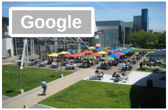
– SSDA 2019 / Islamabad –
Aug 20, 2019

Convolutional Neural Networks (CNNs)

Prof. Dr. Adrian Ulges
DCSM Department
RheinMain University of Applied Sciences

Outline

Introduction



Working Group "Learning and Visual Systems" > lavis.cs.hs-rm.de

Prof. Dr. Ralf Dörner

Computer Graphics
Visualization

Prof. Dr. Dirk Krechel

Content Analytics
Knowledge Management

Prof. Dr. Ulrich Schwanecke

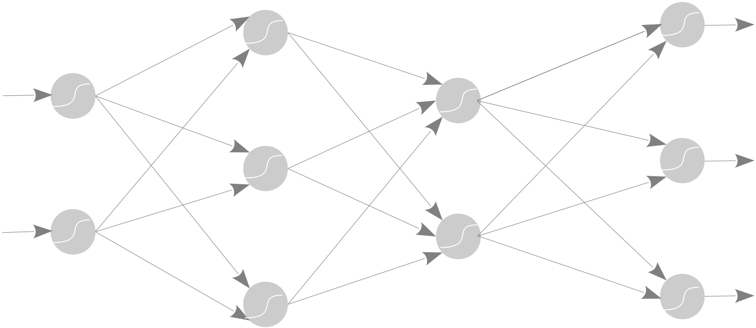
Computer Vision
Mixed Reality

Prof. Dr. Adrian Ulges

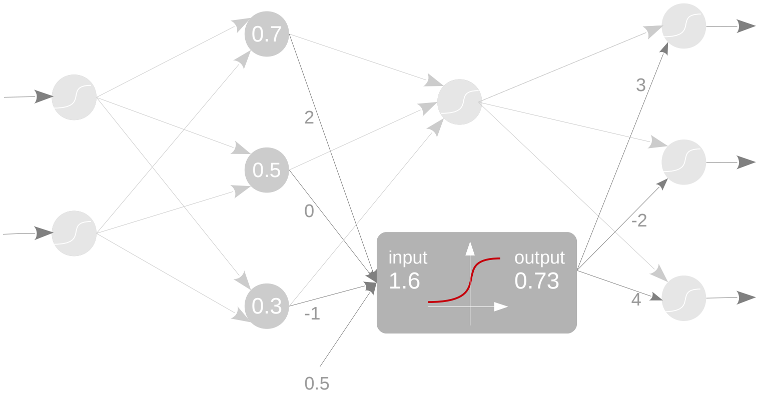
Machine Learning
Data Science

Outline

Neural Networks



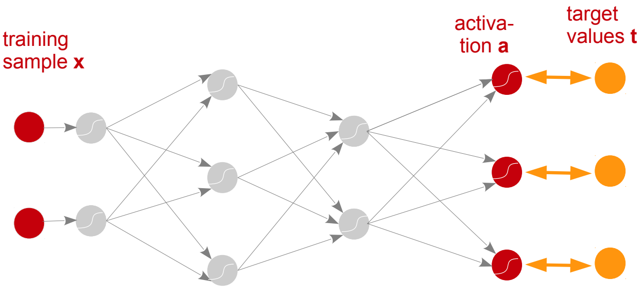
Neural Networks



Each neuron...

1. ... weighs its inputs
2. ... aggregates incoming energy
3. ... applies an activation function

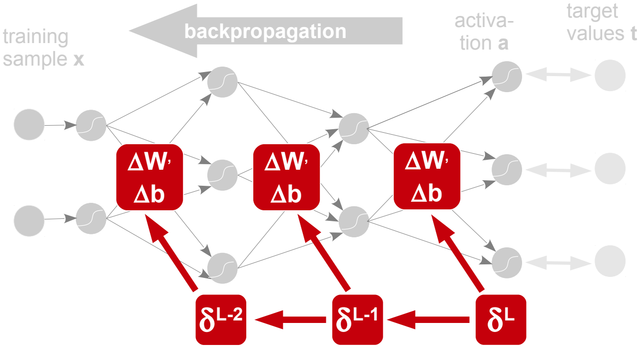
Neural Networks Training: Backpropagation



Training = Adjusting Weights

- iteratively, pick a training sample.
- compute the deviation from the target with a **loss** function.

Neural Networks Training: Backpropagation



Training = Adjusting Weights

- iteratively, pick a training sample.
- compute the deviation from the target with a **loss** function.
- minimize loss by **gradient descent**.

Enter: Deep Learning

	SVMs	**Deep Learning**
	1990s – 2000s	2012 – today
nonlinearity through...	kernels	stacking layers *(at least 3)*

Enter: Deep Learning

	SVMs	**Deep Learning**
	1990s – 2000s	2012 – today
nonlinearity through...	kernels	stacking layers *(at least 3)*
optimization	easy	hard

Enter: Deep Learning

	SVMs	**Deep Learning**
	1990s – 2000s	2012 – today
nonlinearity through...	kernels	stacking layers *(at least 3)*
optimization	easy	hard
#samples	$\leq 20K$	up to 10^{10}

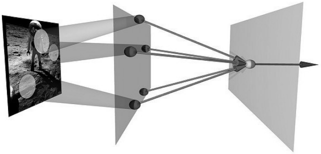
Enter: Deep Learning

	SVMs	**Deep Learning**
	1990s – 2000s	2012 – today
nonlinearity through...	kernels	stacking layers *(at least 3)*
optimization	easy	hard
#samples	$\leq 20K$	up to 10^{10}
feature engineering	heavy	none *(representation learning)*

Enter: Deep Learning

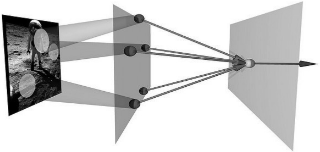
	SVMs	**Deep Learning**
	1990s – 2000s	2012 – today
nonlinearity through...	kernels	stacking layers *(at least 3)*
optimization	easy	hard
#samples	$\leq 20K$	up to 10^{10}
feature engineering	heavy	none *(representation learning)*
facilitated by ...	—	▸ hardware *(GPUs)* ▸ clever topologies *(CNNs, LSTMs)* ▸ tweaks *(losses+activations, optimizers, shortlinks, regularizers, attention)*

Neural Networks on Images: General Thoughts

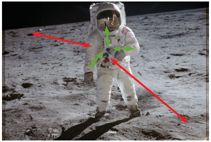


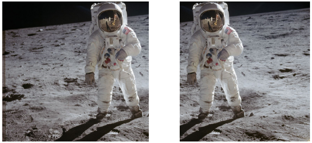
- A network **feeds pixel values** to neurons.
- Why not connect a neuron with **all pixels**?

