron? Which would have most affected neuron output, had they been different?
- For which pixels  $x_i$  will  $\frac{\partial \text{neuron}}{\partial x_i}$  be large?
- Use backpropagation to compute this gradient for a specific neuron



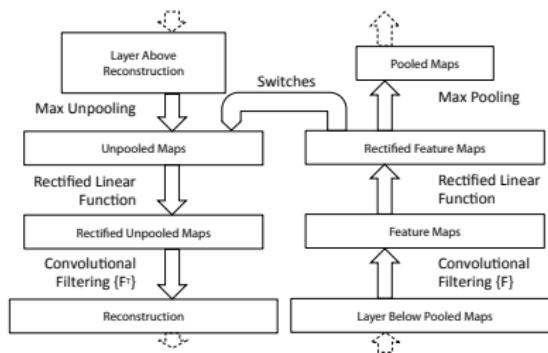
Source: [12]

## Backpropagation for Visualization [11]

- Backpropagation → we need a loss that we can backpropagate
- Pseudo loss:  $f_n(x)$
- $f_n$ : activation of an arbitrary neuron of any layer
- Nearly equivalent alternative (next slide): Use “reverse” network

## Feature Visualization – Deconvnet [15]

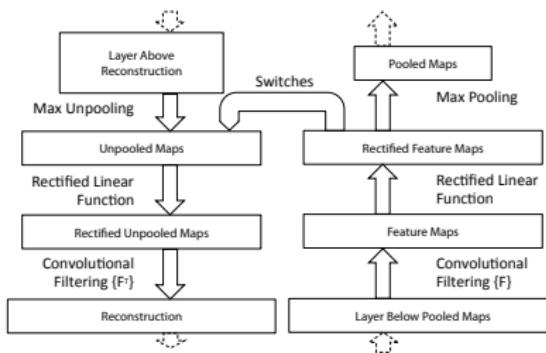
- Input: trained network, image
- Choose one activation and set all others to zero
- Use reverse network (deconvnet)



Source: [15]

## Feature Visualization – Deconvnet [15]

- Input: trained network, image
- Choose one activation and set all others to zero
- Use reverse network (deconvnet)

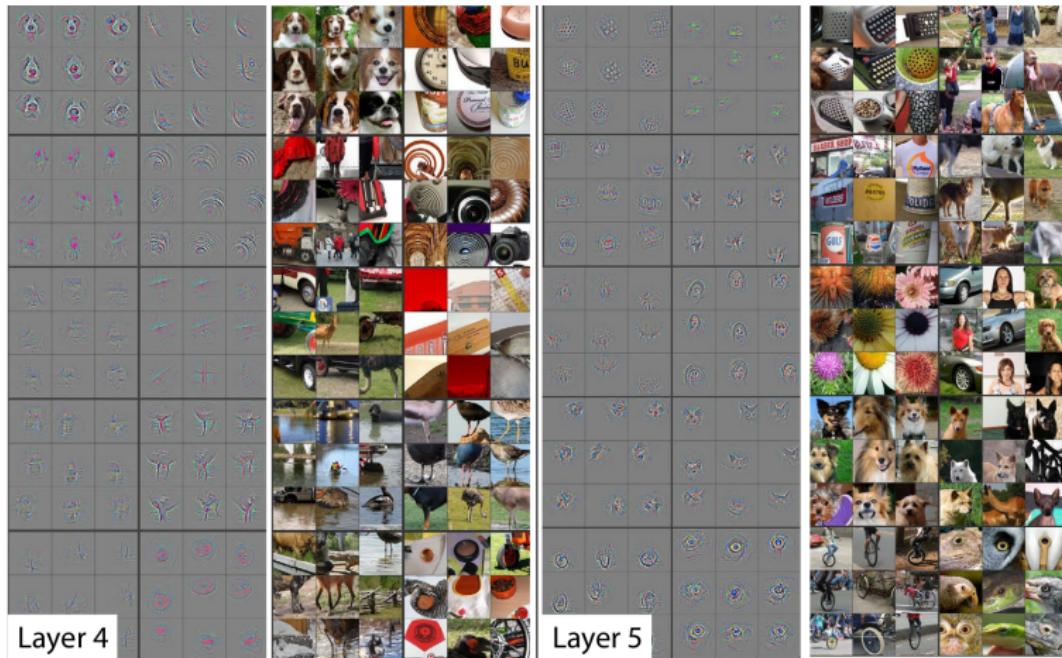


- No training involved (just record pooling locations: “switches”)
- Forward pass of reverse network effectively the same as backward pass of the network (apart from ReLU)!

