



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Visualization and Attention Mechanisms

**A. Maier, V. Christlein, K. Breininger, Z. Yang, L. Rist, M. Nau, S. Jaganathan, C. Liu, N. Maul, L. Folle,
K. Packhäuser, M. Zinnen**

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

April 24, 2023



Outline

Motivation

Network Architecture Visualization

Visualization of Training

Visualization of Parameters

Simple Parameter Visualization

Gradient-Based Visualization

Parameter Visualization via Optimization

Attention Mechanisms



FAU

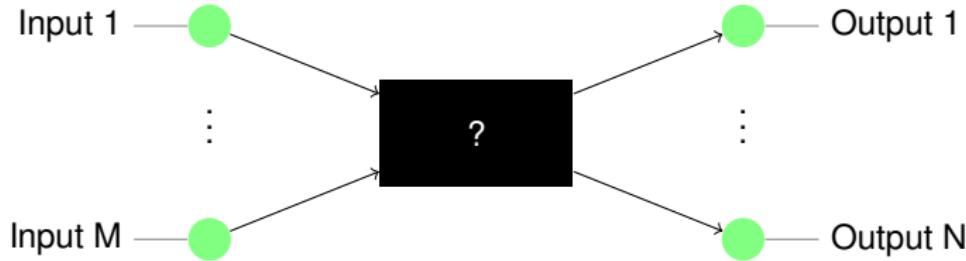
FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

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

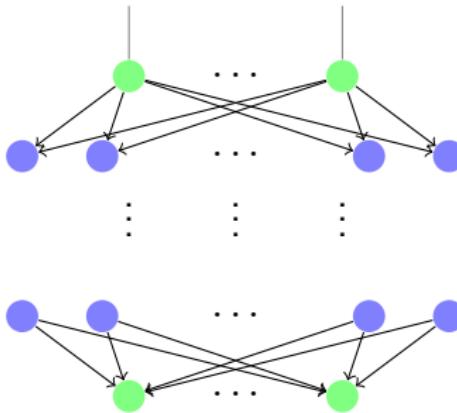
- Important to communicate architectures effectively
- Priors imposed by the architecture may be most important factor for good performance
- Mostly graph-based structures with different granularity
- Compare Lecture 6: Neural Network Architectures

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



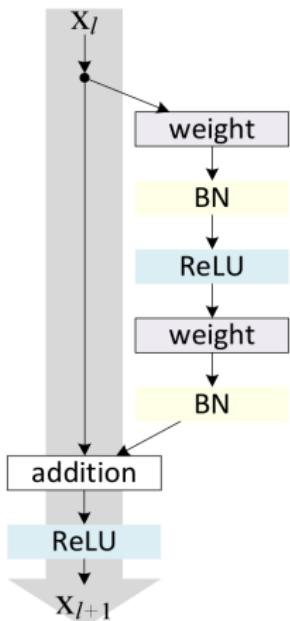
- Detailed representation, focus on connectivity
- Only for small (sub-)networks, building blocks
- Variants, e.g., with explicit weights, recurrent connections, ...

Network Architecture Visualization

Three categories:

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

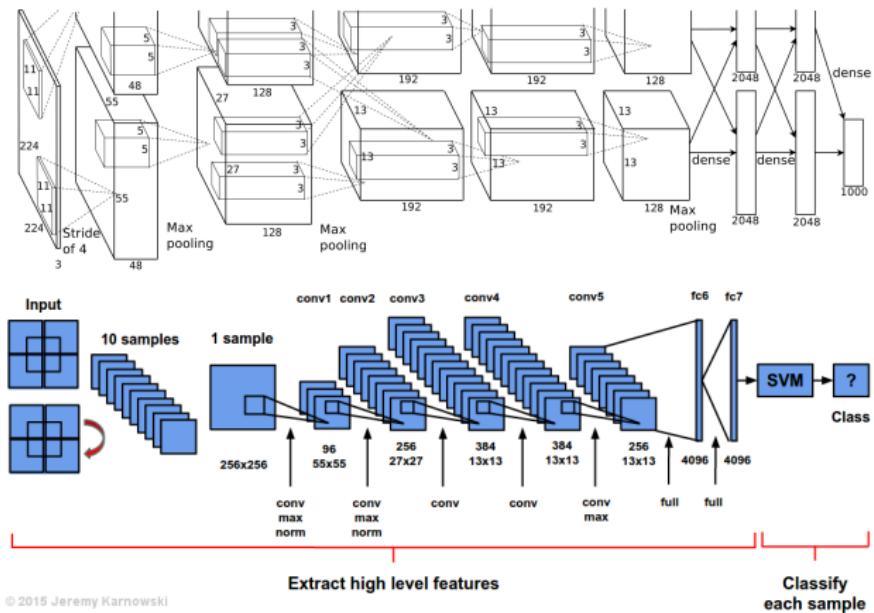
Network Architecture Visualization - Block diagrams



- 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.)

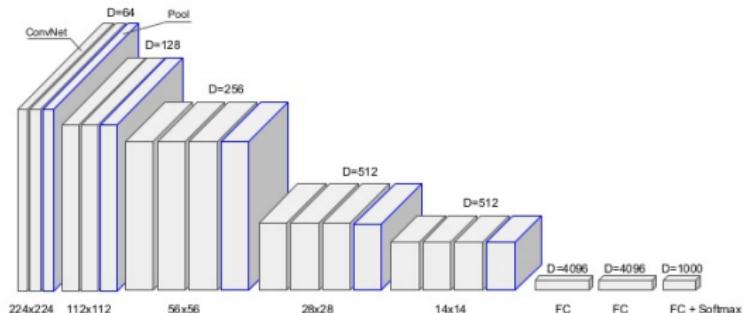


Two different visualizations for AlexNet

Source: <https://jeremykarnowski.wordpress.com/2015/07/15/alexnet-visualization/>, [12]

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

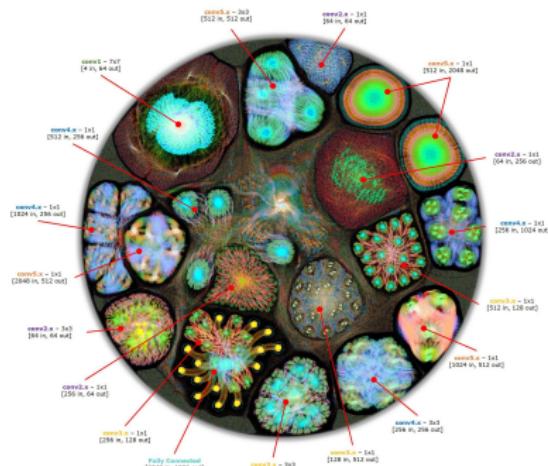
Network Architecture Visualization

Visualization strategies break down into three categories:

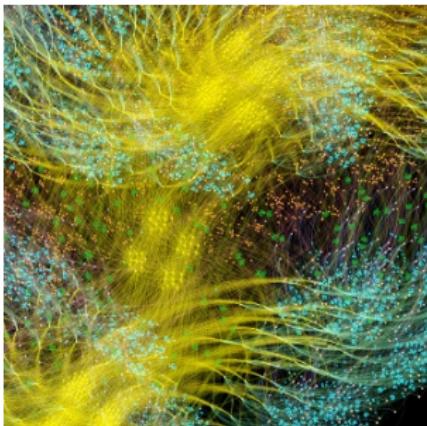
- Node-link diagrams: Neurons = nodes, weighted connections = edges
- Block diagrams: Layer = solid block, single connection between layers
- “Others”

Network Architecture Visualization - Others

- Deep Visualization [25]: 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

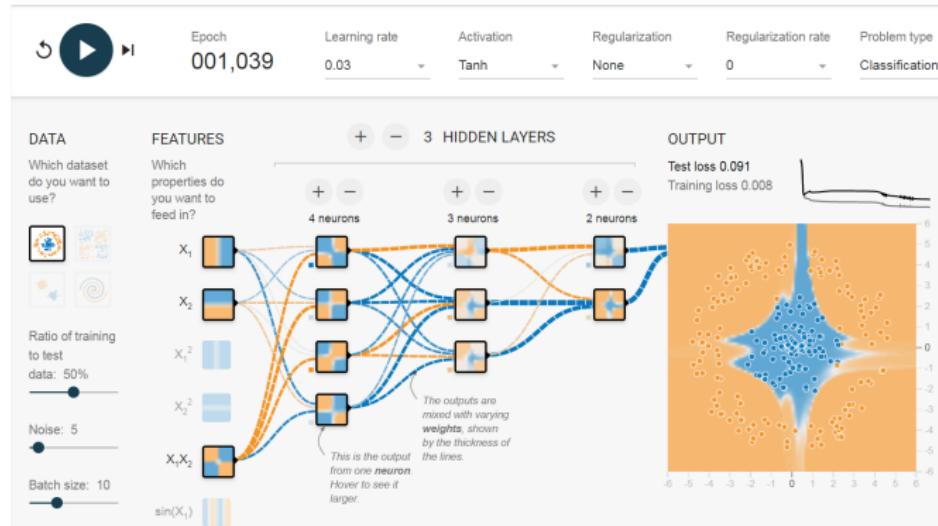
Lots of interesting information available during training:

- 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

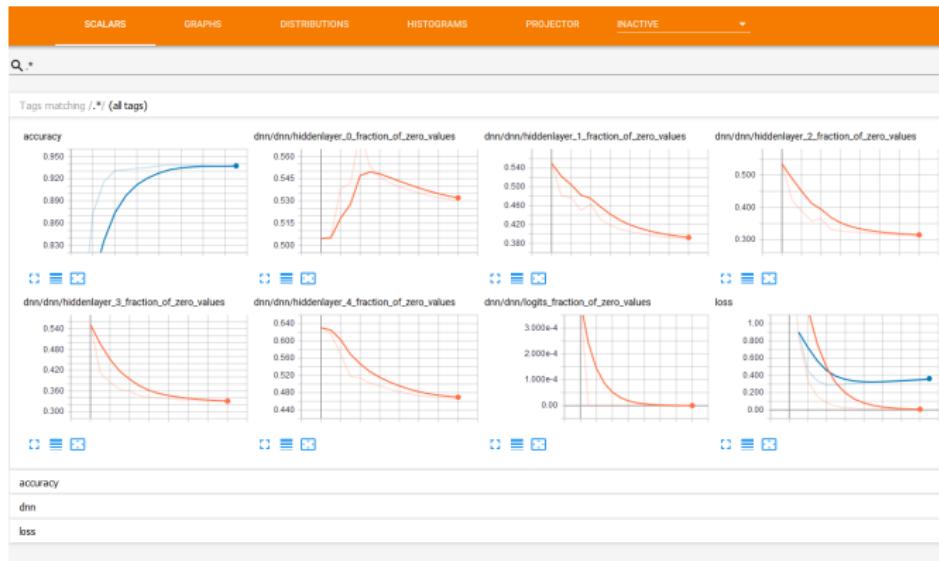


Tensorflow playground

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!**

NEXT TIME
ON DEEP LEARNING



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Visualization and Attention Mechanisms - Part 2

A. Maier, V. Christlein, K. Breininger, Z. Yang, L. Rist, M. Nau, S. Jaganathan, C. Liu, N. Maul, L. Folle,
K. Packhäuser, M. Zinnen

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

April 24, 2023





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 [5], [10]

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: [5]

Motivation - Confounds

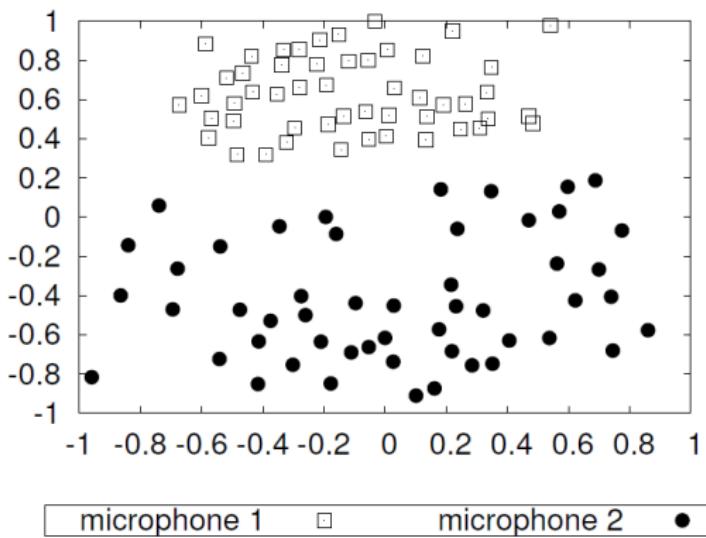


- Task: Identify whether image shows a tank
- Problem: All tank images recorded on cloudy days, all non-tank images on sunny days
- Network learned the **correlated feature** weather, **not** to identify the tank!
- **Important:** Not a fault in the learning algorithm, but in the data!

Source: [5]

Motivation - Confounds (cont.)

Example: Speech recordings with two microphones [15]



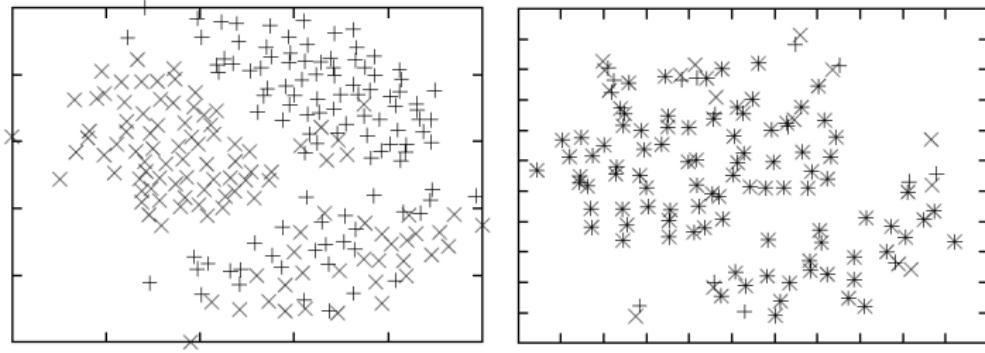
Recordings of the same speaker group with two different microphones

→ ML will focus on the most discriminative features!

Source: Maier et al. [15]

Motivation - Confounds

- If confounder is **known**, we may be able to correct for it:
- Maier et al. [15]: QMOS – Use knowledge of same participants

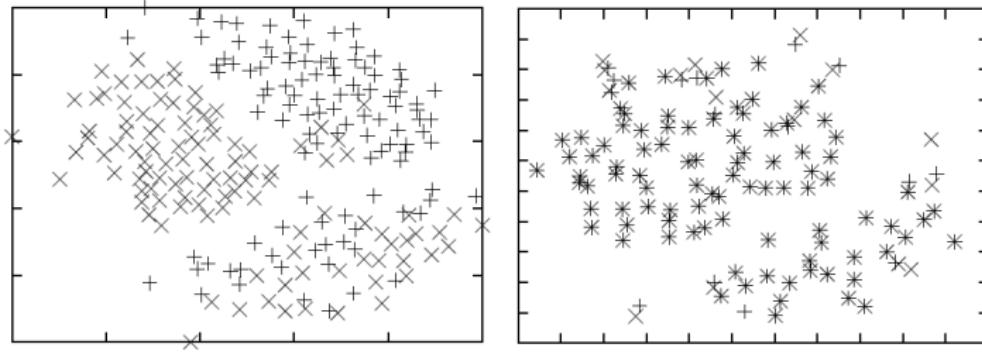


Speaker visualization before (left) and after (right) correction for different microphones ('x' and '+').

Source: Maier et al. [15]

Motivation - Confounds

- If confounder is **known**, we may be able to correct for it:
- Maier et al. [15]: 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. [15]

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. [19])

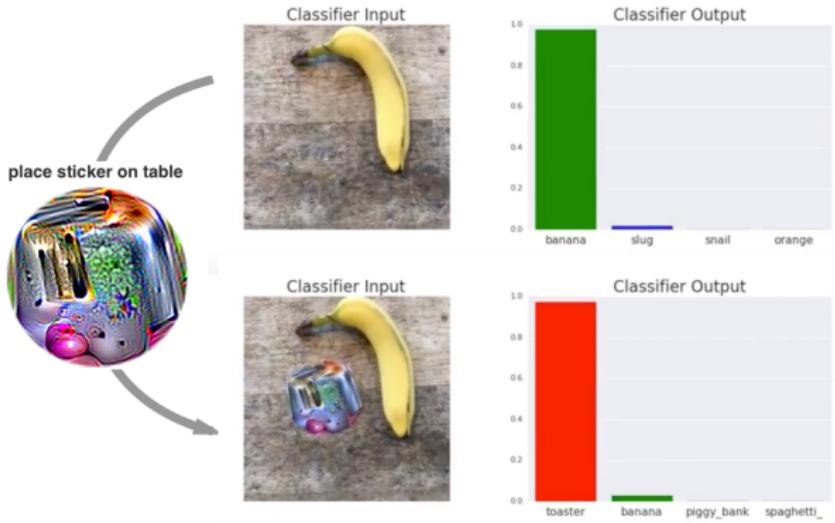


Reese Witherspoon impersonating Russel Crowe with a set of glasses

Source: Sharif et al. [19]

Motivation - Adversarial Examples

- Example #2: Adversarial Stickers [2]



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

Source: Brown et al. [2]

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)

**NEXT TIME
ON DEEP LEARNING**

Visualization and Attention Mechanisms - Part 3

A. Maier, V. Christlein, K. Breininger, Z. Yang, L. Rist, M. Nau, S. Jaganathan, C. Liu, N. Maul, L. Folle,
K. Packhäuser, M. Zinnen

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

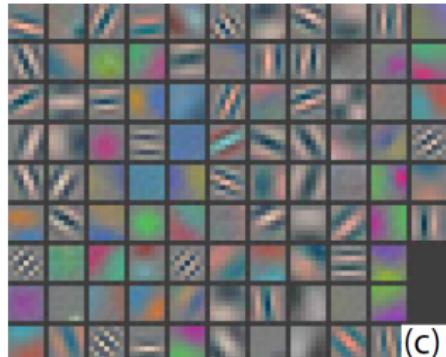
April 24, 2023



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

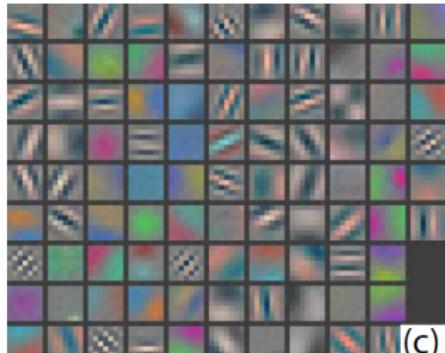


First layer filters in AlexNet (11×11 conv)

Source: Image from [26]

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., 3×3



First layer filters in AlexNet (11×11 conv)