Neural Networks on Images: General Thoughts



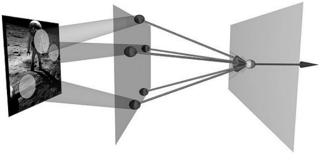
- A network **feeds pixel values** to neurons.
- Why not connect a neuron with **all pixels**?



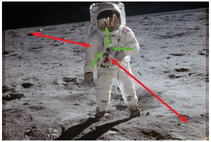


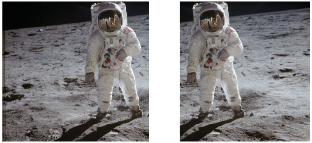
1. pixels interact mostly **locally**
2. **shift invariance**: recognize an object no matter where!

Neural Networks on Images: General Thoughts



- A network **feeds pixel values** to neurons.
- Why not connect a neuron with **all pixels**?





1. pixels interact mostly **locally**
2. **shift invariance**: recognize an object no matter where!

→ **Convolution gives us just that!**

Outline

Correlation/Convolution

A convolution is a transformation of **signals** *(here, images)*:

Definition (Input Images)

Let the input to a neural network be a (grayscale) **image x** *of size* $N \times M$*. Each pixel* (i, j) *has a value* $x(i, j) \in \mathbb{R}$*.*

Correlation/Convolution

A convolution is a transformation of **signals** *(here, images)*:

Definition (Input Images)

Let the input to a neural network be a (grayscale) **image x** *of size* $N \times M$*. Each pixel* (i, j) *has a value* $x(i, j) \in \mathbb{R}$*.*

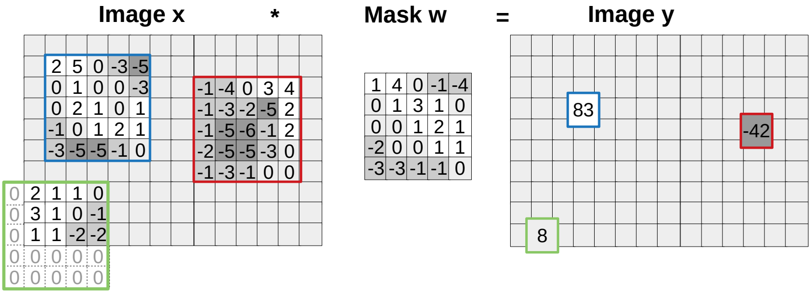
Definition (Convolution)

We define a **filter mask w** *as a matrix of size* $W \times W$*.*
Then the **convolution** *of image* **x** *with mask* **w** *is defined as:*

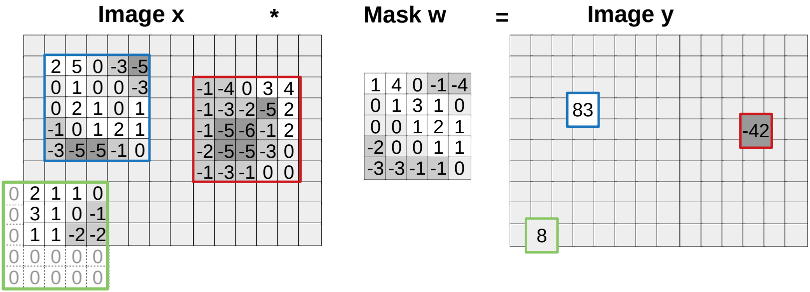
$$y(i,j) := \sum_{k,l=-W/2}^{W/2} w(k,l) \cdot x(i+k, j+l) \quad \text{// correlation}$$

$$y(i,j) := \sum_{k,l=-W/2}^{W/2} w(k,l) \cdot x(i-k, j-l) \quad \text{// convolution}$$

Correlation/Convolution (cont'd)



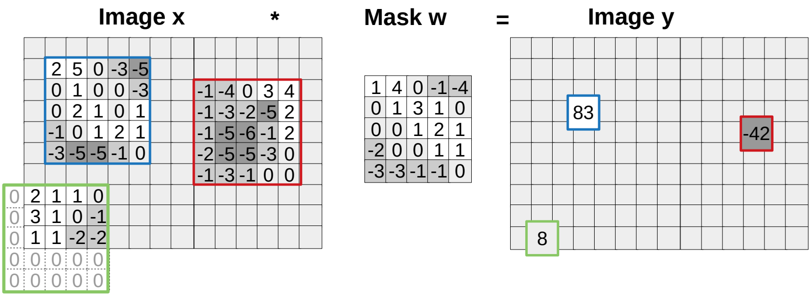
Correlation/Convolution (cont'd)



Remarks

- **Intuition**: **Shift the filter mask** over the image.
  At each position, compute the **weighted sum**.

Correlation/Convolution (cont'd)



Remarks

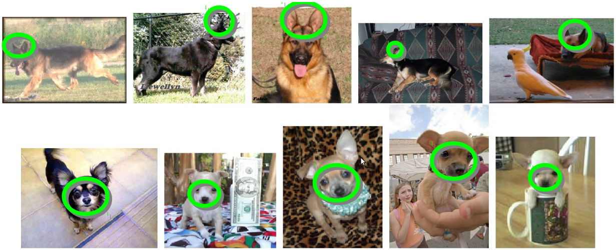
- **Intuition**: **Shift the filter mask** over the image.
  At each position, compute the **weighted sum**.
- At the **boundary**, the mask reaches outside the image.
  We can **pad** the input image *(here: zero padding)*.

Convolution = Feature Detection image: [2]

- By carefully designing filter masks, convolution allows us to scan the image for certain **features** *(here, the t-junction in the "4")*.

Convolution = Feature Detection image: [2]

- By carefully designing filter masks, convolution allows us to scan the image for certain **features** *(here, the t-junction in the "4")*.
- The resulting **feature map** shows where "interesting" regions in the image are.