Source: [15]

# Deconvnet Examples

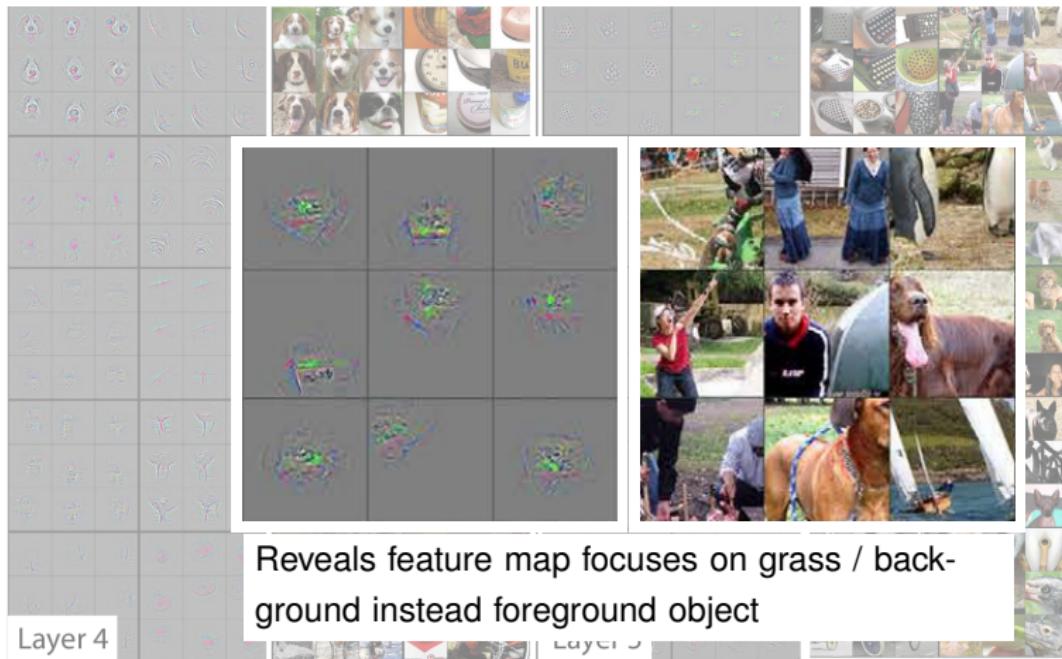
Visualization of the top 9 activations + corresponding patch



Source: [15]

# Deconvnet Examples

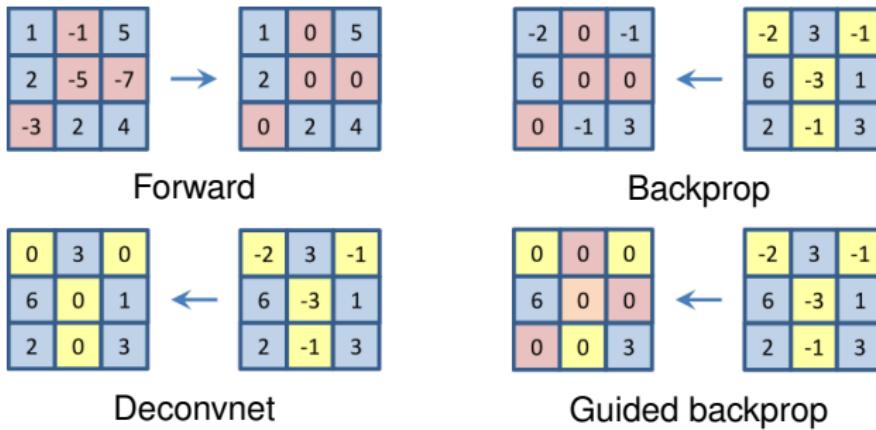
Visualization of the top 9 activations + corresponding patch



Source: [15]

