Check for updates

Course Notes

CreativeAI: Deep Learning for Graphics

SIGGRAPH Asia 2018

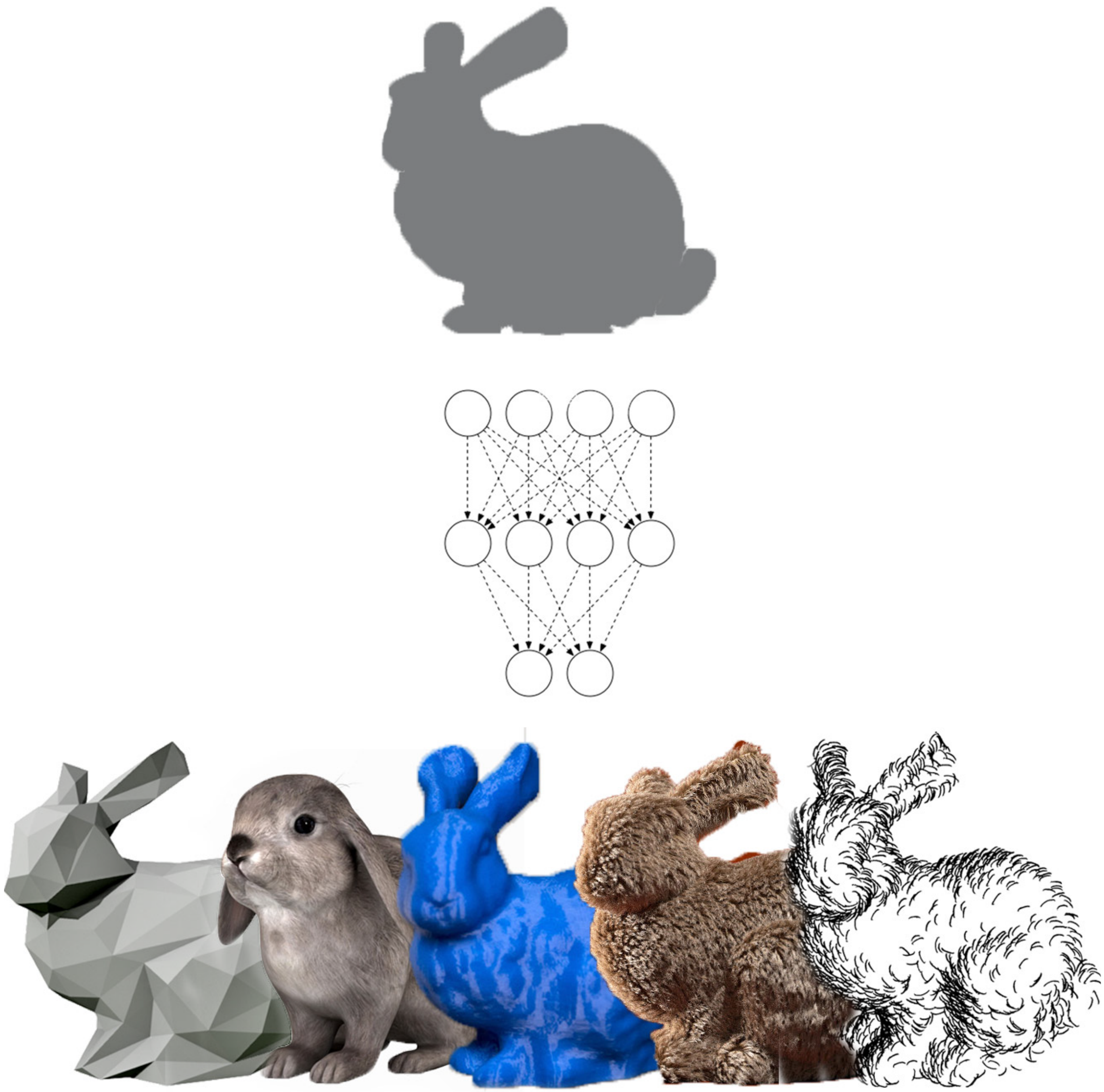
Niloy J. Mitra
UCL

Iasonas Kokkinos
UCL / Facebook

Paul Guerrero
UCL

Nils Thuerey
TU Munich

Tobias Ritschel
UCL



Abstract

In computer graphics, many traditional problems are now better handled by deep-learning based data-driven methods. In applications that operate on regular 2D domains, like image processing and computational photography, deep networks are state-of-the-art, beating dedicated hand-crafted methods by significant margins. More recently, other domains such as geometry processing, animation, video processing, and physical simulations have benefited from deep learning methods as well. The massive volume of research that has emerged in just a few years is often difficult to grasp for researchers new to this area. This tutorial gives an organized overview of core theory, practice, and graphics-related applications of deep learning.

1 Course Content and Syllabus

Introduction (10 min.) .... page 3

Niloy J. Mitra, Iasonas Kokkinos, Paul Guerrero, Nils Thuerey and Tobias Ritschel

Theory (30 min.) .... page 15

Niloy J. Mitra and Nils Thuerey

Neural Network Basics (30 min.) .... page 62

Niloy J. Mitra and Iasonas Kokkinos

Alternatives to Direct Supervision (30 min.) .... page 154

Paul Guerrero

(15 min. break)

Feature Visualization (15 min.) .... page 182

Tobias Ritschel

Image Domains (30 min.) .... page 187

Iasonas Kokkinos and Tobias Ritschel

3D Domains (30 min.) .... page 207

Paul Guerrero and Tobias Ritschel

Motion and Physics (30 min.) .... page 241

Nils Thuerey and Niloy J. Mitra

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
Copyright is held by the owner/author(s).
SA '18 Courses, December 04-07, 2018, Tokyo, Japan
ACM 978-1-4503-6026-5/18/12.
10.1145/3277644.3277775

2 About the Lecturers

Niloy J. Mitra leads the Smart Geometry Processing group in the Department of Computer Science at University College London. He received his PhD degree from Stanford University. His research interests include shape analysis, geometry processing, and computational design and fabrication. Niloy received the ACM Siggraph Significant New Researcher Award in 2013 and the BCS Roger Needham award in 2015. His work has twice been selected and featured as research highlights in the Communication of ACM, received best paper award at ACM Symposium on Geometry Processing 2014, best software SGP 2017, and Honourable Mention at Eurographics 2014.

Iasonas Kokkinos obtained the Diploma of Engineering in 2001 and the Ph.D. Degree in 2006 from the School of Electrical and Computer Engineering of the National Technical University of Athens in Greece, and the Habilitation Degree in 2013 from Universit Paris-Est. He is currently a faculty at the University College London and Facebook AI Research (FAIR). His research activity is currently focused on deep learning for computer vision, focusing in particular on structured prediction for deep learning and multi-task learning architectures. He has been awarded a young researcher grant by the French National Research Agency, has served as associate editor for the Image and Vision Computing journal and the Computer Vision and Image Understanding journal, and serves regularly as a reviewer and area chair for all major computer vision conferences and journals.