Outline

Convolutional Neural Networks

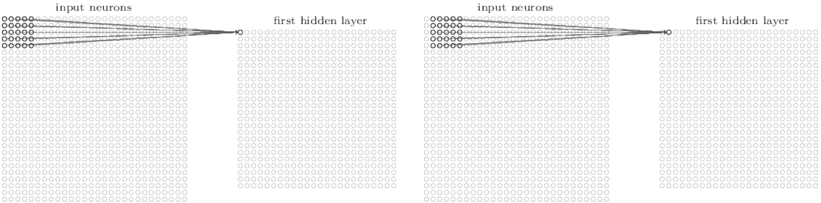
Idea: a neural network that applies convolutions (=CNN)

- **Layer 1**: run filters over the image.
- **Layer 2**: classify based on feature maps.

Convolutional Neural Networks

Idea: a neural network that applies convolutions (=CNN)

- **Layer 1**: run filters over the image.
- **Layer 2**: classify based on feature maps.
- **Such a convolutional neural network can learn its filter masks by backpropagation!**

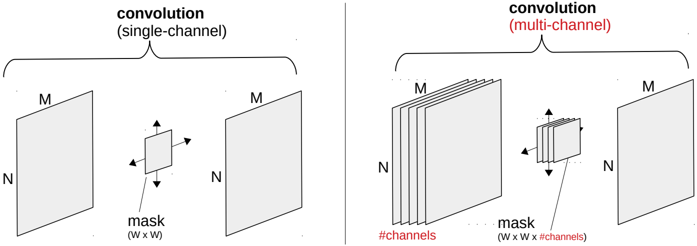


Convolutional Layers: Multi-Channel Input

- In practice, inputs to a convolution can have **multiple channels** *(e.g. color images: R,G,B)*.

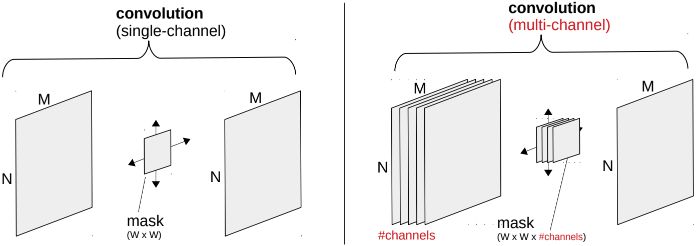
Convolutional Layers: Multi-Channel Input

- In practice, inputs to a convolution can have **multiple channels** *(e.g. color images: R,G,B)*.



Convolutional Layers: Multi-Channel Input

- In practice, inputs to a convolution can have **multiple channels** *(e.g. color images: R,G,B)*.



- We extend the convolution to sum **over the channels** too:

$$y(i,j) := \sum_{k,l=-W/2}^{W/2} \sum_{c=1}^{\#channels} w(k,l,c) \cdot x(i+k, j+l, c)$$

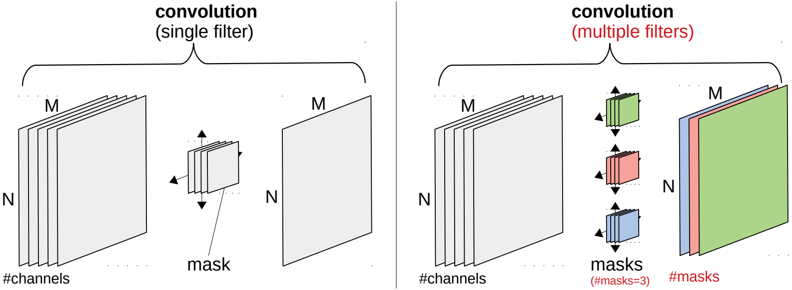
- Input and mask become 3D data "cubes" *(tensors)*.

Convolutional Layers: Multi-Channel Output

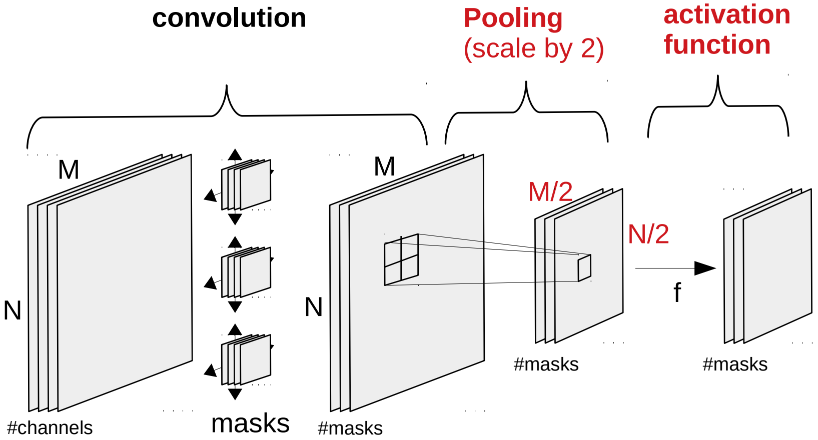
- In practice, we are not interested in detecting only <u>one</u> feature.
- We apply **multiple filters**, obtaining **multiple** feature maps.

Convolutional Layers: Multi-Channel Output

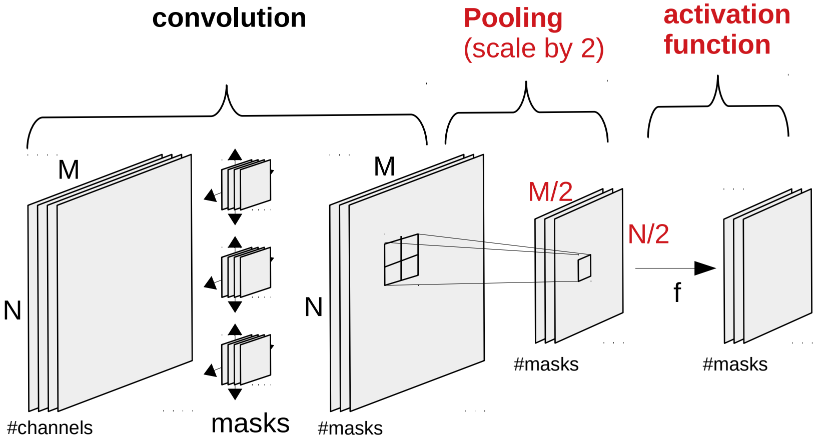
- In practice, we are not interested in detecting only one feature.
- We apply **multiple filters**, obtaining **multiple** feature maps.
- For K filters, we obtain a $N \times M \times K$ output tensor.