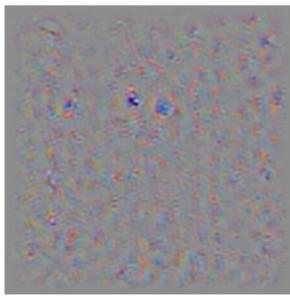
# Guided Backpropagation

- Improve result by "guiding" the backpropagation process
  - Idea behind guided backpropagation:
    - Positive gradients = features the neuron is interested in
    - Negative gradients = features the neuron is not interested in
- Set all negative gradients in the backpropagation process to zero
- Propagating through ReLU:

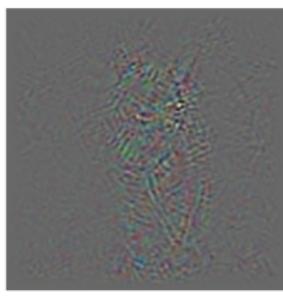


## Guided backpropagation

Often interesting for higher-layer neurons → might reveal neurons that focus on very abstract features



Deconvnet



Backprop

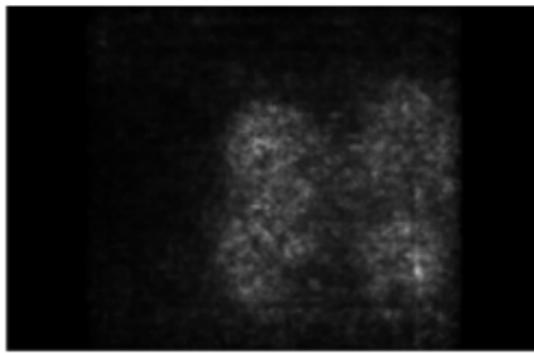


Guided backprop

Source: [12]

## Saliency maps

- Instead of investigating what influences neurons, investigate impact of pixels on **class score**
- Pseudo loss is now unnormalized class score  $f_c$
- Compute gradient w.r.t. image pixels and use the absolute values
- Interesting observation: Saliency map “localizes” dog in the image, even though the network was never trained on localization!



Saliency map w.r.t. class score ‘dog’

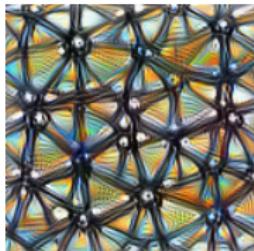
Source: [11]

# Parameter Visualization via Optimization

# Optimization Objectives



Neuron



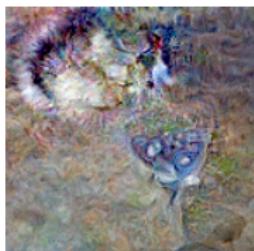
Activation map



Layer



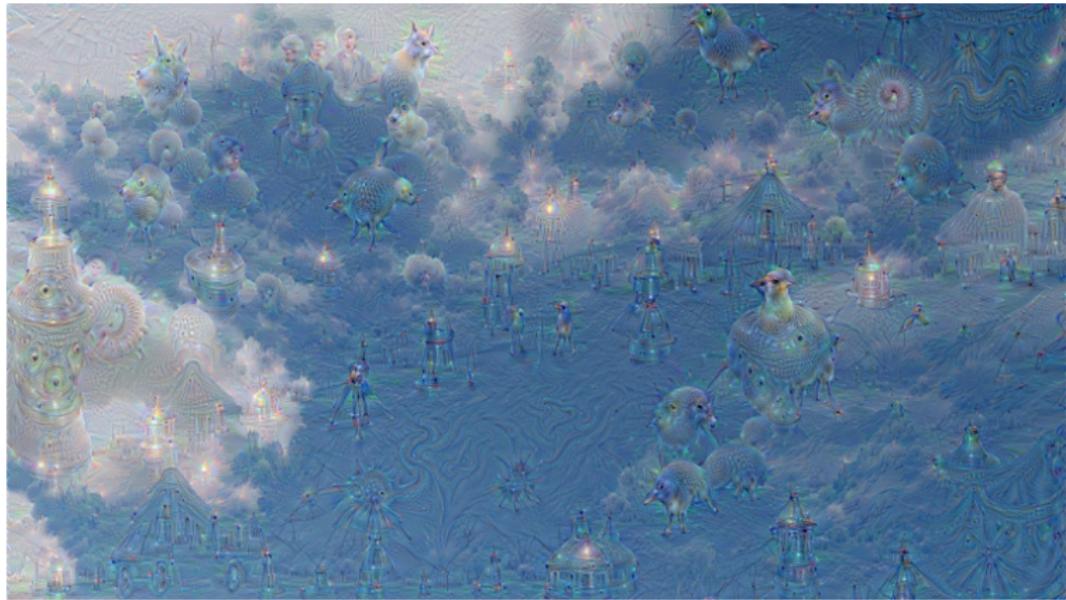
Logits



Class probability  
(softmax)