Paul Guerrero is a Post-Doc at University College London, working on shape analysis and image editing, combining methods from machine learning, optimization, and computational geometry. He received his PhD in computer science from Vienna University of Technology. Paul has published several research papers in high-quality journals, is a regular reviewer fo conferences and journals, and a conference IPC member.

Nils Thuerey is an Associate Professor at the Technical University of Munich (TUM). He works in the field of computer graphics, with a particular emphasis on physics simulations and deep learning algorithms. After studying computer science, Nils Thuerey acquired a PhD on liquid simulations in 2006 (both at the University of Erlangen-Nuremberg). Until 2010 he held a position as a post-doctoral researcher at ETH Zurich. He received a tech-Oscar from the AMPAS in 2013 for his research on controllable smoke effects. Subsequently, he worked for three years as R&D lead at ScanlineVFX, before he started at TUM in October 2013.

Tobias Ritschel is a Senior Lecturer at University College London. Previously he was a junior research group leader at the Max Planck Center for Visual Computing and Communication at Max Planck Institut Informatik. His interests include interactive and non-photorealistic rendering, human perception, and data-driven graphics. Ritschel received a PhD in computer graphics from Max Planck Institut Informatik. In 2011, he received the Eurographics PhD dissertation award and the Eurographics Young Researcher Award in 2014.

SIGGRAPH ASIA 2018 TOKYO

CreativeAI:
Deep Learning for Graphics

Niloy Mitra	Iasonas Kokkinos	Paul Guerrero	Nils Thuerey	Tobias Ritschel
UCL	UCL/Facebook	UCL	TU Munich	UCL

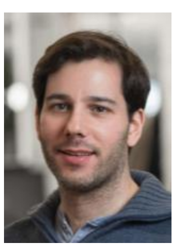
UCL

facebook Artificial Intelligence Research

TUM Technische Universität München

People

Niloy Mitra



Iasonas Kokkinos

Paul Guerrero

Nils Thuerey

Tobias Ritschel

SIGGRAPH ASIA 2018 TOKYO

2

Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics** 3

Code Examples

PCA/SVD basis
Linear Regression
Polynomial Regression
Stochastic Gradient Descent vs. Gradient Descent
Multi-layer Perceptron
Edge Filter 'Network'
Convolutional Network
Filter Visualization
Weight Initialization Strategies
Colorization Network
Autoencoder
Variational Autoencoder
Generative Adversarial Network

http://geometry.cs.ucl.ac.uk/dl4g/

Two-way Communication

- *This tutorial is given for the first time!*
- Our aim is to convey what we found to be relevant so far.
- You are invited/encouraged to give feedback
  - On-line form
  - Speakup. Please send us your criticism/comments/suggestions
  - Ask questions, please!
- **Thanks to many people who helped so far with slides/comments.**

Course Overview

- **Part I: Introduction and ML Basics**
- Part II: Supervised Neural Networks: Theory and Applications
- Part III: Unsupervised Neural Networks: Theory and Applications
- Part IV: Beyond Image Data

6

Representations in CG

- Images (e.g., pixel grid)
- Volume (e.g., voxel grid)
- Meshes (e.g., vertices/edges/faces)
- Animation (e.g., skeletal positions over time; cloth dynamics over time)
- Pointclouds (e.g., point arrays)
- Physics simulations (e.g., fluid flow over space/time)

SIGGRAPH ASIA 2018 TOKYO

7

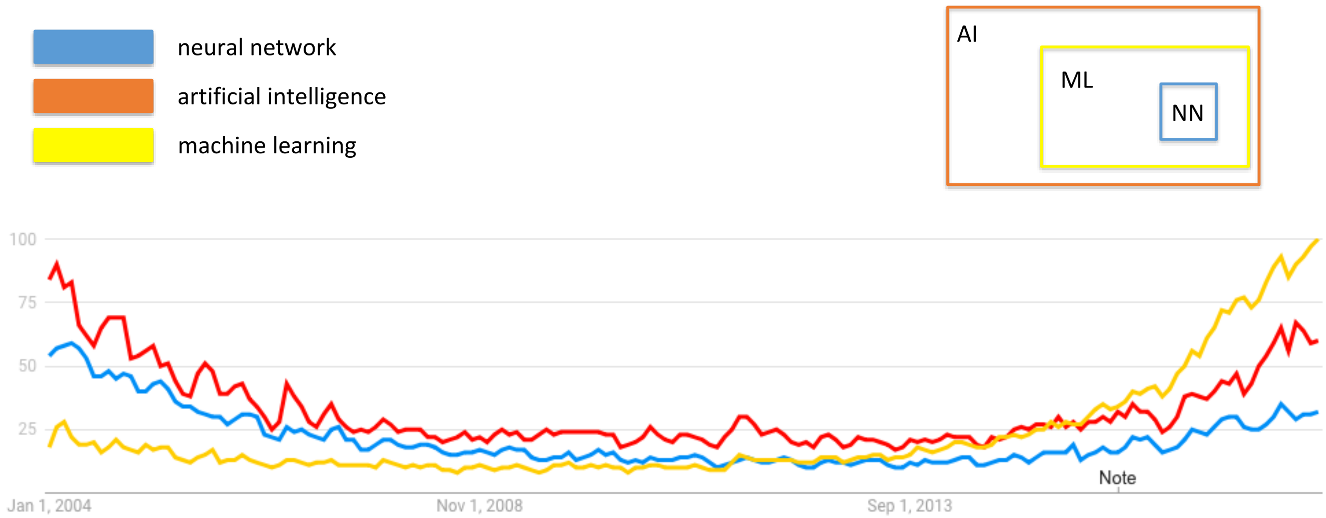
Problems in Computer Graphics

- Feature detection (image features, point features) $\mathbb{R}^{m\times m} \rightarrow \mathbb{Z}$
- Denoising, Smoothing, etc. $\mathbb{R}^{m\times m} \rightarrow \mathbb{R}^{m\times m}$
- Embedding, Distance computation $\mathbb{R}^{m\times m, m\times m} \rightarrow \mathbb{R}^{d}$
- Rendering $\mathbb{R}^{m\times m} \rightarrow \mathbb{R}^{m\times m}$
- Animation $\mathbb{R}^{3m\times t} \rightarrow \mathbb{R}^{3m}$
- Physical simulation $\mathbb{R}^{3m\times t} \rightarrow \mathbb{R}^{3m}$
- Generative models $\mathbb{R}^{d} \rightarrow \mathbb{R}^{m\times m}$