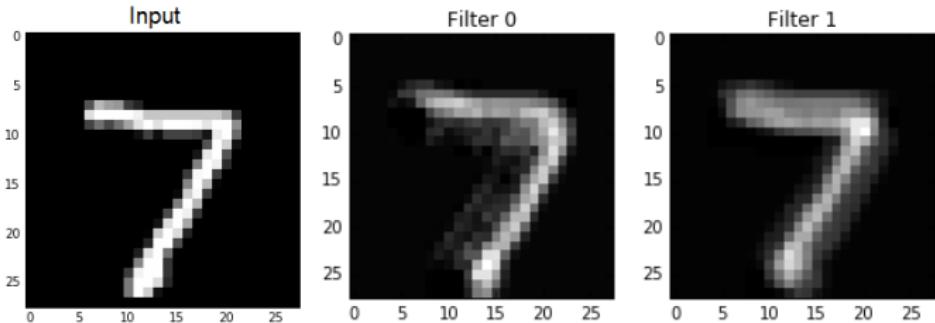
Visualization of Activations

- 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

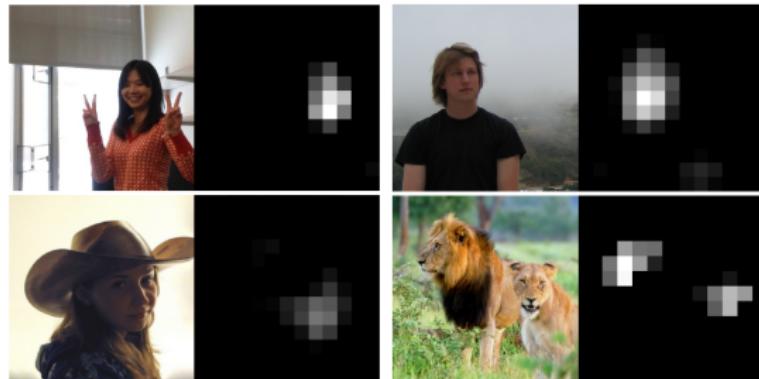
- Problem: Kernels difficult to interpret
→ Idea: Instead visualize activations generated by kernels
- Strong response: feature is present, weak response: feature is absent
- Possible for any layer/neuron in the network, with different resolutions
- For first-layer neurons, activations look like normal 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 [25]: Code online available
- Drawback: No insight into what exactly caused the response, coarse representation

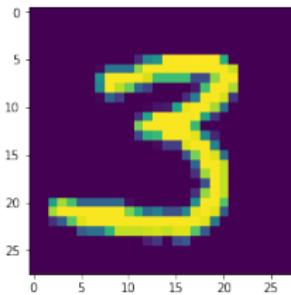
Source: Yosinski et al. [25]

Investigating Features via Occlusion

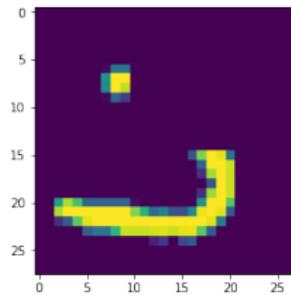
- Idea: Move a masking patch around the input image
- If occlusions cause a significant drop in prediction confidence → area important for classification (Zeiler et al. [26]).

Investigating Features via Occlusion

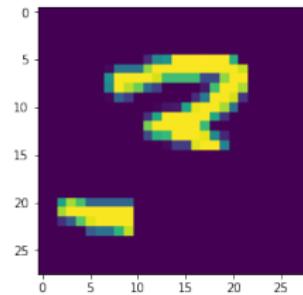
- Idea: Move a masking patch around the input image
- If occlusions cause a significant drop in prediction confidence → area important for classification (Zeiler et al. [26]).



3 with 97.4% confidence



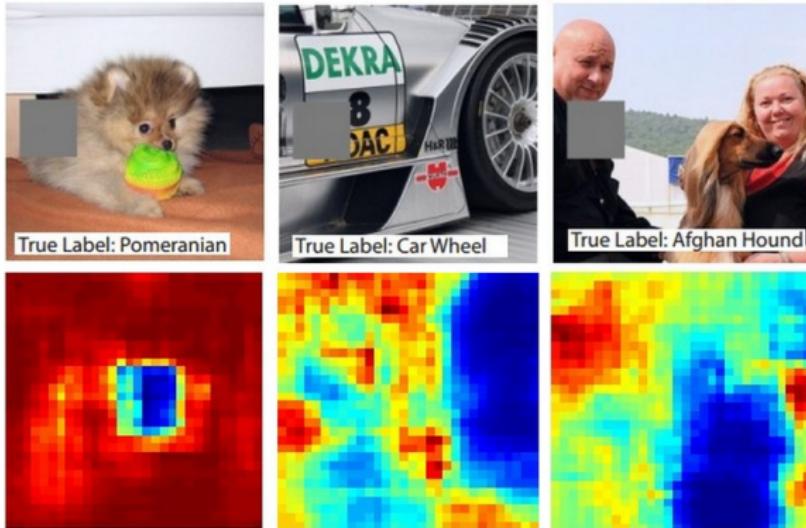
3 with 83% confidence



3 with 94% confidence

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

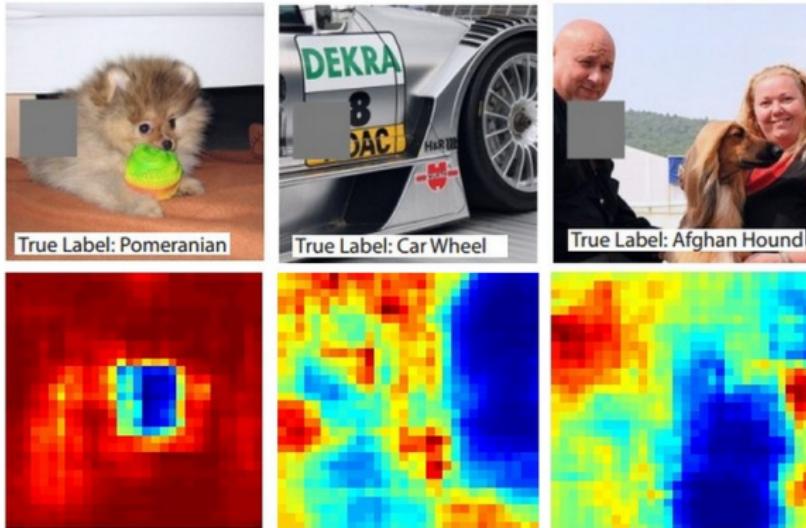
Investigating Features via Occlusion (cont.)



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

Source: Zeiler et al. [26]

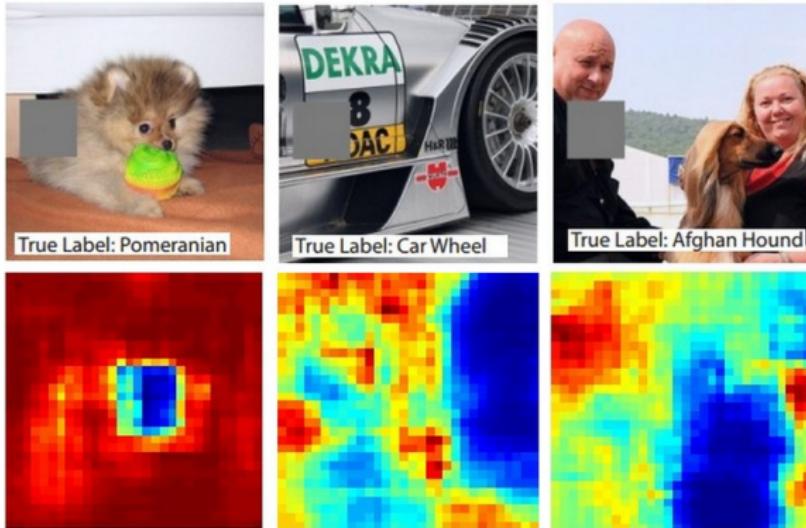
Investigating Features via Occlusion (cont.)



- 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. [26]

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. [26]

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. [6]

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” [6]

Source: Girshick et al. [6]

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” [6]
- Example: Dog face? Dark spots?



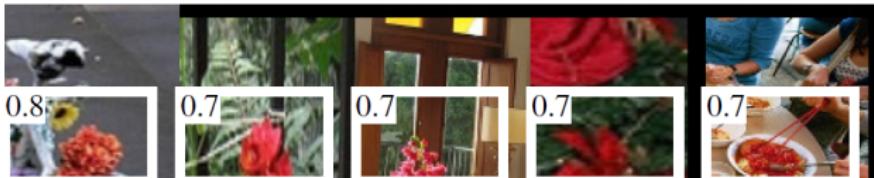
Source: Girshick et al. [6]

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.)