Convolutional Layers: Pooling

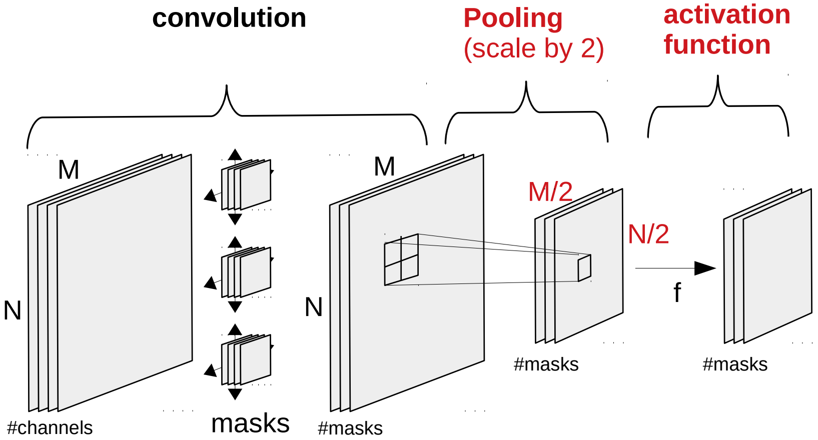


Convolutional Layers: Pooling



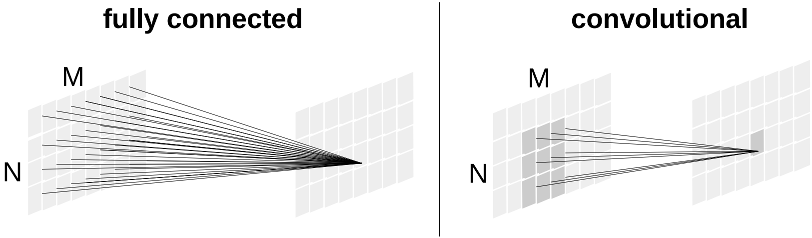
- Finally, we downscale the output masks using **pooling**.
- Pooling simply picks the mean or max value out of 2×2 pixels.

Convolutional Layers: Pooling



- Finally, we downscale the output masks using **pooling**.
- Pooling simply picks the mean or max value out of 2×2 pixels.
- We also apply an activation function for each pixel.

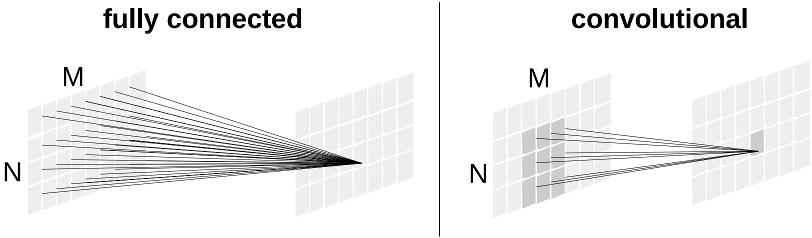
Convoution = Few Weights



Fully Connected Layer

- $N \times M$ pixels left, $N \times M$ pixels right.
- All are connected pairwise: $(N \times M)^2$ edges!
- **Example**: 320×240 input → 5.8 bio. parameters ☹

Convoution = Few Weights



Fully Connected Layer

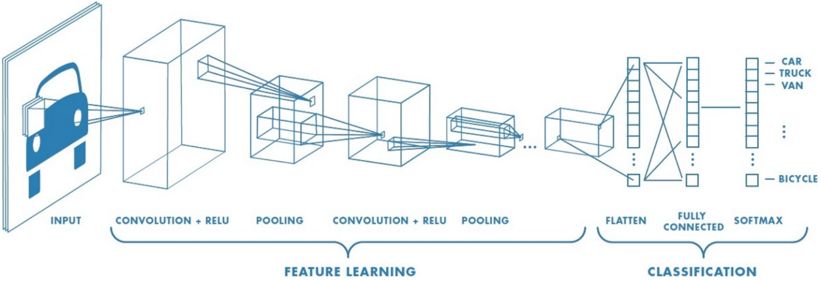
- $N \times M$ pixels left, $N \times M$ pixels right.
- All are connected pairwise: $(N \times M)^2$ edges!
- **Example**: 320×240 input → 5.8 bio. parameters ☹

Convolutional Layer

- Each pixel has a small local **receptive field**.
- K filters. Each filter has $W \times W$ values.
- **Example**: 20 filters, 5×5 → 500 parameters ☺

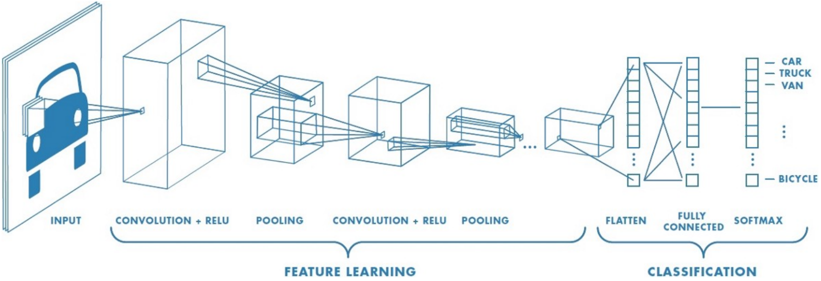
Outline

From CNNs to Deep CNNs



- State-of-the-Art CNNs are deep: They repeat convolution and pooling multiple times.

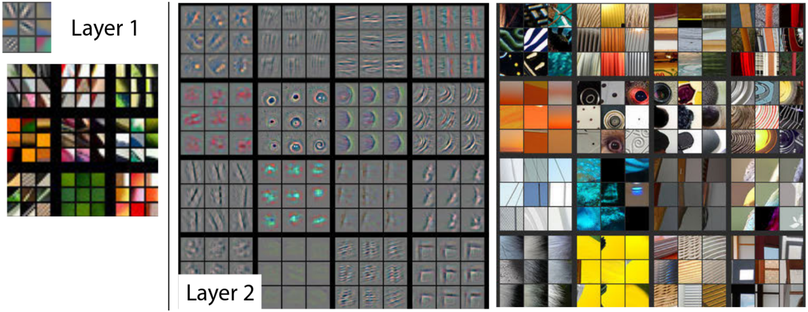
From CNNs to Deep CNNs



- State-of-the-Art CNNs are deep: They repeat convolution and pooling multiple times.
- Spatial resolution decreases.
- The number of kernels increases.