# Google DeepDream / Inceptionism

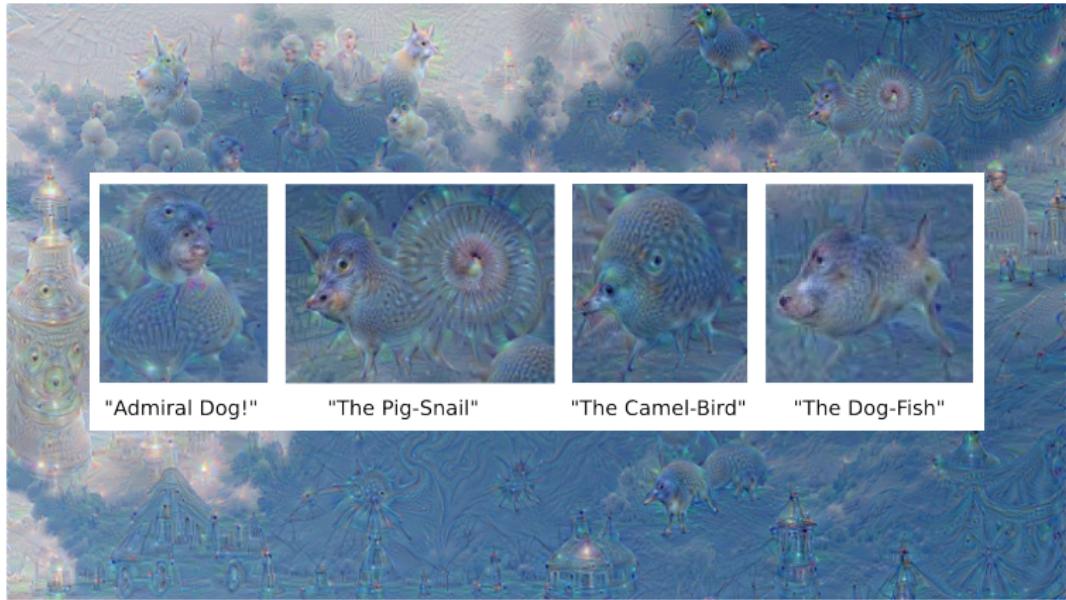
Recall from first lecture:



Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

# Google DeepDream / Inceptionism

Recall from first lecture:



Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

# Google DeepDream / Inceptionism

Recall from first lecture:



Original idea: layer visualization

Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

# Google DeepDream / Inceptionism

Attempt to understand the inner workings of the network: What it "dreams" about when presented with images.

## Idea:

- Arbitrary image or noise as input.
- Instead of adjusting network parameters, tweak image towards high activations of a complete layer.  
→ Search for images, the layer finds interesting
- Different layers enhance different features (low or high level).



Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

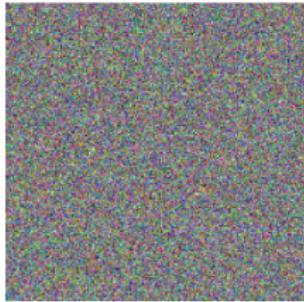
# Google DeepDream / Inceptionism

Find an  $\ell_2$ -regularised image  $\mathbf{x}$ , such that the activation  $f_n(\mathbf{x})$  is high:

$$\max_{\mathbf{x}} f_n(\mathbf{x}) - \lambda \|\mathbf{x}\|_2^2$$

## Algorithm

- For some fixed #iterations (10-20) do
  - Forward propagate until this layer
  - No cost minimization. Instead: maximize the L2 norm of activations of particular NN layer  
(gradients = layer's activations)
  - Backpropagate **all** the way to the input layer
  - Input image gets modified!
- Abstract features emerge
- Use image cascade (from small to large scales)
  - “Inceptionism”



↓ optimize



Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

## Google DeepDream / Inceptionism

- This can reveal hidden weaknesses in the NN classification process



Dumbbell

- Problem: NN learned arm as part of the dumbbell
- Once more: good data is important!

Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

## Inversion

- Inversion attempts to construct an image from a given layer activation  $\hat{\mathbf{y}}$
- This problem corresponds to the following optimization problem [7]:

$$\mathbf{x}^* = \min_{\mathbf{x}} (\|f(\mathbf{x}) - \hat{\mathbf{y}}\|_2^2 + \lambda r(\mathbf{x}))$$

- $\mathbf{x}^*$  is the reconstructed RGB-image
- $f(\mathbf{x})$  is the network output for input image  $\mathbf{x}$
- $\hat{\mathbf{y}}$  is the measured network output
- $r(\cdot)$  is a regularization function (e.g.:  $\ell_2$ , TV)
- Clearly visible features in the reconstructed image correspond to features which matter most to the CNN

## Three families of Regularizers



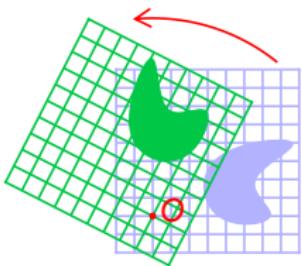
### High frequency penalization

- **High frequency noise** degrades the reconstructions
- **TV** can be used as  $r(\mathbf{x})$  to encourage sparse gradients
- **Low-pass filters** can be applied in every iteration to  $\mathbf{x}$
- **Edge preserving** filters have also been proposed

### Properties

- + Simple
- + **Effective** in producing recognizable features
- **Suppresses legitimate** high frequency features

## Three families of Regularizers



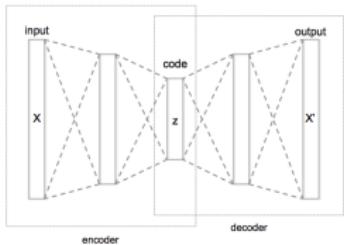
### Transform robustness

- Input should be **invariant** to spatial **transformations**
- Same as **data augmentations**
- Randomly **rotate, scale or jitter x**

### Properties

- + Simple
- + **Effective** in producing recognizable features
- **Orientation is suppressed** even if it was informative

# Three families of Regularizers



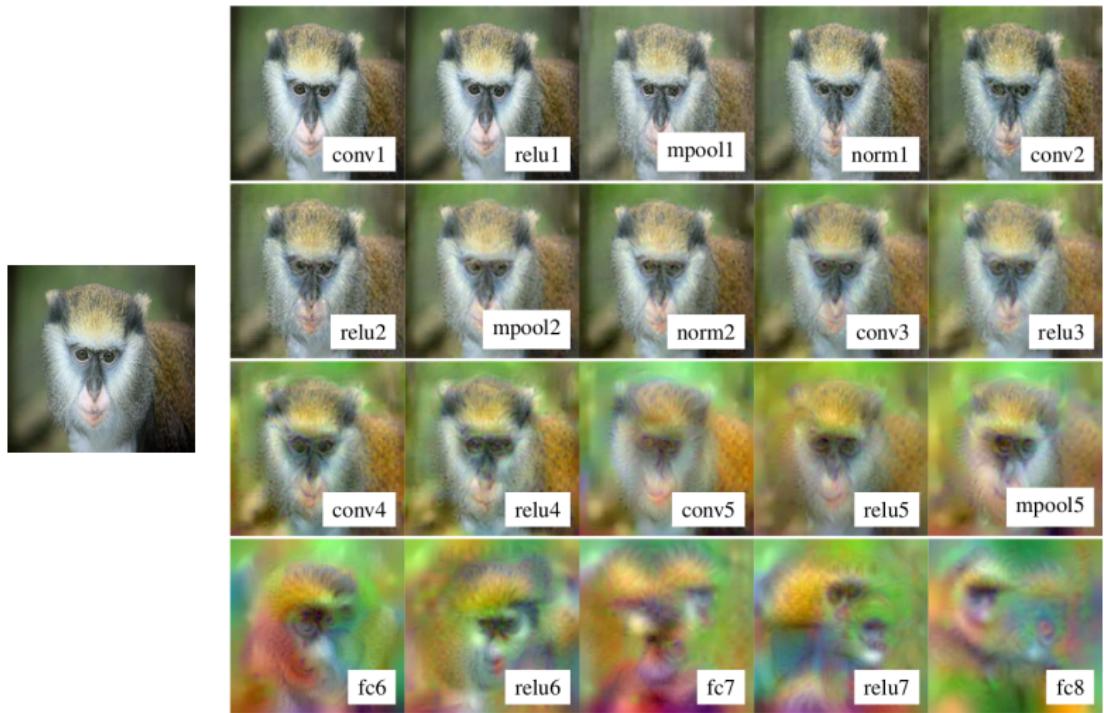
## Learned priors

- Enforce a **prior** on  $x$  to look like a natural image
- Needs a **generative model**
- Use a **GAN** or **VAE**
- Instead of optimizing  $x$  optimize in **latent space**