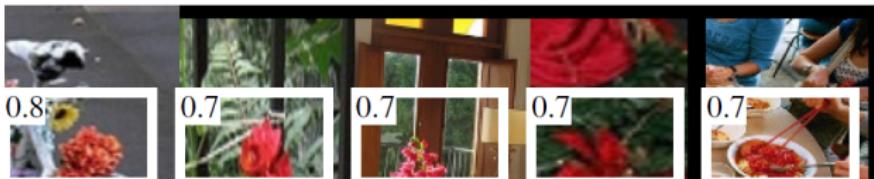
- Benefits: Easy to implement, “false friends” are comparatively easy to find
- Drawbacks: Neurons don’t necessarily have semantic meaning by themselves, rather “basis vectors” of a representation



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. [6]

t-SNE visualization of CNN codes

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

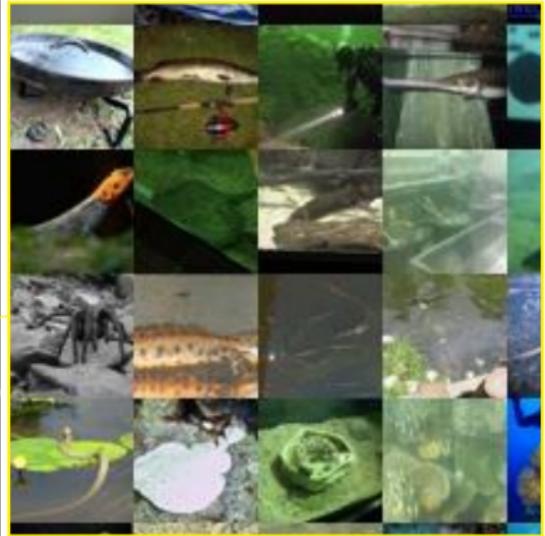
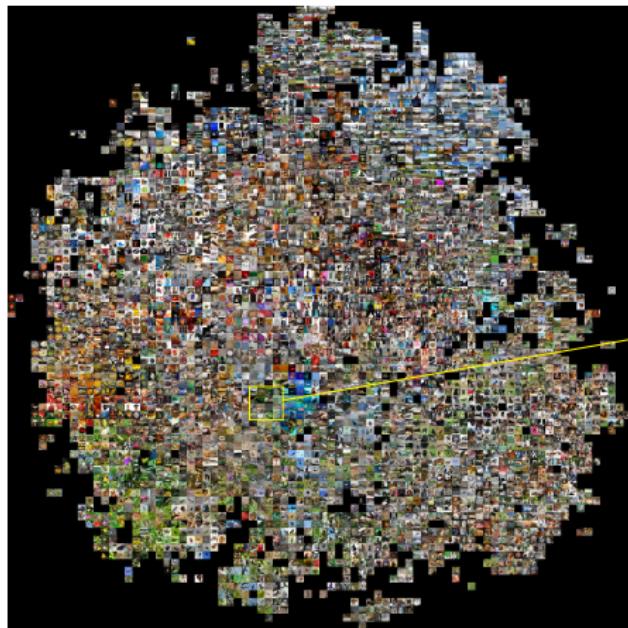
t-SNE visualization of CNN codes

- Idea: Understand which images the network regards as “similar”
- Question: How to define and show similarity?
- Karpathy [11]: 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 [11]: 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/>

**NEXT TIME
ON DEEP LEARNING**

Visualization and Attention Mechanisms - Part 4

A. Maier, V. Christlein, K. Breininger, Z. Yang, L. Rist, M. Nau, S. Jaganathan, C. Liu, N. Maul, L. Folle,
K. Packhäuser, M. Zinnen

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

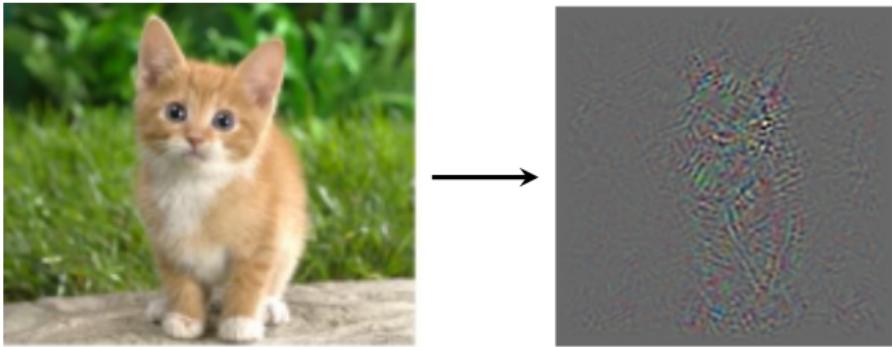
April 24, 2023



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



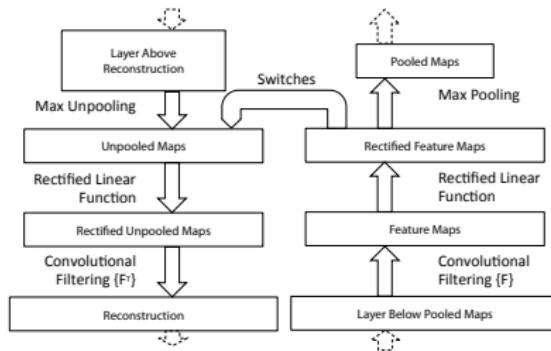
Source: [21]

Backpropagation for Visualization [20]

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

Feature Visualization – Deconvnet [26]

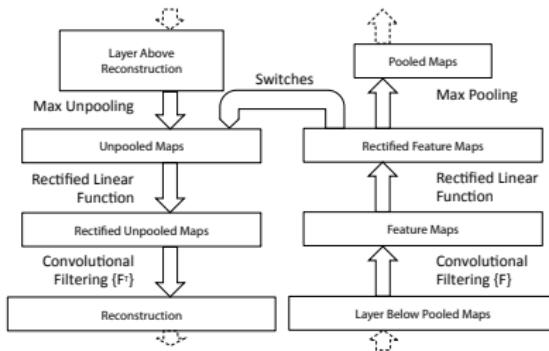
- Input: trained network, image
- Choose one activation and set all others to zero
- Use reverse network (deconvnet)



Source: [26]

Feature Visualization – Deconvnet [26]

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

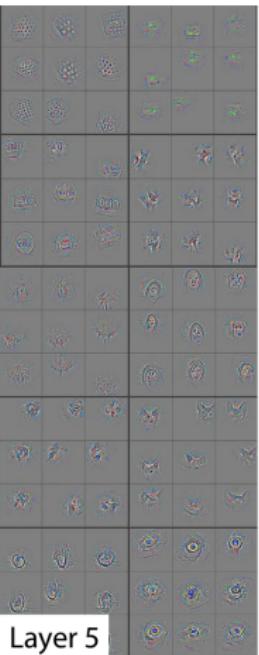
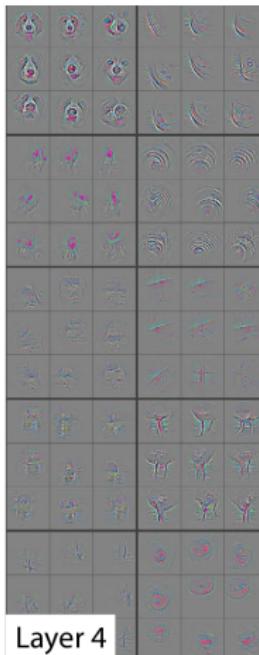


- No training involved (just record pooling locations: “switches”)
- Forward pass of reverse network effectively the same as backward pass of the network (apart from ReLU)!

Source: [26]

Deconvnet Examples

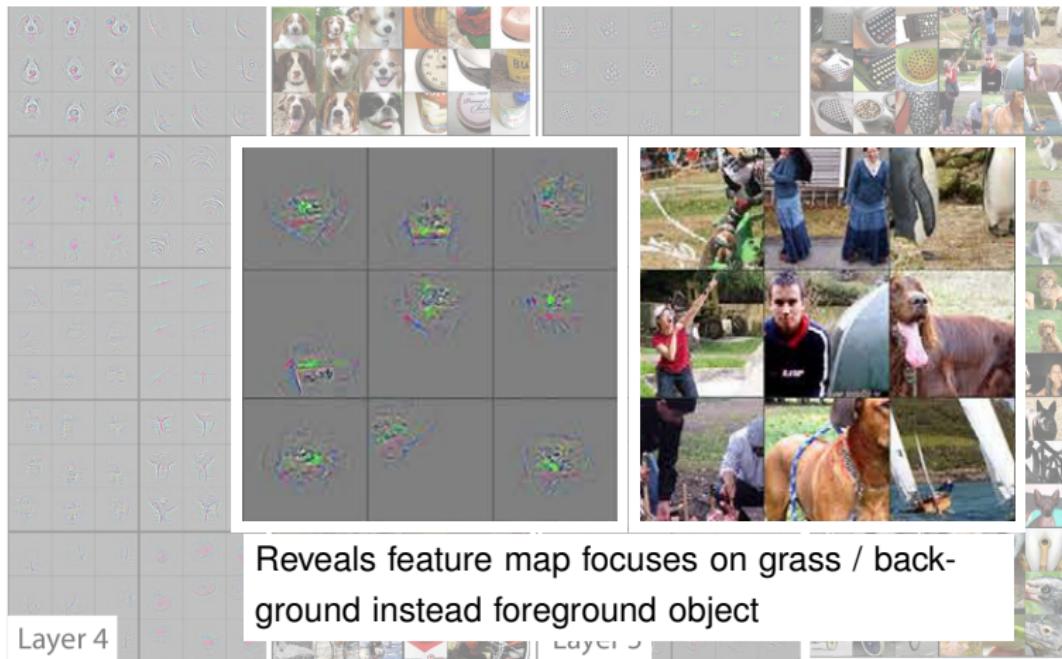
Visualization of the top 9 activations + corresponding patch



Source: [26]