SIGGRAPH ASIA 2018 TOKYO

8

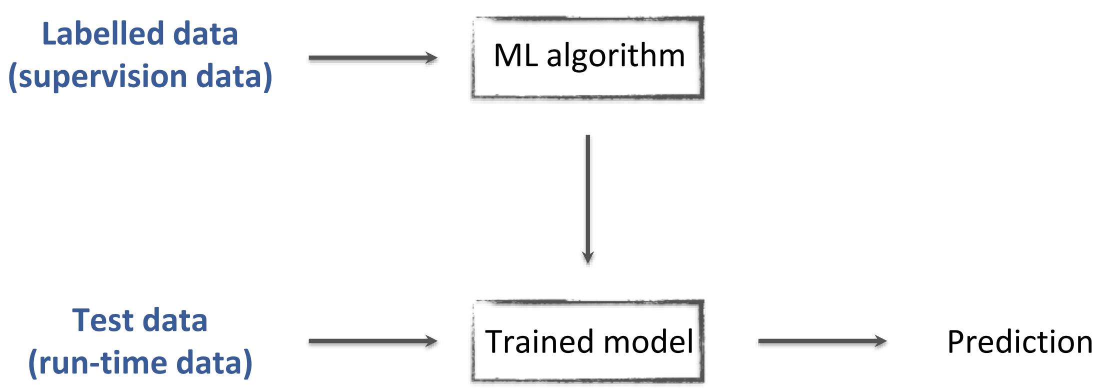
Rise of Machine Learning



SIGGRAPH ASIA 2018 TOKYO

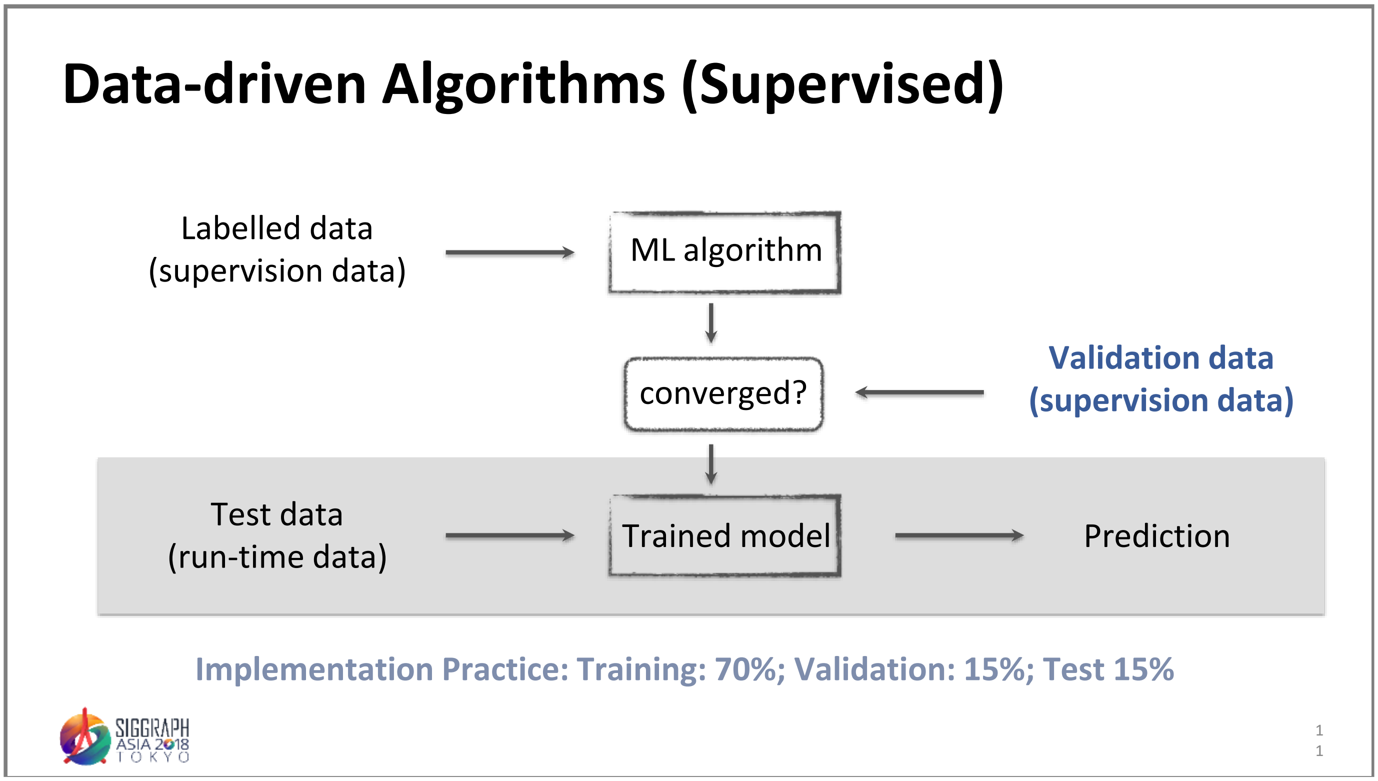
9

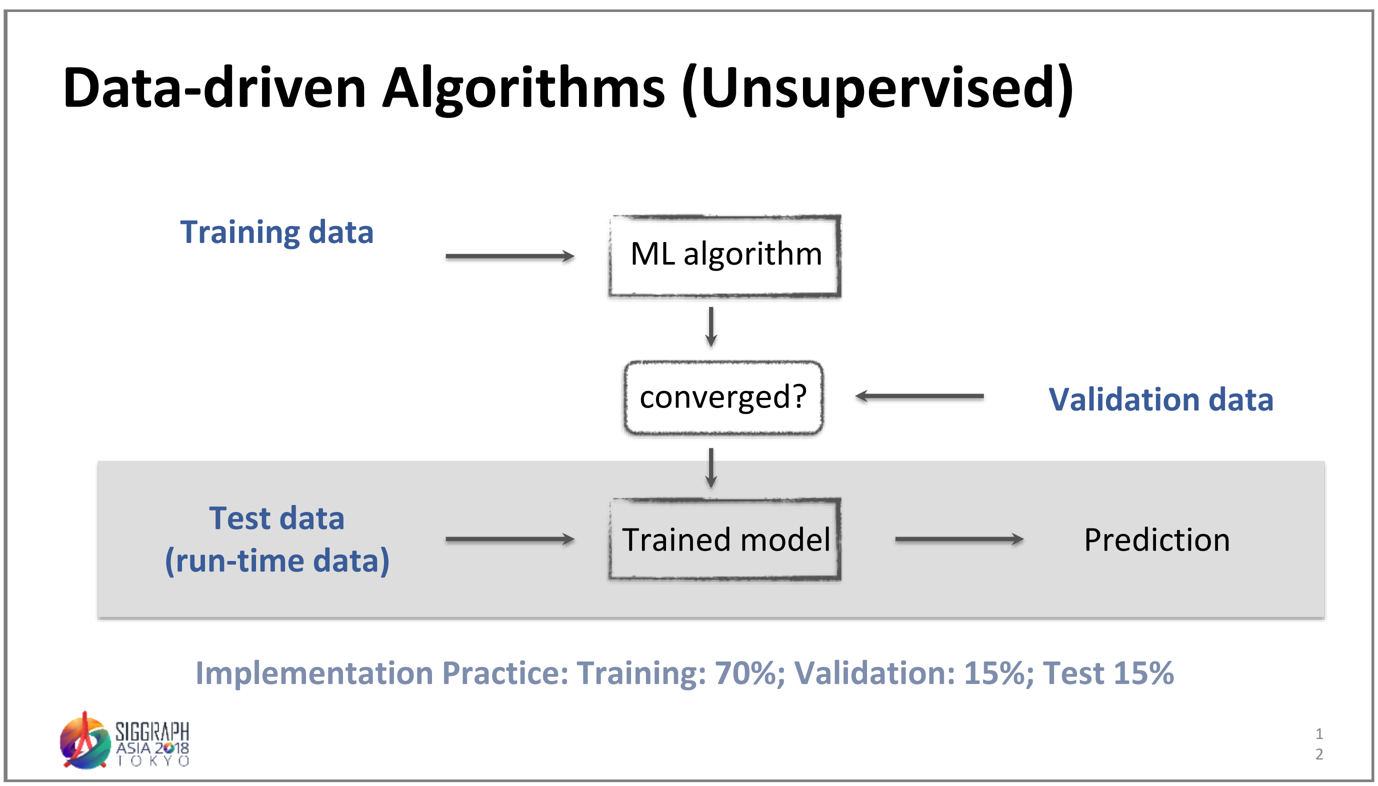
Data-driven Algorithms (Supervised)