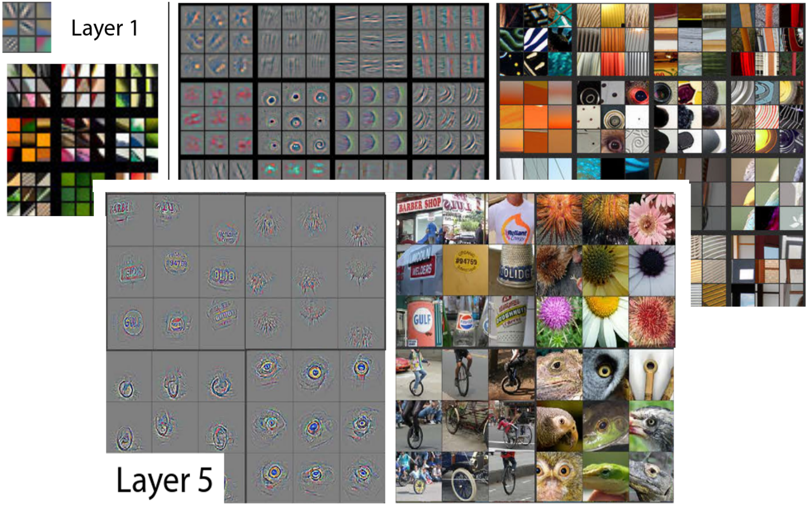
Deep CNNs images: [6]

With more layers, the level of abstraction increases

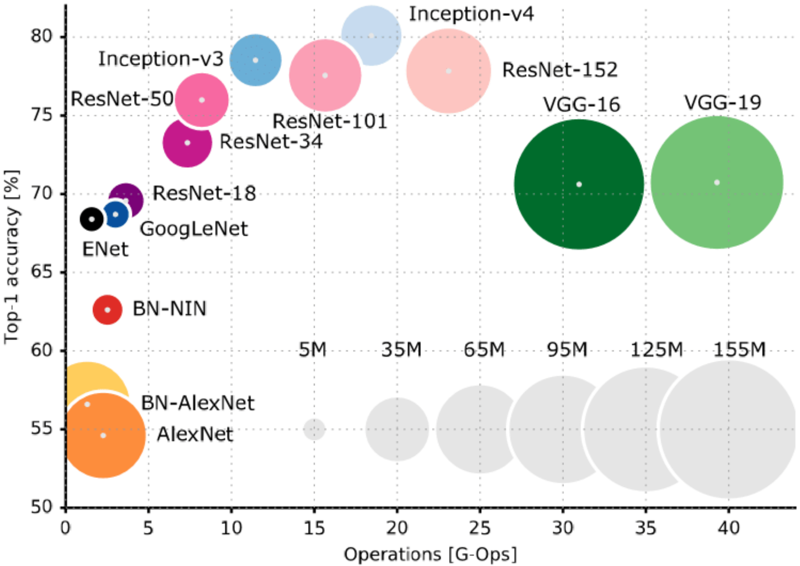


Deep CNNs images: [6]

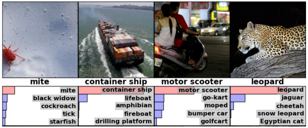
With more layers, the level of abstraction increases



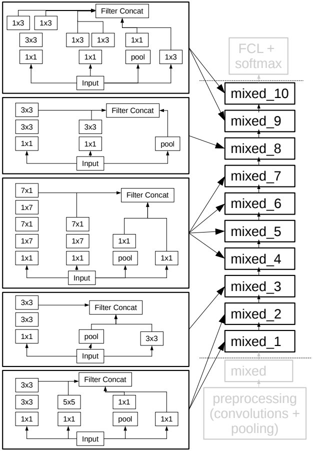
DeepCNNs: Architectures image: [5]

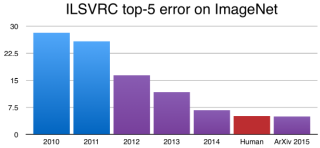


Example: Inception v3 [1]

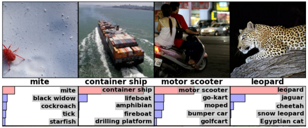


- A **deep CNN** *(convolutional neural network)*
- 22 layers, about 25 mio. parameters
- 5 bio. multiply-adds per inference
- pre-trained on 1.2 mio. images to recognize 1000 object categories

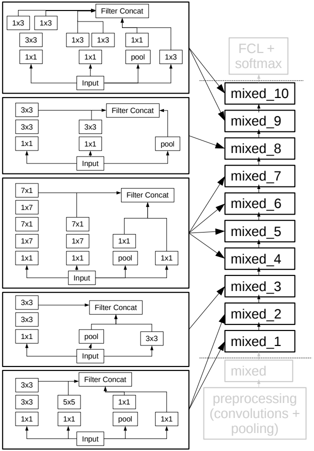


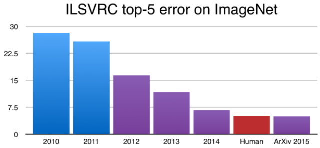


Example: Inception v3 [1]

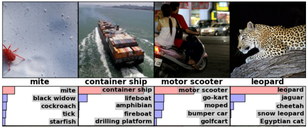


- A **deep CNN** *(convolutional neural network)*
- 22 layers, about 25 mio. parameters
- 5 bio. multiply-adds per inference
- pre-trained on 1.2 mio. images to recognize 1000 object categories
- **human-level object recognition** (ImageNet: 6.8% top-5-error)

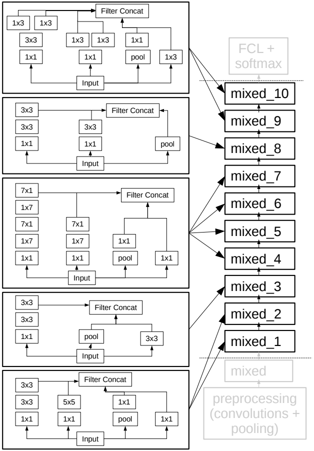


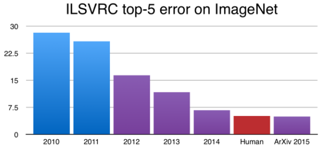


Example: Inception v3 [1]

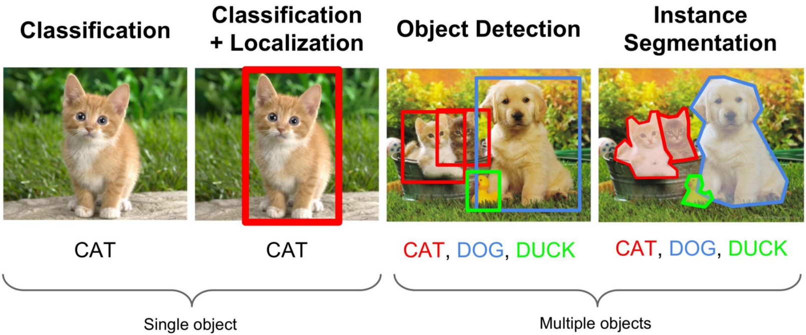


- A **deep CNN** *(convolutional neural network)*
- 22 layers, about 25 mio. parameters
- 5 bio. multiply-adds per inference
- pre-trained on 1.2 mio. images to recognize 1000 object categories
- **human-level object recognition** (ImageNet: 6.8% top-5-error)
- good basis for **transfer learning**.