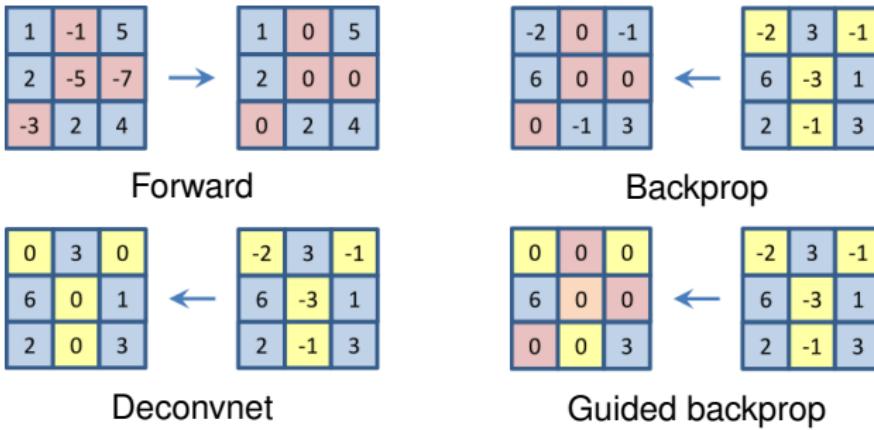
Deconvnet Examples

Visualization of the top 9 activations + corresponding patch



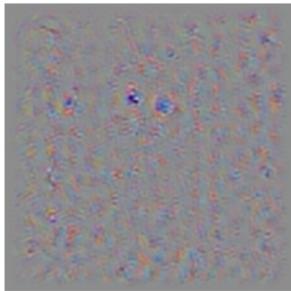
Guided Backpropagation

- 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:

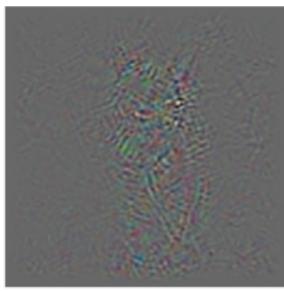


Guided backpropagation

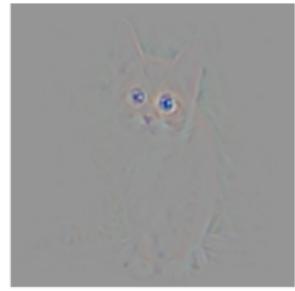
Often interesting for higher-layer neurons → might reveal neurons that focus on very abstract features



Deconvnet



Backprop

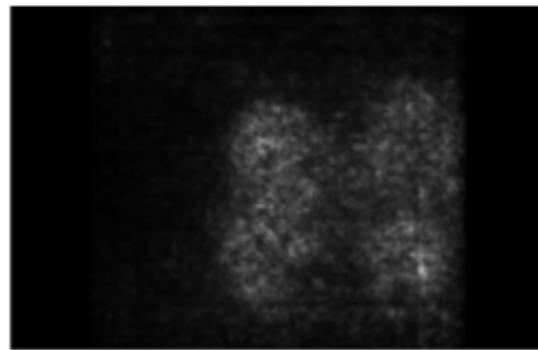


Guided backprop

Source: [21]

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’

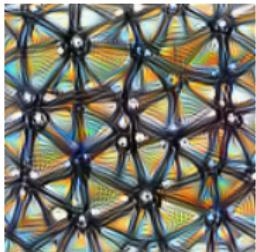
Source: [20]

Parameter Visualization via Optimization

Optimization Objectives



Neuron



Activation map



Layer



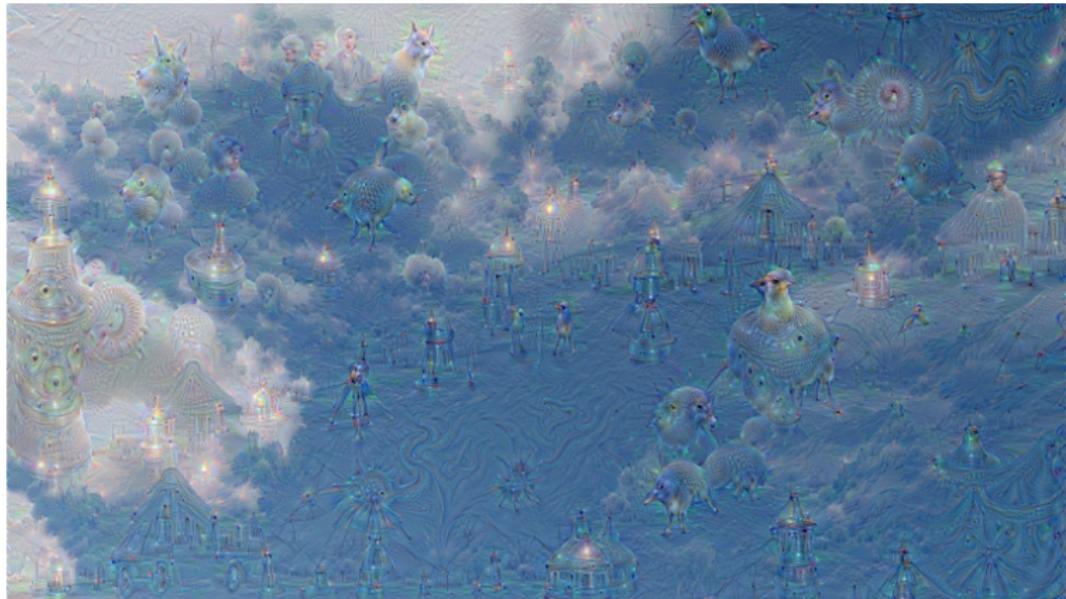
Logits



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

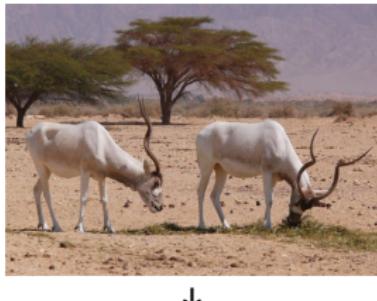
Source: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

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>

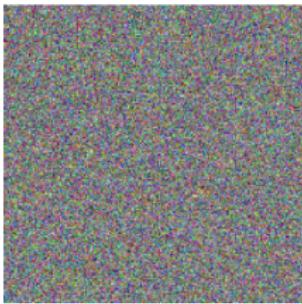
Google DeepDream / Inceptionism

Find an ℓ_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 [14]:

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

- \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.: ℓ_2 , TV)
- Clearly visible features in the reconstructed image correspond to features which matter most to the CNN

Three families of Regularizers



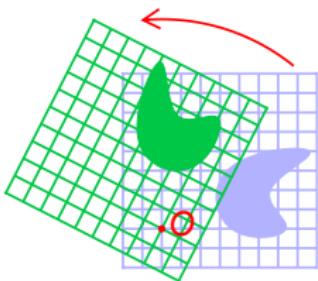
High frequency penalization

- **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



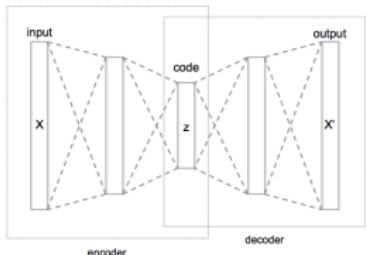
Transform robustness

- 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



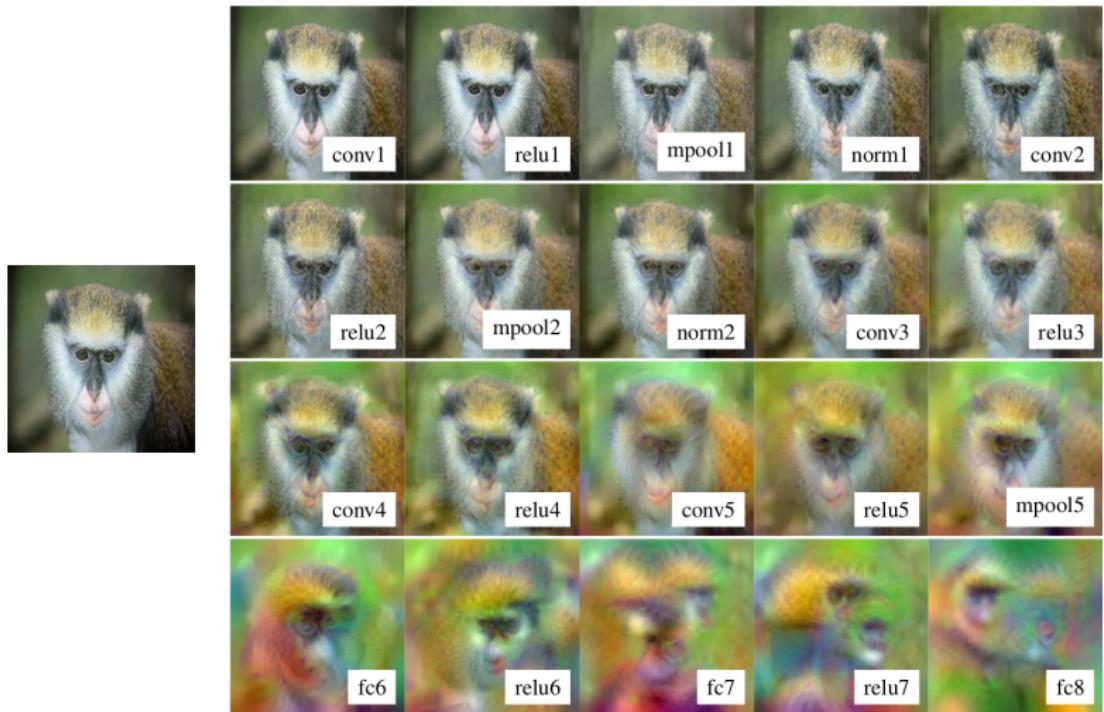
Learned priors

- 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: [14]

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Visualization and Attention Mechanisms - Part 5

A. Maier, V. Christlein, K. Breininger, Z. Yang, L. Rist, M. Nau, S. Jaganathan, C. Liu, N. Maul, L. Folle,
K. Packhäuser, M. Zinnen

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

April 24, 2023





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Attention Mechanisms



What is Attention?