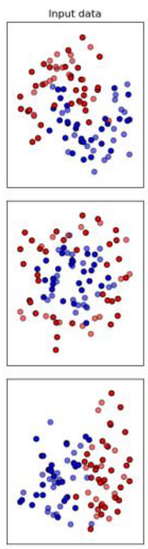
SIGGRAPH ASIA 2018 TOKYO

10


Data-driven Algorithms (Supervised)
Labelled data
(supervision data)
ML algorithm
converged?
Validation data
(supervision data)
Test data
(run-time data)
Trained model
Prediction
Implementation Practice: Training: 70%; Validation: 15%; Test 15%
SIGGRAPH ASIA 2018 TOKYO
11


Data-driven Algorithms (Unsupervised)
Training data
ML algorithm
converged?
Validation data
Test data
(run-time data)
Trained model
Prediction
Implementation Practice: Training: 70%; Validation: 15%; Test 15%
SIGGRAPH ASIA 2018 TOKYO
12

Various ML Approaches (Supervised approaches)



http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

SIGGRAPH ASIA 2018 TOKYO

13

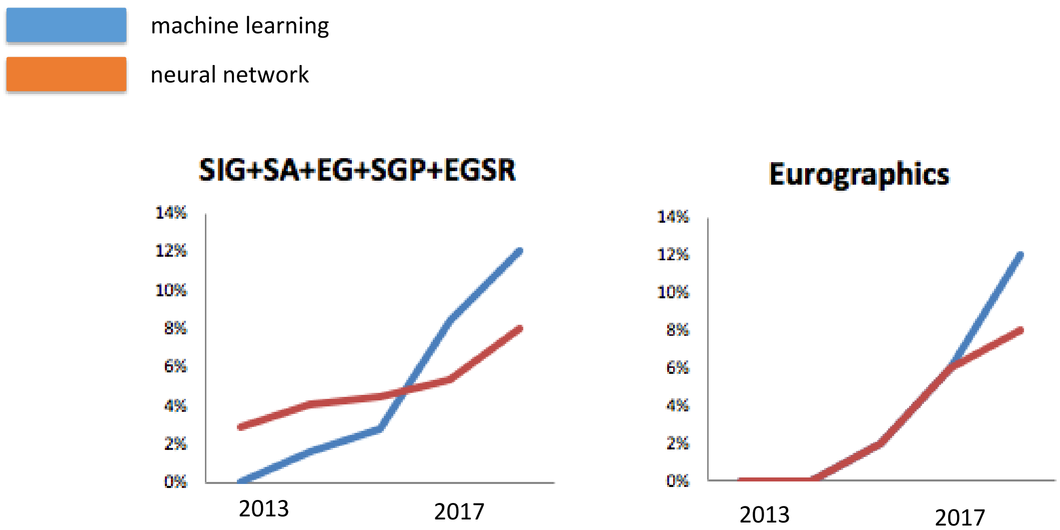
Rise of Learning

- 1958: Perceptron
- 1974: Backpropagation
- 1981: Hubel & Wiesel wins Nobel prize for ‘visual system’
- 1990s: SVM era
- 1998: CNN used for handwriting analysis
- **2012: AlexNet wins ImageNet**

SIGGRAPH ASIA 2018 TOKYO

14

Rise of Machine Learning (in Graphics)



SIGGRAPH ASIA 2018 TOKYO

15

What is Special about Graphics?

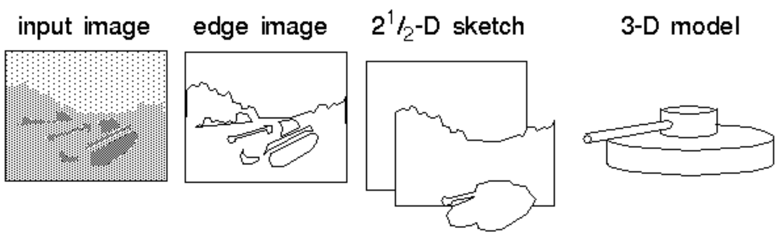
- **Image Processing** (image translation tasks)
- Many sources of input data — **model building** (e.g., images, scanners, motion capture)
- Many sources of **synthetic data** — can serve as supervision data (e.g., rendering, animation)
- Many problems in **generative models**

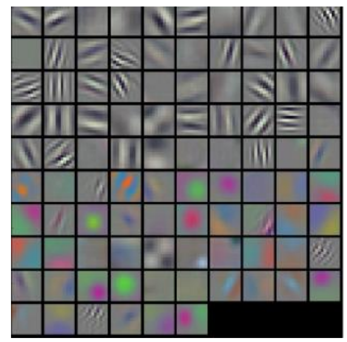
SIGGRAPH ASIA 2018 TOKYO

1
6

End-to-end: **Features**

- *Old days*
  - First some handy features were extracted, e.g. edges or corners (hand-crafted)
  - Second, some AI was ran on that features (optimized)
- *Now*
  - End-to-end
  - Move away from hand-crafted representations





17

End-to-end: **Loss**

- *Old days*
  - Evaluation came after
  - It was a bit optional:
    - You might still have a good algorithm without a good way of quantifying it
    - Evaluation helped publishing
- *Now*
  - It is essential and build-in
  - If the loss is not good, the result is not good
  - Evaluation happens automatically
- While still much is left to do, this makes graphics much more reproducable

18

End-to-end: **Data**

- *Old days*
  - Test with some toy examples
  - Deploy on real stuff
  - Maybe collect some data later
- *Now*
  - Test and deploy need to be as identical as you can
  - Need to collect data first
  - No two steps