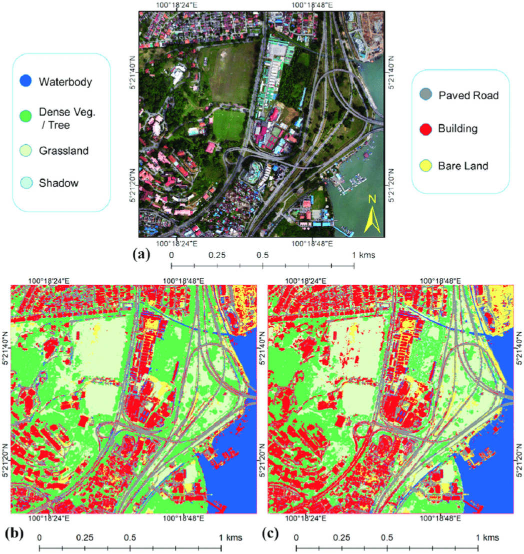


Tasks



- segmentation
- object detection
- object categorization
- similarity matching
- ...

Tasks (cont'd) image: [3]



Outline

Deep Learning Toolboxes

Deep Learning has become easy...

- Multiple **Frameworks**
  push deep learning
  (Tensorflow, Pytorch, Keras, ...)

Deep Learning Toolboxes

Deep Learning has become easy...

- Multiple **Frameworks** push deep learning *(Tensorflow, Pytorch, Keras, ...)*
- These are well-supported and widely used *(Google, Facebook, Nvidia, DropBox, ebay, airbnb, Airbus, Intel, Uber, ...)*

Deep Learning Toolboxes

Deep Learning has become easy...

- Multiple **Frameworks** push deep learning *(Tensorflow, Pytorch, Keras, ...)*
- These are well-supported and widely used *(Google, Facebook, Nvidia, DropBox, ebay, airbnb, Airbus, Intel, Uber, ...)*

Typical Features

- Flexible design of neural networks as **Flow Graphs**

Deep Learning Toolboxes

Deep Learning has become easy...

- Multiple **Frameworks** push deep learning *(Tensorflow, Pytorch, Keras, ...)*
- These are well-supported and widely used *(Google, Facebook, Nvidia, DropBox, ebay, airbnb, Airbus, Intel, Uber, ...)*

Typical Features

- Flexible design of neural networks as **Flow Graphs**
- **Backpropagation** built-in *(automatic differentiation)*

Deep Learning Toolboxes

Deep Learning has become easy...

- Multiple **Frameworks** push deep learning *(Tensorflow, Pytorch, Keras, ...)*
- These are well-supported and widely used *(Google, Facebook, Nvidia, DropBox, ebay, airbnb, Airbus, Intel, Uber, ...)*

Typical Features

- Flexible design of neural networks as **Flow Graphs**
- **Backpropagation** built-in *(automatic differentiation)*
- Out-of-the-box **Parallelization** on CPUs / GPUs

Deep Learning Toolboxes

Deep Learning has become easy...

- Multiple **Frameworks** push deep learning *(Tensorflow, Pytorch, Keras, ...)*
- These are well-supported and widely used *(Google, Facebook, Nvidia, DropBox, ebay, airbnb, Airbus, Intel, Uber, ...)*

Typical Features

- Flexible design of neural networks as **Flow Graphs**
- **Backpropagation** built-in *(automatic differentiation)*
- Out-of-the-box **Parallelization** on CPUs / GPUs
- **pre-trained networks** available *(model zoos)*

Deep Learning Toolboxes

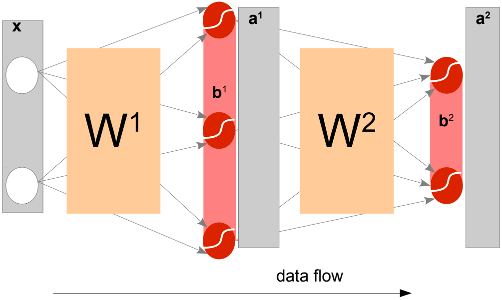
Deep Learning has become easy...

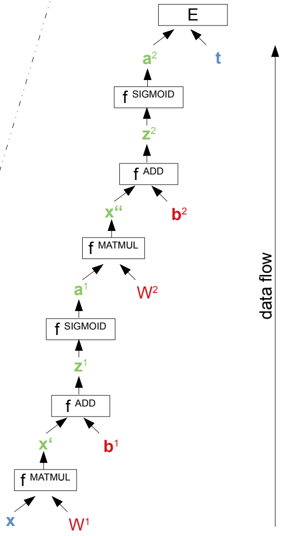
- Multiple **Frameworks** push deep learning *(Tensorflow, Pytorch, Keras, ...)*
- These are well-supported and widely used *(Google, Facebook, Nvidia, DropBox, ebay, airbnb, Airbus, Intel, Uber, ...)*

Typical Features

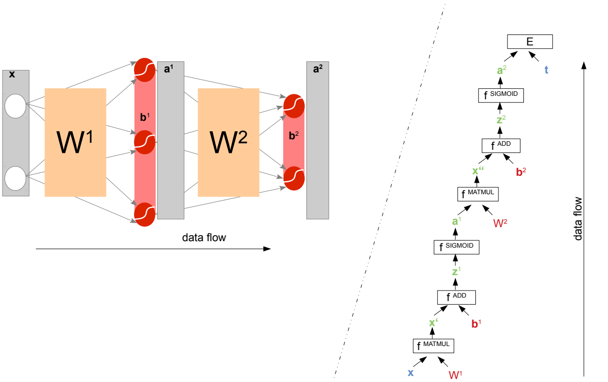
- Flexible design of neural networks as **Flow Graphs**
- **Backpropagation** built-in *(automatic differentiation)*
- Out-of-the-box **Parallelization** on CPUs / GPUs
- **pre-trained networks** available *(model zoos)*
- **Visualization** of network behavior *(e.g., tensorboard)*