Humans process data by actively shifting their focus:

- Different parts of an image carry different information
- Words derive their specific meaning from **context**
- Remember specific, **related** events in the past
- Allows to follow one thought at a time while suppressing information **irrelevant** to the task
- Example: Cocktail party problem



Source: <https://www.nidcd.nih.gov/newsletter/2012/summer/cocktail-party-problem-how-brain-decides-what-not-hear>

Implicit attention



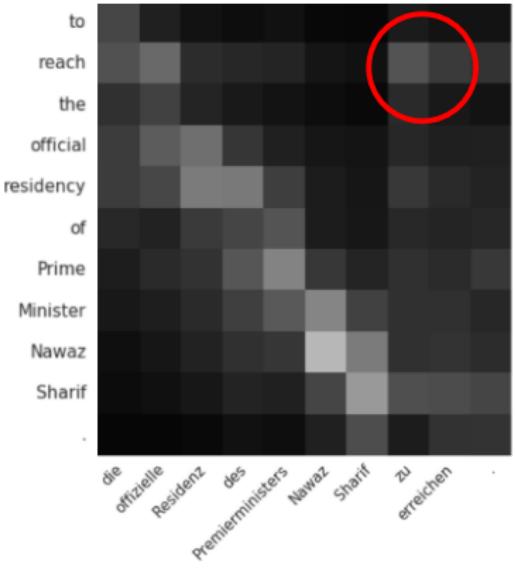
Reminder: Saliency maps (impact of pixels on class score)

Source: Adapted from [9], [20]

Implicit attention



Reminder: Saliency maps (impact of pixels on class score)



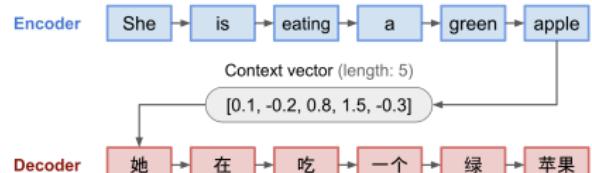
Plotting gradients from CNN in English to German translation

- Networks learn attention implicitly
- Can **explicit attention mechanisms** boost performance?

Source: Adapted from [9], [20]

Sequence to sequence models

- Encoder/decoder architecture, typically RNNs

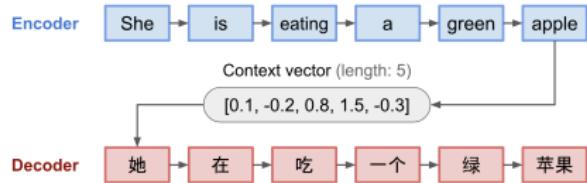


Encoder-decoder for translation

Source: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#neural-turing-machines>

Sequence to sequence models

- Encoder/decoder architecture, typically RNNs
- Encoder network
 - Receives input sequence $\{x_1, \dots, x_T\}$
 - Computes hidden states $\{h_1, \dots, h_T\}$

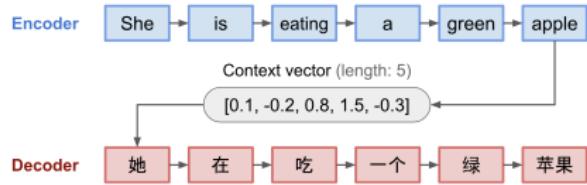


Encoder-decoder for translation

Source: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#neural-turing-machines>

Sequence to sequence models

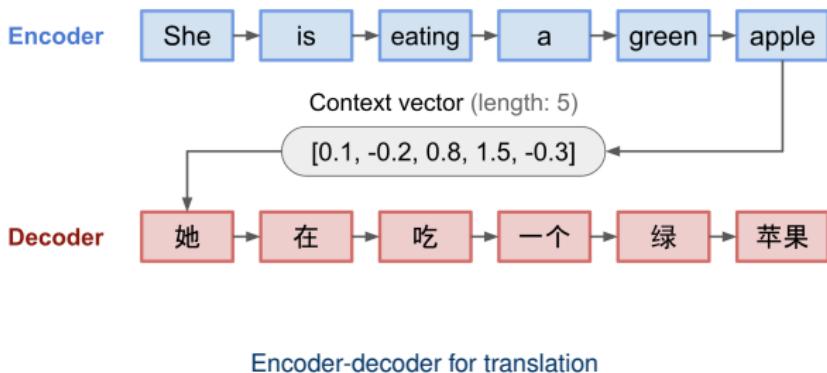
- Encoder/decoder architecture, typically RNNs
- Encoder network
 - Receives input sequence $\{x_1, \dots, x_T\}$
 - Computes hidden states $\{h_1, \dots, h_T\}$
- Decoder network:
 - Receives "**context vector**" h_T
 - Computes own hidden states $\{s_1, \dots, s_{T'}\}$
 - Generates output sequence $\{y_1, \dots, y_{T'}\}$
- Split allows different length in input/output



Encoder-decoder for translation

Source: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#neural-turing-machines>

Sequence to sequence models (cont.)

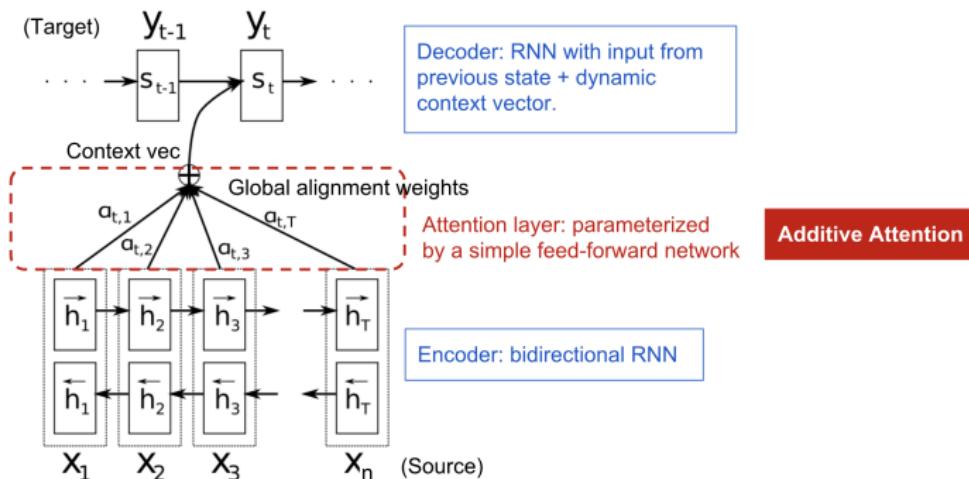


- Different parts of input relate to different parts of output
- Encoding complete content difficult (even for **LSTMs**)

Source: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#neural-turing-machines>

Attention for Seq2Seq: Context vector

- Issue: Context vector \mathbf{h}_T provides no access to earlier inputs
- Provide access with shortcuts: **dynamic** context vector as **combination** of all previous hidden states



Source: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#neural-turing-machines>, adapted from [1]

Attention for Seq2Seq: Context vector

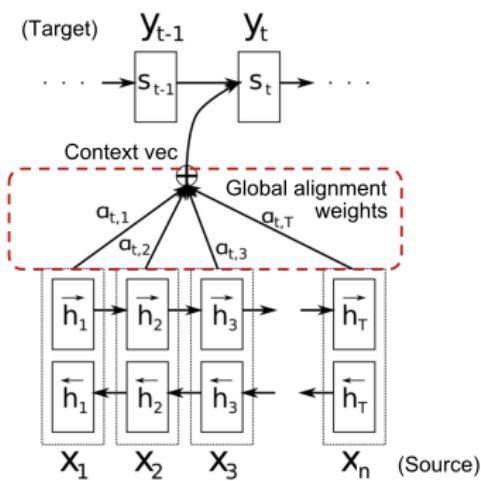
Bahdanau et al. [1]:

"Neural machine translation by jointly learning to align and translate"

- Encoder: **Bidirectional LSTM** with
 $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i]$
- Dynamic context vector
 $\mathbf{c}_t = \sum_i \alpha_{t,i} \mathbf{h}_i$
 with alignment weights

$$\begin{aligned}\alpha_{t,i} &= \text{align}(y_t, x_i) \\ &= \frac{\exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_i \exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i))}\end{aligned}$$

according to a **score** function



Source: Adapted from [1]

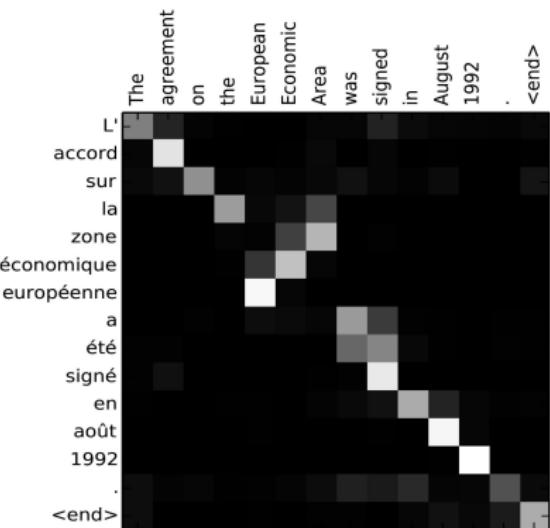
Attention for Seq2Seq: Score Function (cont.)