SIGGRAPH ASIA 2018 TOKYO

19

Examples in Graphics

Geometry

Image manipulation

Animation

Rendering

SIGGRAPH ASIA 2018 TOKYO

20

Examples in Graphics

Geometry

Procedural modelling

Mesh segmentation

Learning deformations

Colorization

Sketch simplification

Image manipulation

Animation

Boxification

BRDF estimation

Fluid

Real-time rendering

Animation

Rendering

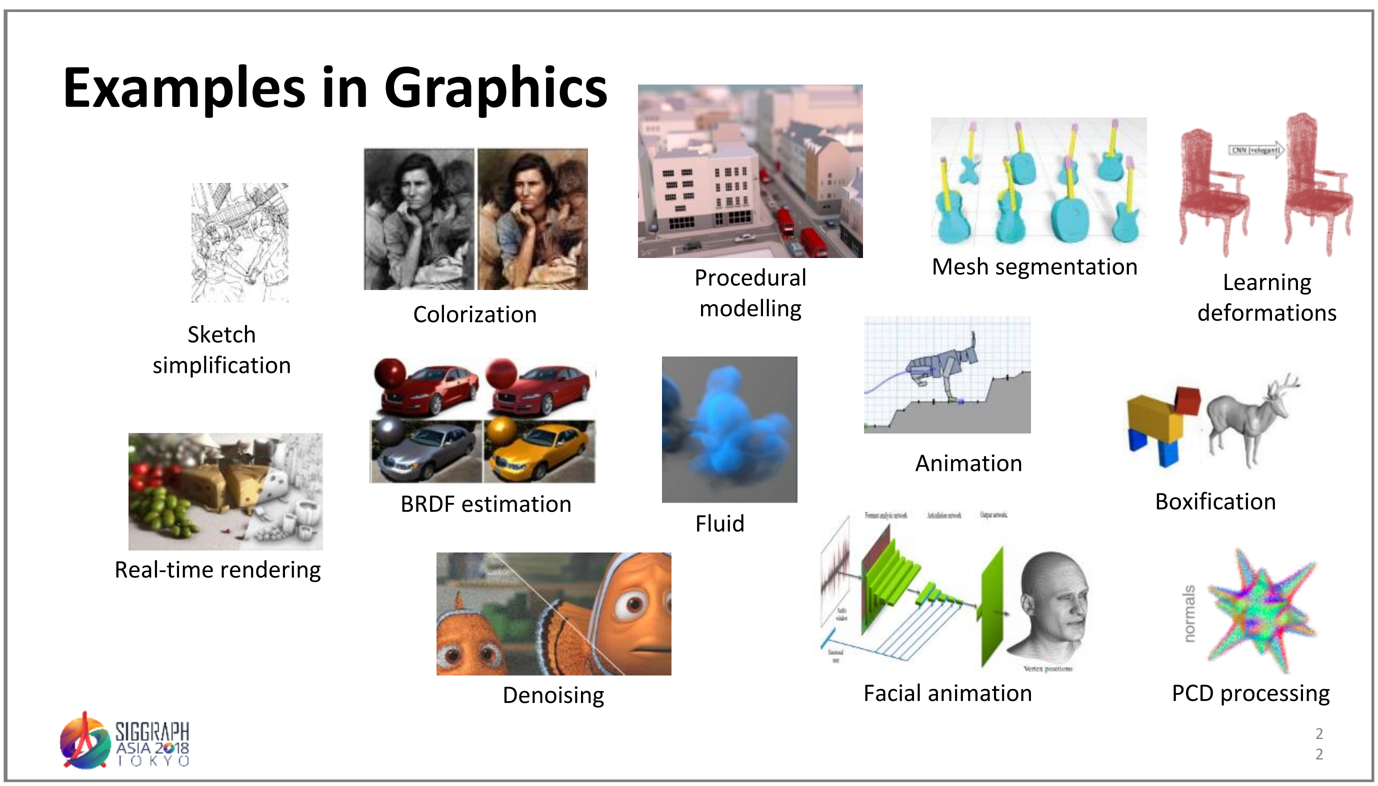
Denoising

Facial animation

PCD processing

SIGGRAPH ASIA 2018 TOKYO

21



Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics**

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

Theory

Niloy Mitra UCL
Iasonas Kokkinos UCL/Facebook
Paul Guerrero UCL
Nils Thuerey TU Munich
Tobias Ritschel UCL

UCL
facebook Artificial Intelligence Research
TUM Technische Universität München

Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

SIGGRAPH Asia Course CreativeAI: Deep Learning for Graphics 2

Machine Learning

Machine learning is a field of **computer science** that uses statistical techniques to give computer systems the ability to ***learn*** (i.e., progressively improve performance on a specific task) with **data**, without being explicitly programmed.

'ML' coined by Arthur Samuel, 1959.

data → **model building** → **prediction**

SIGGRAPH ASIA 2018 TOKYO

3

Machine Learning Variants

- **Supervised**
  - Classification
  - Regression
  - Data consolidation
- **Unsupervised**
  - Clustering
  - Dimensionality Reduction
- **Weakly supervised/semi-supervised**
  Some data supervised, some unsupervised
- **Reinforcement learning**
  Supervision: sparse reward for a sequence of decisions

SIGGRAPH ASIA 2018 TOKYO

4

Machine Learning Variants

- **Supervised**
  - **Classification**
  - Regression
  - Data consolidation
- **Unsupervised**
  - Clustering
  - Dimensionality Reduction
- **Weakly supervised/semi-supervised**
  Some data supervised, some unsupervised
- **Reinforcement learning**
  Supervision: sparse reward for a sequence of decisions