Neural Networks as Flow Graphs





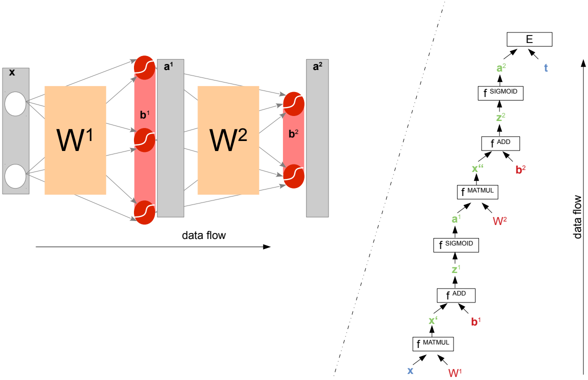
Neural Networks as Flow Graphs



Deep Learning Frameworks view NNs as so-called flow graphs:

- The boxes correspond to operations/functions:
  Matrix multiplications, vector-adds, sigmoids, ...
- The nodes are data objects: vectors, matrices, or more generally n-dimensional *tensors*.

Neural Networks as Flow Graphs



There are three different kinds of tensors:

1. **Inputs**: features $\mathbf{x}$, targets $\mathbf{t}$, ...
2. **Parameters**: weight matrices (W^2, W^3), biases $(\mathbf{b}^2, \mathbf{b}^3)$, ...
3. **Results** from applying operations / functions $(\mathbf{a}^2, \mathbf{a}^3)$

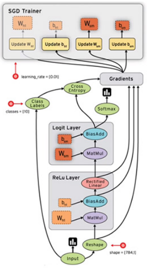
Tensorflow

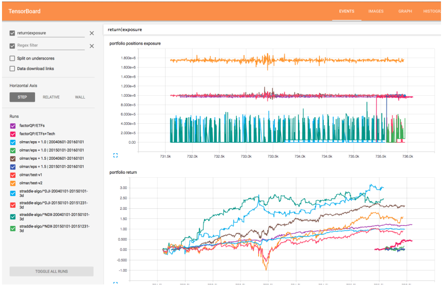
Tensorflow is the most commonly used deep learning framework.

Features

- developed by Google
- open-source (License: Apache 2.0)
- Interfaces: Python, C/C++
- Platforms: Linux, Mac OS X, Windows, Android

Tensorboard: Illustrations





Outline

Outline

Image-Graphics Retrieval

- R&D Project SMULGRAS (1 year, 2016-17)
- development of 2D-to-3D image-graphics search

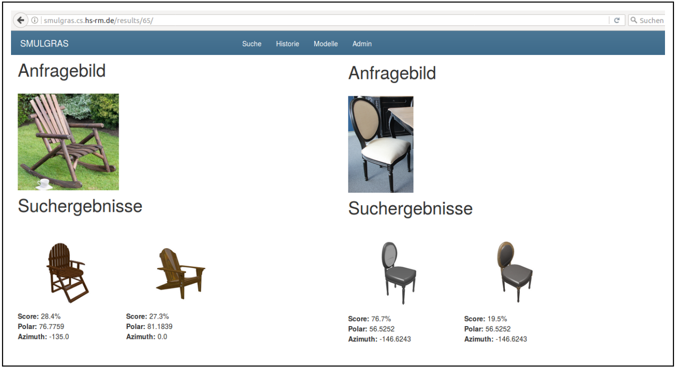
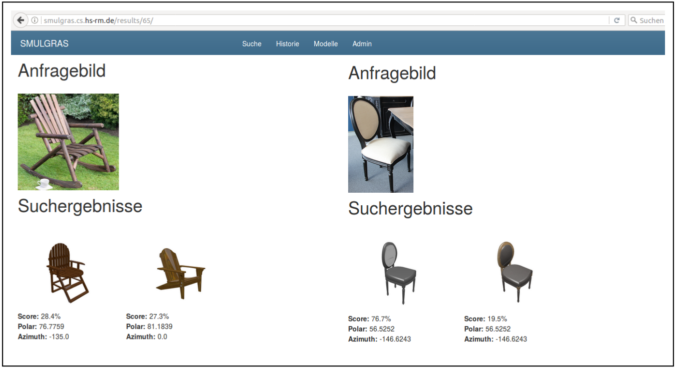


Image-Graphics Retrieval

- R&D Project SMULGRAS (1 year, 2016-17)
- development of 2D-to-3D image-graphics search



- Applications: customized product design, community-based modeling, copyright infringement, ...

Image-Graphics Retrieval

Challenges

photo to 3D model
(≈ 3D object recognition)

pose estimation

3D model registration

Image-Graphics Retrieval

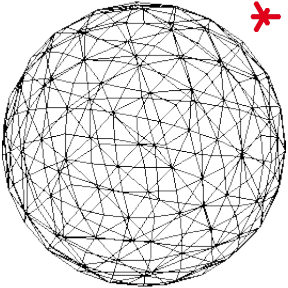
Challenges

photo to 3D model
(≈ 3D object recognition)

pose estimation

3D model registration

Preprocessing: Rendering

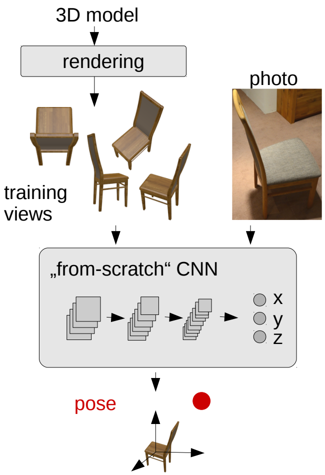


- representation of 3D model
  with rendered **views**
- **camera** sampling: Monte carlo / subdivision
- camera points at **object center**, roll $= 0°$
- **background**: plain, Flickr skybox
- **graphics**: high (raytracing, casted shadows)
  low (Phong shading)