## Properties

- Needs a **trained** generative model
- + Very nice images
- Ambiguous whether to attribute features in result to the prior or the network

## Inversion – Examples



Source: [7]

**NEXT TIME  
ON DEEP LEARNING**

## Coming Up: Deep Reinforcement Learning

- A training paradigm capable of producing **superhuman** performance
- An algorithm to determine a game **strategy** from playing
- Neural networks moving beyond perception to making **decisions**
- Instructions to finally **beat all your friends** in Atari **games**
- A recipe to **beat every human** in Go

## Comprehensive Questions

- Why is visualization important?
- What can visualization help with? What can't it help with?
- Why are confounds a problem?
- Why could adversarial examples pose a security problem?
- How does occlusion work?
- What is the difference between deconvolution, backpropagation and guided backpropagation regarding feature visualization?
- How is Google DeepDream related to visualization?
- What is a Saliency map?

## Further Reading

- Yosinski et al.: Deep Visualization Toolbox  
<http://yosinski.com/deepvis>
- Olah et al.: Feature Visualization  
<https://distill.pub/2017/feature-visualization/>
- Adam Harley: MNIST Demo  
<http://scs.ryerson.ca/~aharley/vis/conv/>



**FAU**

FRIEDRICH-ALEXANDER-  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
SCHOOL OF ENGINEERING

# References



## References I

- [1] T. B. Brown, D. Mané, A. Roy, et al. "Adversarial Patch". In: [ArXiv e-prints](#) (Dec. 2017). arXiv: 1712.09665 [cs.CV].
- [2] Neil Frazer. [Neural Network Follies](#). 1998. URL:  
<https://neil.fraser.name/writing/tank/> (visited on 01/07/2018).
- [3] Ross B. Girshick, Jeff Donahue, Trevor Darrell, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: [CoRR abs/1311.2524](#) (2013). arXiv: 1311.2524.
- [4] L. N. Kanal and N. C. Randall. "Recognition System Design by Statistical Analysis". In: [Proceedings of the 1964 19th ACM National Conference](#). ACM '64. New York, NY, USA: ACM, 1964, pp. 42.501–42.5020.

## References II

- [5] Andrej Karpathy. t-SNE visualization of CNN codes. URL:  
<http://cs.stanford.edu/people/karpathy/cnnembed/> (visited on 01/07/2018).
- [6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: Advances In Neural Information Processing Systems 25. Curran Associates, Inc., 2012, pp. 1097–1105. arXiv: 1102.0183.
- [7] A. Mahendran and A. Vedaldi. “Understanding deep image representations by inverting them”. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2015, pp. 5188–5196.

## References III

- [8] Andreas Maier, Stefan Wenhardt, Tino Haderlein, et al. "A Microphone-independent Visualization Technique for Speech Disorders". In: Proceedings of the 10th Annual Conference of the International Speech Communication Association. Brighton, England, 2009, pp. 951–954.
- [9] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: Distill (2017). <https://distill.pub/2017/feature-visualization>.
- [10] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, et al. "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition". In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16. Vienna, Austria: ACM, 2016, pp. 1528–1540.

## References IV

- [11] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: [International Conference on Learning Representations \(ICLR\) \(workshop track\)](#) 2014.
- [12] J.T. Springenberg, A. Dosovitskiy, T. Brox, et al. "Striving for Simplicity: The All Convolutional Net". In: [International Conference on Learning Representations \(ICRL\) \(workshop track\)](#) 2015.
- [13] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: [CoRR](#) abs/1711.10925 (2017). arXiv: 1711.10925.
- [14] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, et al. "Understanding Neural Networks Through Deep Visualization". In: [CoRR](#) abs/1506.06579 (2015). arXiv: 1506.06579.

## References V

- [15] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland Cham: Springer International Publishing, 2014, pp. 818–833.