SIGGRAPH ASIA 2018 TOKYO

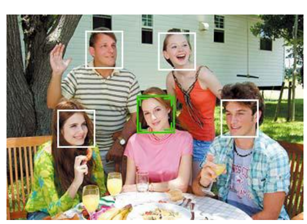
5

Classification Examples

- Digit Recognition

- Spam Detection

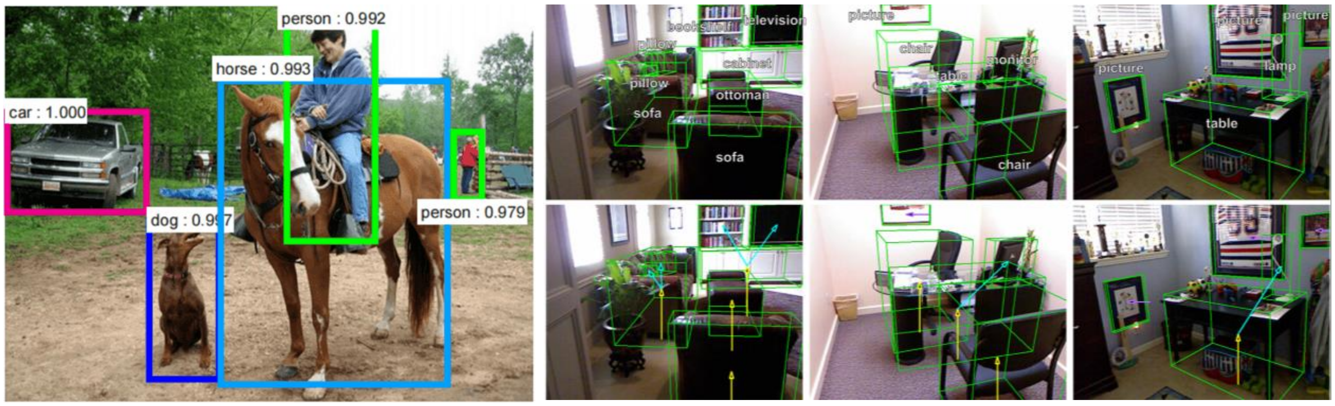
- Face detection



SIGGRAPH ASIA 2018 TOKYO

6

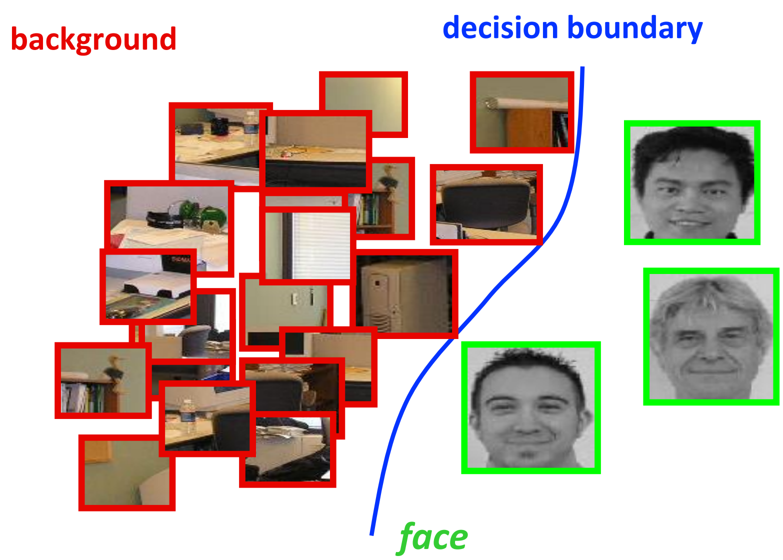
Segmentation + Classification in Real Images



Evaluation measures: Confusion matrix, ROC curve, precision, recall, etc.

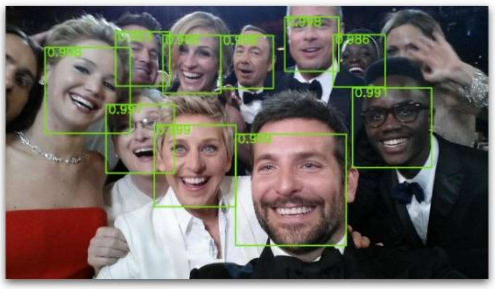
7

`Faceness' Function: Classifier



8

Face Detection



CMU Science Lab

SIGGRAPH ASIA 2018 TOKYO

9

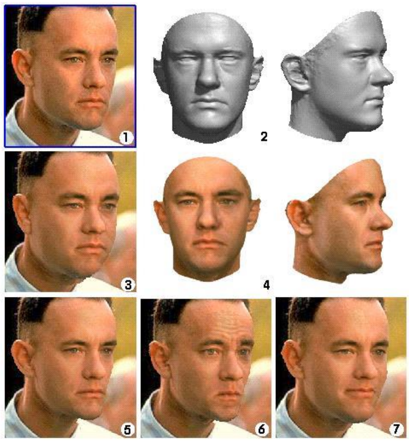
Machine Learning Variants

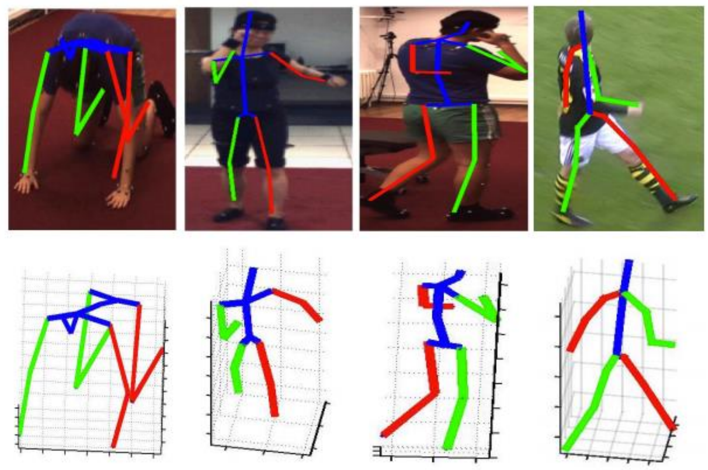
- **Supervised**
  - Classification
  - **Regression**
  - Data consolidation
- **Unsupervised**
  - Clustering
  - Dimensionality Reduction
- **Weakly supervised/semi-supervised**
  Some data supervised, some unsupervised
- **Reinforcement learning**
  Supervision: sparse reward for a sequence of decisions

SIGGRAPH ASIA 2018 TOKYO

10

Human Face/Pose Estimation



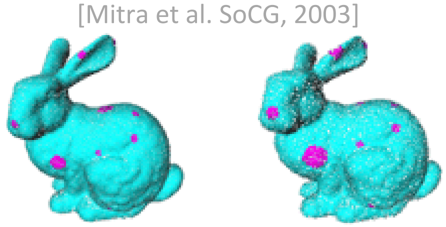


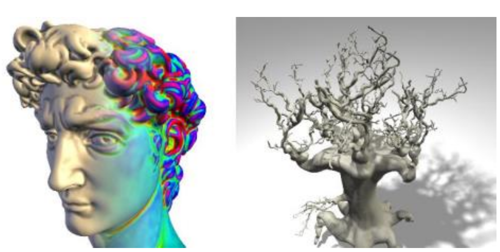
[Blanz and Vetter, Siggraph, 1999]

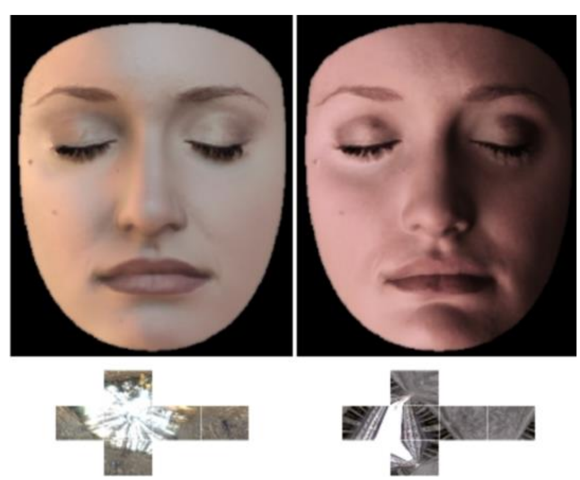
11

Regression: Model Estimation

[Mitra et al. SoCG, 2003]







[Guennebaud et al., Siggraph, 2007]

[Zwicker et al., EGSR, 2005]