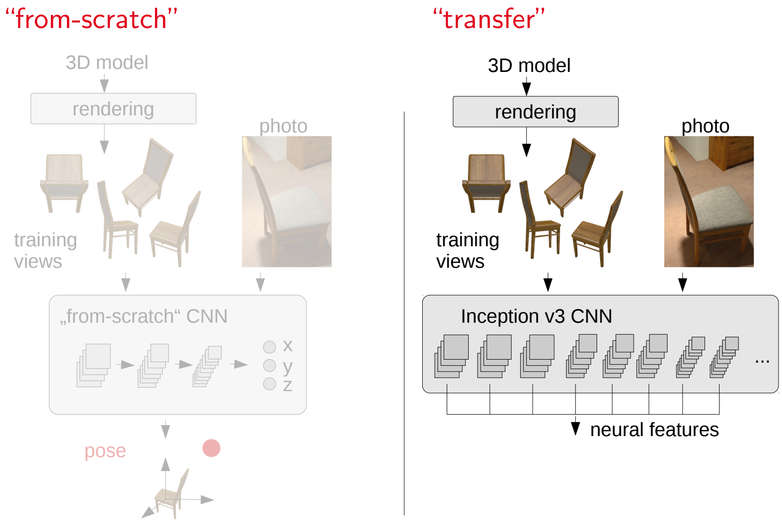
View-based Approach: Two Models

“from-scratch”

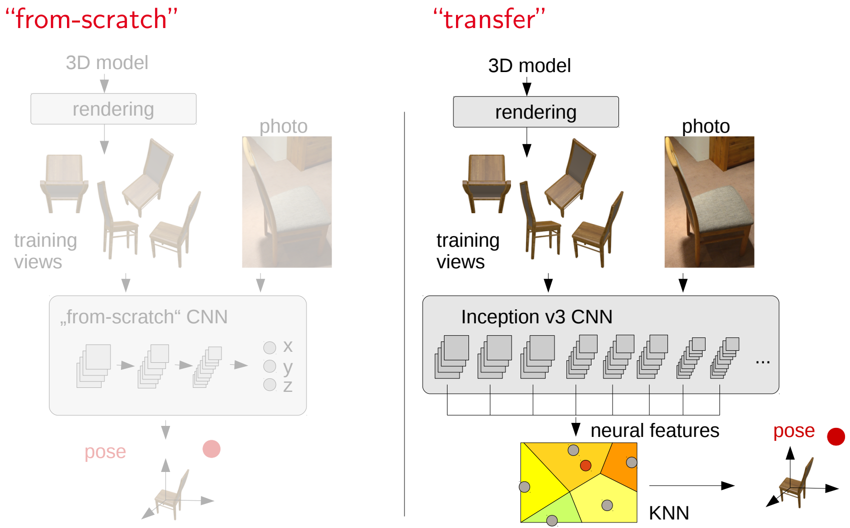
“transfer”



View-based Approach: Two Models



View-based Approach: Two Models



Experiments: Sample Results

- 200 models of chairs
  (≈ 40,000 views, subset of [4])
- 340 (calibrated) photos of chairs
  (self-captured, ground truth
  by chessboard marker)

Experiments: Sample Results

- 200 models of chairs
  (≈ 40,000 views, subset of [4])
- 340 (calibrated) photos of chairs
  (self-captured, ground truth
  by chessboard marker)

Recognition

Pose Estimation

Experiments: Pose Estimation

Setup

- 200-400 random training views per chair
- **accuracy measure**: angle between camera positions c and c’

$$E(c, c') := arccos\left(\frac{c^T \cdot c'}{||c|| \cdot ||c'||}\right)$$

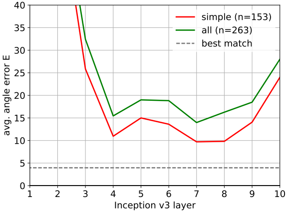
Experiments: Pose Estimation

Setup

- 200-400 random training views per chair
- **accuracy measure**: angle between camera positions c and c'

$$E(c, c') := arccos\left(\frac{c^T \cdot c'}{||c|| \cdot ||c'||}\right)$$

Results: Transfer Learning

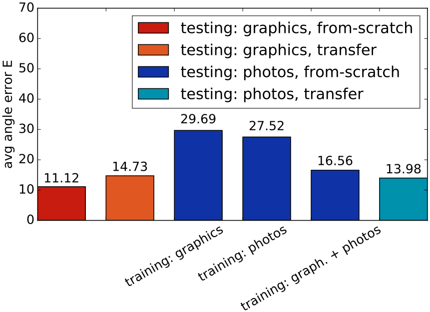


- best generalization: Inception Layer 7 (768x8x8 dimensions)

Experiments: Comparing both models

Testing on graphics

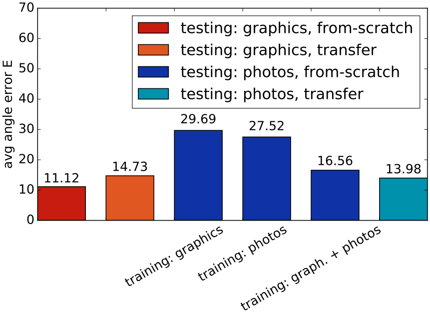
- generalization between models
- best results with from-scratch CNN
- average angle error of about **11.12°** (compared to 14.73°)



Experiments: Comparing both models

Testing on graphics

- generalization between models
- best results with from-scratch CNN
- average angle error of about **11.12°** (compared to 14.73°)



Testing on photos

- **from-scratch CNN**: strong **domain drift** observed
- **transfer learning**: model outperforms all from-scratch runs

Experiments: Sample results

- from-scratch CNN (left) and transfer learning (right)
- Example 1: Angle error of about 30°
- Example 4: Angle error of about 13.5°

The End

References I

[1] Mocha.jl: Deep Learning for Julia.
https://devblogs.nvidia.com/parallelforall/mocha-jl-deep-learning-julia/ (retrieved: Nov 2016).

[2] picture shared by Christoph Lampert.
contact: http://pub.ist.ac.at/~chl/.

[3] H. A H Al-Najjar, B. Kalantar, B. Pradhan, V. Saeidi, A. Abdul Halin, N. Ueda, and S. Mansor.
remote sensing land cover classification from fused dsm and uav images using convolutional neural networks.
Remote Sensing, 2019:1461, 06 2019.

[4] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic.
Seeing 3D chairs: Exemplar Part-based 2D-3D Alignment using a Large Dataset of CAD models.
In Proc. CVPR, pages 3762–3769, 2014.

[5] C. Kawatsu, F. Koss, A. Gillies, A. Zhao, J. Crossman, B. Purman, D. Stone, and D. Dahn.
Gesture recognition for robotic control using deep learning.
08 2017.

[6] M. D. Zeiler and R. Fergus.
Visualizing and Understanding Convolutional Networks.
CoRR, abs/1311.2901, 2013.