- Bahdanau et al.: **score** function represented by single layer FCN

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_\alpha^T \mathbf{W}_\alpha [\mathbf{s}_t, \mathbf{h}_t]$$

with **trainable** weights \mathbf{v}_α and \mathbf{W}_α

- Determines **alignment**: Which inputs are important for which outputs
- Alignment scores allow for **interpretation**



Source: Adapted from [1]

Attention for Seq2Seq: Alternative Score Functions

- Content-based (cosine similarity) [7]:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \text{cosine}(\mathbf{s}_t, \mathbf{h}_i)$$

- General [13]:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^T \mathbf{W}_\alpha \mathbf{h}_i$$

- Dot-product [13]:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^T \mathbf{h}_i$$

- Scaled Dot-product [23]:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \frac{\mathbf{s}_t^T \mathbf{h}_i}{\sqrt{n}}$$

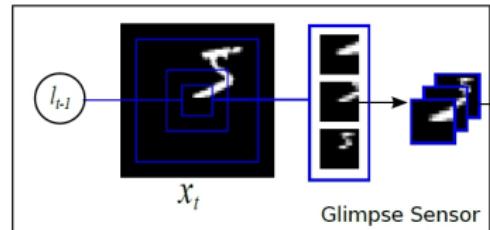
where n is the size of the hidden state

Soft Attention vs. Hard Attention

- So far: Attention is computed over all patches/inputs: **Soft/global attention**
- + Model is fully differentiable
- Not effective/efficient for large inputs

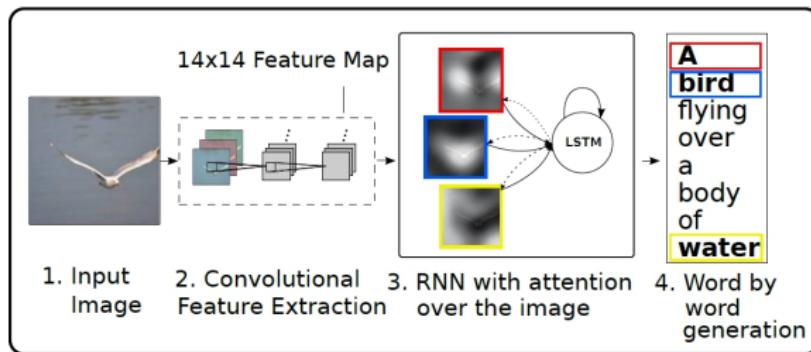
Soft Attention vs. Hard Attention

- So far: Attention is computed over all patches/inputs: **Soft/global attention**
 - + Model is fully differentiable
 - Not effective/efficient for large inputs
- Alternative: **Hard attention**
 Fixed size glimpses on the input, e.g., by sampling from a distribution
 - + Lower computation times
 - Not differentiable, requires e.g.
 reinforcement learning techniques to train
 (\rightarrow Lecture 9)
- "Local attention" [8], [13] as blend: predict center position for focus window/kernel



Number parsing using glimpses [16]

Show-Attend-Tell [24]



Show-Attend-Tell

- Task: Automatic generation of image captions
- Different elements and their relationship in the image trigger different words
- Attention mechanism to improve caption quality

Source: Adapted from [24]

Show-Attend-Tell [24] (cont.)

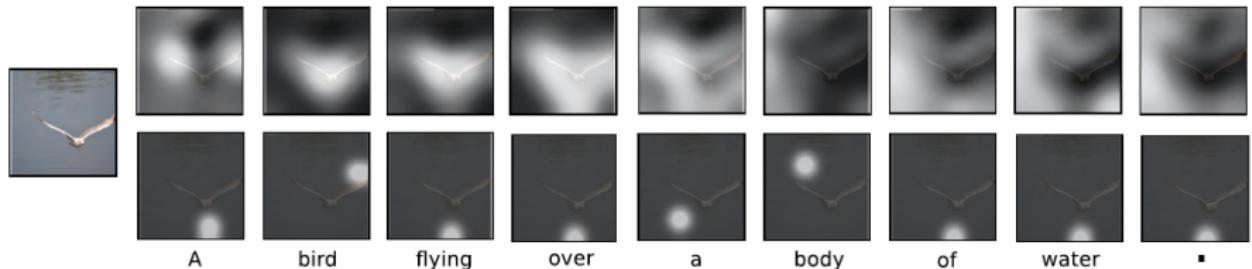


A woman is throwing a frisbee in a park.

Attention weighs CNN feature maps according to caption produced **so far** → allows for interpretation

Source: Adapted from [24]

Show-Attend-Tell [24] - Soft vs. Hard Attention



Soft (top row) vs. hard attention (bottom row). Both models produced the same sequence.

- Deterministic soft attention: Trained end-to-end
- Stochastic hard attention: Trained with reinforcement learning
- Sharp vs. fuzzy attention
- Slight performance benefit of hard attention in [24], but RL usually connected to much longer training times

Self-Attention

- Computes attention of sequence to "itself"
- Example: "*The animal didn't cross the street because it was too tired.*"
Does "it" refer to "animal" or "street"?

Source: [3]

Self-Attention

- Computes attention of sequence to "itself"
- Example: "*The animal didn't cross the street because it was too tired.*"
Does "it" refer to "animal" or "street"?
- Allows to enrich **representation** of tokens with context information
- Important for machine reading, question answering, reasoning, ...

Source: [3]

Self-Attention

- Computes attention of sequence to "itself"
- Example: "*The animal didn't cross the street because it was too tired.*"
Does "it" refer to "animal" or "street"?
- Allows to enrich **representation** of tokens with context information
- Important for machine reading, question answering, reasoning, ...

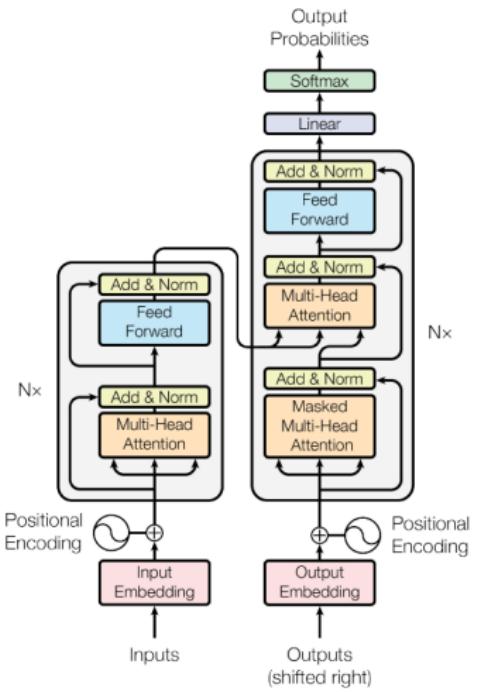
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

Self-attention w. r. t. word in red.

Source: [3]

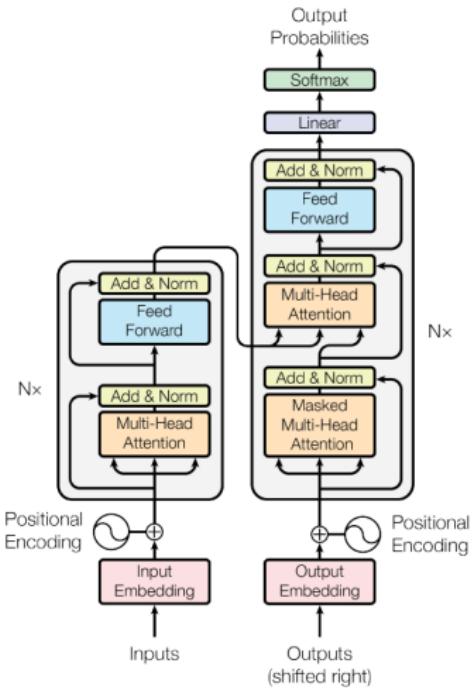
Attention is all you need (AIAYN) [23]

- Vaswani et al. [23]: Machine translation based **only on attention**, no convolution or recurrence
- Core idea: Iteratively improve representation by self-attention
- “Transformer” architecture



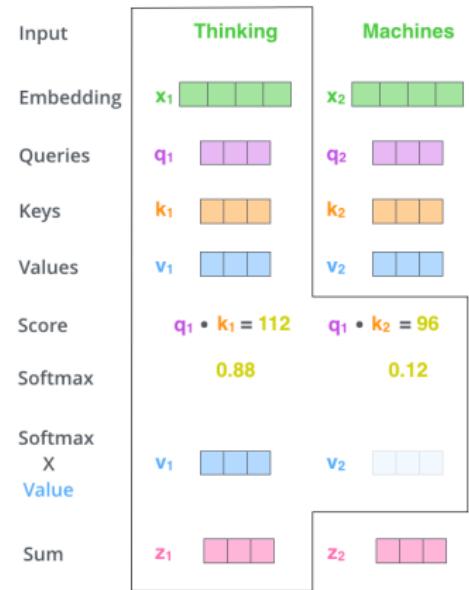
Attention is all you need (AIAYN) [23]

- Vaswani et al. [23]: Machine translation based **only on attention**, no convolution or recurrence
- Core idea: Iteratively improve representation by self-attention
- “Transformer” architecture
- Core blocks encoder:
 - Self-attention step
 - Local fully connected layer