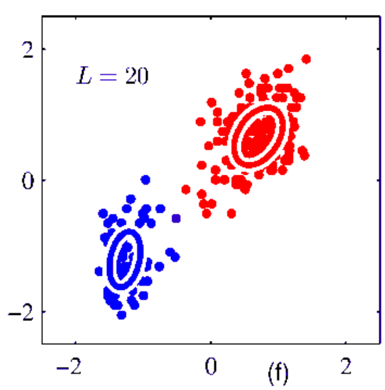
12

Machine Learning Variants

- **Supervised**
  - Classification
  - Regression
  - Data consolidation
- **Unsupervised**
  - **Clustering**
  - Dimensionality Reduction
- **Weakly supervised/semi-supervised**
  Some data supervised, some unsupervised
- **Reinforcement learning**
  Supervision: sparse reward for a sequence of decisions

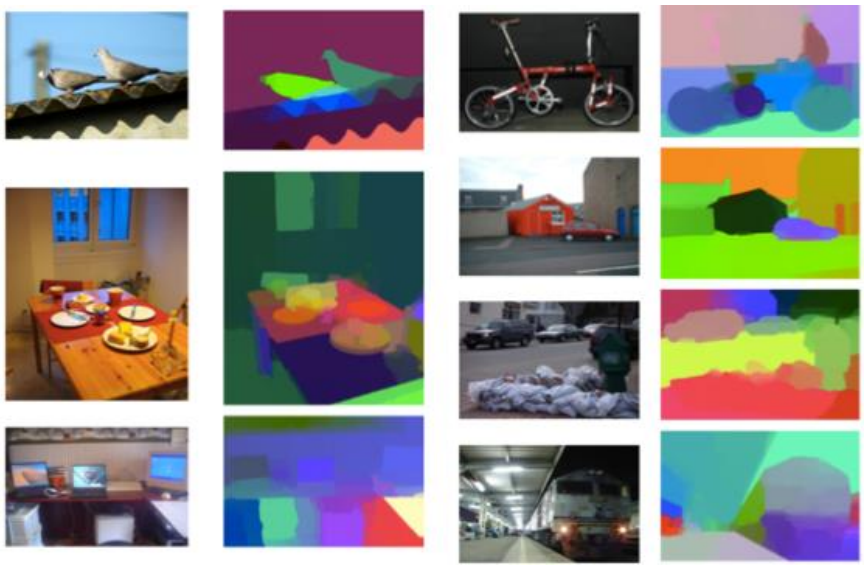
SIGGRAPH ASIA 2018 TOKYO 13

Clustering: Color Points According to X



SIGGRAPH ASIA 2018 TOKYO 14

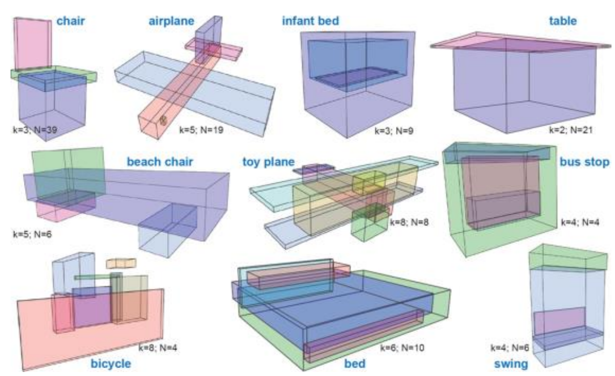
Clustering Examples: Image Segmentation using NCuts



15

Clustering Examples

[Chu et al., TVCG, 2009]



[Zheng et al., Eurographics, 2014]

16

Machine Learning Variants

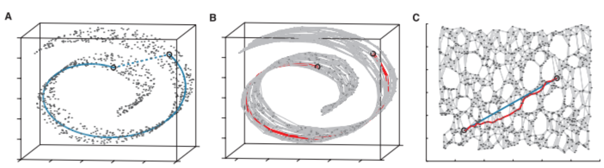
- **Supervised**
  - Classification
  - Regression
  - Data consolidation
- **Unsupervised**
  - Clustering
  - **Dimensionality Reduction**
- **Weakly supervised/semi-supervised**
  Some data supervised, some unsupervised
- **Reinforcement learning**
  Supervision: sparse reward for a sequence of decisions

SIGGRAPH ASIA 2018 TOKYO

17

Dimensionality Reduction (Manifold Learning)

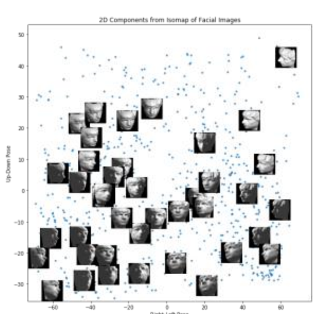
Isomap

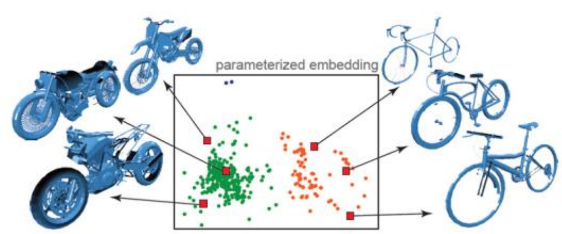


[Tenenbaum et al., Science, 2000]

[Yang et al., TOG, 2011]

Face Manifold





[Averkiou et al., Eurographics, 2014]

SIGGRAPH ASIA 2018 TOKYO

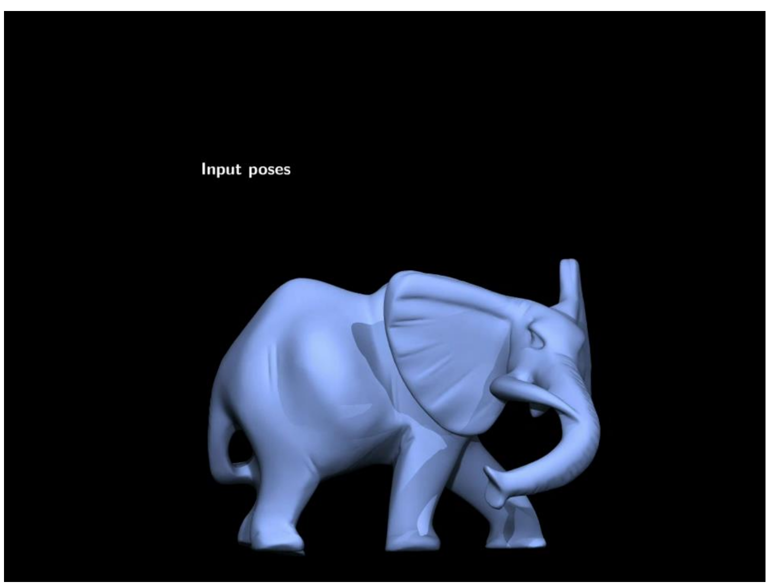
18

Example of Nonlinear Manifold: Faces

$\mathbf{x}_1$ $\frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2)$ $\mathbf{x}_2$