AIAYN: Encoder

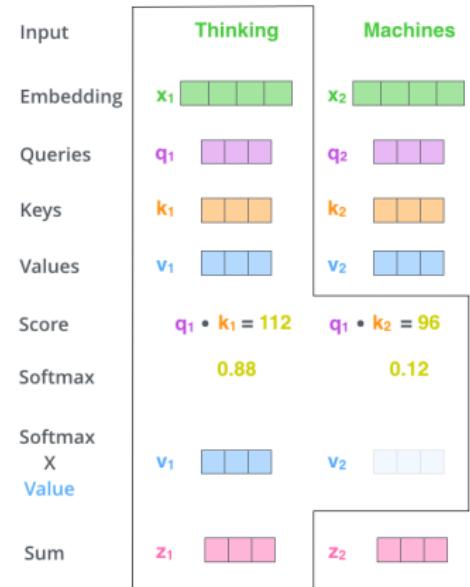
- Each input token is translated into vector of same length using an **embedding** algorithm



Source: <http://jalammar.github.io/illustrated-transformer/>

AIAYN: Encoder

- Each input token is translated into vector of same length using an **embedding** algorithm
 - Self attention: For each token, compute
 - query **q**: what a token is "looking for"
 - key **k**: "description" for query
 - value **v**: potential "information"
- using trainable weights \mathbf{W}_q , \mathbf{W}_k , \mathbf{W}_v

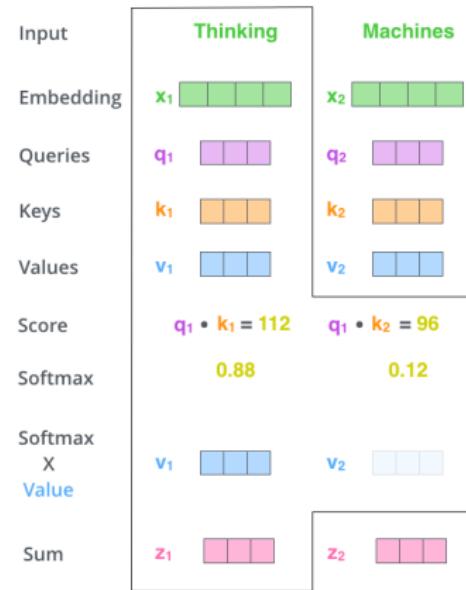


Source: <http://jalammar.github.io/illustrated-transformer/>

AIAYN: Encoder

- Each input token is translated into vector of same length using an **embedding** algorithm
- Self attention: For each token, compute
 - query **q**: what a token is "looking for"
 - key **k**: "description" for query
 - value **v**: potential "information"
 using trainable weights \mathbf{W}_q , \mathbf{W}_k , \mathbf{W}_v
- Alignment between query and key (scaled dot product) determines influence of elements in **v**:

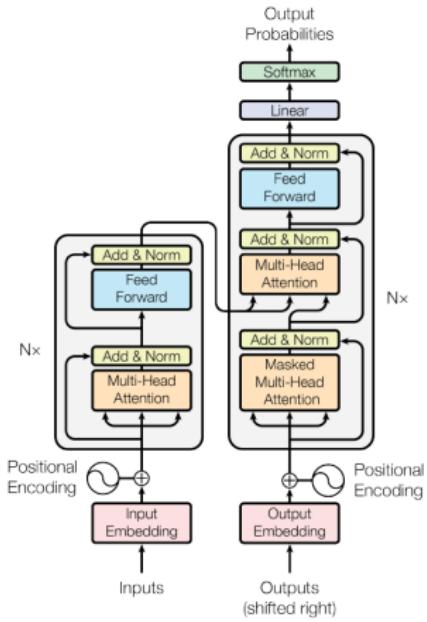
$$\text{attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{softmax}\left(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d_k}}\right)\mathbf{v}$$



Source: <http://jalammar.github.io/illustrated-transformer/>

AIAYN: Encoder (cont.)

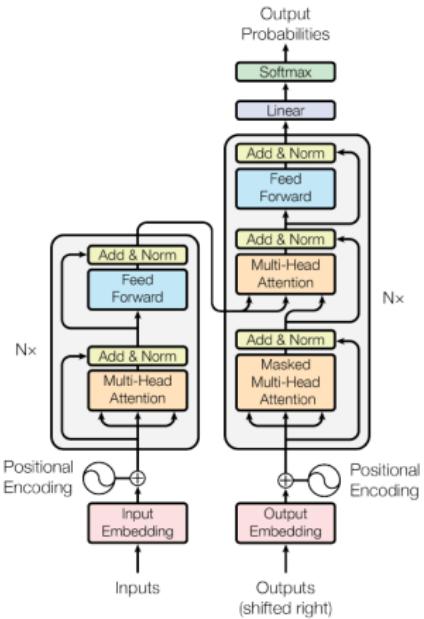
- Multi-Head attention: Multiple attention vectors per token
→ Representation subspaces
- Local (per token) recombination using a fully connected layer



Source: [23]

AIAYN: Encoder (cont.)

- Multi-Head attention: Multiple attention vectors per token
→ **Representation subspaces**
- Local (per token) recombination using a fully connected layer
- Stacked attention blocks ($6 \times$ in [23])
→ step-by-step context integration
- Additional positional encoding to represent word order

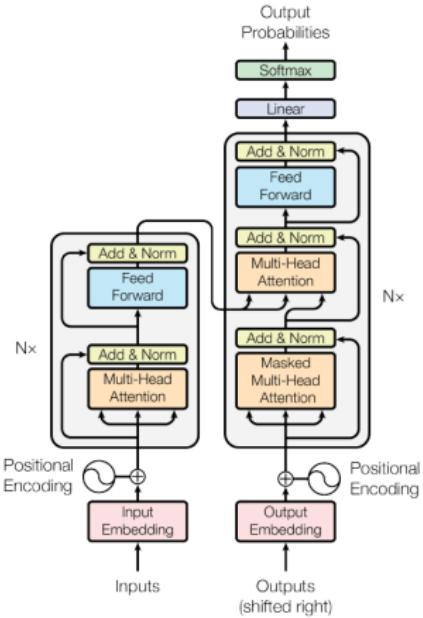


Source: [23]

AIAYN: Decoder

Decoder follows same concept as encoder:

- Additional input/output attention step
- Self-attention only computed on **previous** outputs



Source: [23]

AIAYN: But why?

- Allows for integration of knowledge independent of distance
- Positional encoding still allows to learn convolution-like steps
- Extremely versatile
- Extensions allow pretraining on unlabeled text (e.g., [4])
- **State-of-the-art** performance and **faster** training

Attention: Summary

- Alignment and/or relevance of input elements w. r. t. output elements
- Attention scores allow interpretation
- Allows to reformulate non-sequential tasks as sequential ones
- Attention alone very powerful (Transformer)
- State-of-the-art technique for many NLP tasks, e. g., machine translation, question answering, sentiment analysis, etc.
- ... but also in vision (e. g. SAGAN [27])
- Attention often used in combination with convolutions → Attention layers as replacements for convolutions? [18]

**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?
- What is the idea behind attention?
- What is the difference between "normal" and self-attention
- How does the transformer architecture avoid recurrence and convolutions?

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] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: 3rd International Conference on Learning Representations, ICLR 2015, San Di 2015.
- [2] T. B. Brown, D. Mané, A. Roy, et al. "Adversarial Patch". In: ArXiv e-prints (Dec. 2017). arXiv: 1712.09665 [cs.CV].
- [3] Jianpeng Cheng, Li Dong, and Mirella Lapata. "Long Short-Term Memory-Networks for Machine Reading". In: CoRR abs/1601.06733 (2016). arXiv: 1601.06733.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: CoRR abs/1810.04805 (2018). arXiv: 1810.04805.

References II

- [5] Neil Frazer. Neural Network Follies. 1998. URL:
<https://neil.fraser.name/writing/tank/> (visited on 01/07/2018).
- [6] 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.
- [7] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines". In: CoRR abs/1410.5401 (2014). arXiv: 1410.5401.
- [8] Karol Gregor, Ivo Danihelka, Alex Graves, et al. "DRAW: A Recurrent Neural Network For Image Generation". In: Proceedings of the 32nd International Conference on Machine Learning. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1462–1471.

References III

- [9] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, et al. "Neural Machine Translation in Linear Time". In: [CoRR abs/1610.10099](#) (2016). arXiv: [1610.10099](#).
- [10] 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.
- [11] Andrej Karpathy. [t-SNE visualization of CNN codes](#). URL: <http://cs.stanford.edu/people/karpathy/cnnembed/> (visited on 01/07/2018).
- [12] 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](#).

References IV

- [13] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421.
- [14] 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.
- [15] 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.

References V

- [16] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. "Recurrent Models of Visual Attention". In: [CoRR abs/1406.6247](#) (2014). arXiv: 1406.6247.
- [17] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: [Distill](#) (2017). <https://distill.pub/2017/feature-visualization>.
- [18] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, et al. "Stand-Alone Self-Attention in Vision Models". In: [arXiv e-prints](#), arXiv:1906.05909 (June 2019), arXiv:1906.05909. arXiv: 1906.05909 [cs.CV].
- [19] 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 VI

- [20] 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.
- [21] 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.
- [22] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: [CoRR abs/1711.10925](#) (2017). arXiv: 1711.10925.
- [23] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention Is All You Need". In: [CoRR abs/1706.03762](#) (2017). arXiv: 1706.03762.

References VII

- [24] Kelvin Xu, Jimmy Ba, Ryan Kiros, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: [CoRR abs/1502.03044](#) (2015). arXiv: [1502.03044](#).
- [25] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, et al. "Understanding Neural Networks Through Deep Visualization". In: [CoRR abs/1506.06579](#) (2015). arXiv: [1506.06579](#).
- [26] 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.

References VIII

- [27] Han Zhang, Ian Goodfellow, Dimitris Metaxas, et al. “Self-Attention Generative Adversarial Networks”. In: Proceedings of the 36th International Conference on Machine Learning. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 7354–7363.