19

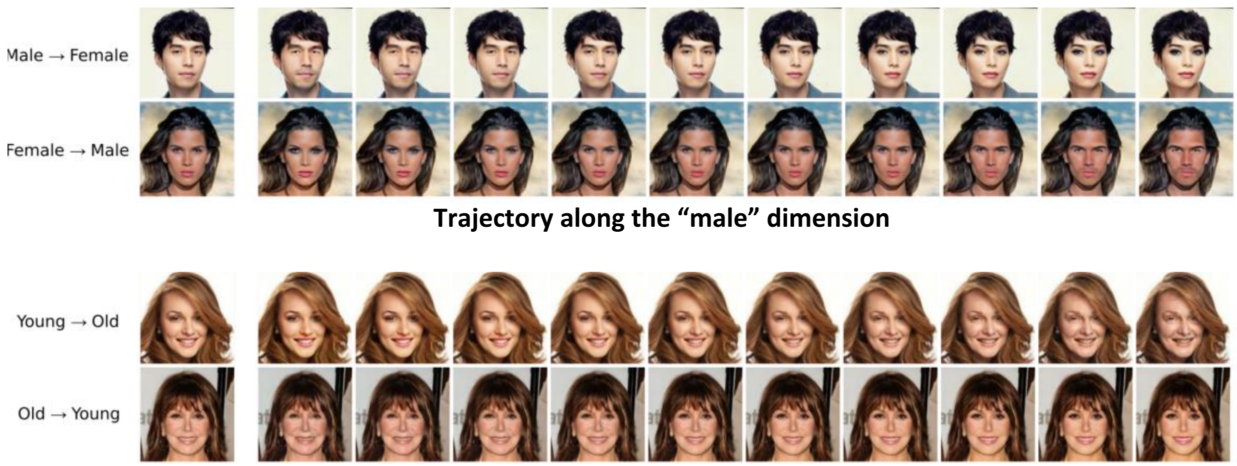
Morphing (Interpolation in Shape Space)



[Kilian et al., Siggraph, 2007]

20

Moving Along Learned Face Manifold



Trajectory along the "male" dimension

Trajectory along the "young" dimension

[Lample et. al. Fader Networks, NIPS 2017]

SIGGRAPH ASIA 2018 TOKYO

21

Notations: Vectors and Matrices

- linear **independence**; **rank** of a matrix
- **span** of a matrix

vector	$\mathbf{x}$	
matrix	$\mathbf{A}_{m \times n} = [\mathbf{a}_1 \ldots \mathbf{a}_n]$	
linear equation	$\mathbf{Ax} = \mathbf{b}$	
inner prod.	$< \mathbf{x}, \mathbf{y} >= \mathbf{x}^T \mathbf{y}$	$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$
		$\mathbf{x}^T \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\theta)$

SIGGRAPH ASIA 2018 TOKYO

22

Notations: Vectors and Matrices (cont.)

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \dots)^{1/p}$$

$$L_1, L_2, L_p, L_\infty$$

$$\|\mathbf{x}\|_p = \max\{|x_1|, |x_2|, \dots\} \qquad p = \infty$$

range $\mathcal{R}(\mathbf{A}) = \{\mathbf{Ax} : \mathbf{x} \in \mathbb{R}^n\}$

null space $\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = 0\}$

SIGGRAPH ASIA 2018 TOKYO 23

Eigenvectors and Eigenvalues

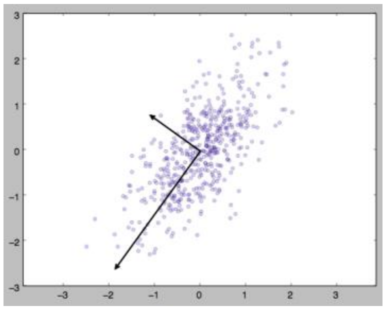
$$\mathbf{y} = \mathbf{Ax} \qquad \mathbf{T} = [\mathbf{v}_1\ \mathbf{v}_2 \dots]$$

$$\mathbf{Ae}_i = \lambda_i \mathbf{e}_i \qquad \mathbf{T}^{-1} A \mathbf{T} = \mathrm{diag}(\lambda_1, \lambda_2, \dots)$$

- All eigenvalues of symmetric matrices are real.
- Any real symmetric nxn matrix has a set of **n** mutually orthogonal eigenvectors.

SIGGRAPH ASIA 2018 TOKYO 24

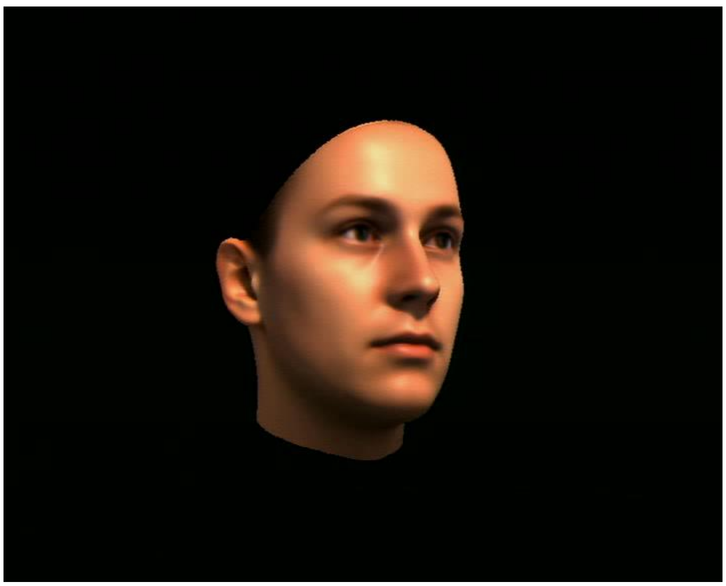
Code Example



```
rng = np.random.RandomState(10)
X = np.dot(rng.rand(2, 2), rng.randn(2, 500)).T

mean_vec = np.mean(X, axis=0)
cov_mat = (X - mean_vec).T.dot((X - mean_vec)) / (X.shape[0]-1)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
```

25

Morphable Faces



26

Singular Value Decomposition (SVD)

- Very useful for matrix manipulation.
- Used for robust numerical computation.

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

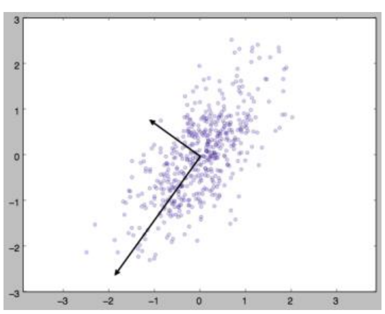
scaling rotation

$$\mathbf{A} = \mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$$

SIGGRAPH ASIA 2018 TOKYO

27

Code Example




```
mean_vec = np.mean(X, axis=0)
cov_mat = (X - mean_vec).T.dot((X - mean_vec)) / (X.shape[0]-1)
matU, sigma, matV = np.linalg.svd(cov_mat)
```

SIGGRAPH ASIA 2018 TOKYO

28

Differentiation (chain rule recap)

$$z = f \circ g(x) = f(g(x))$$

$$z = f(y)$$

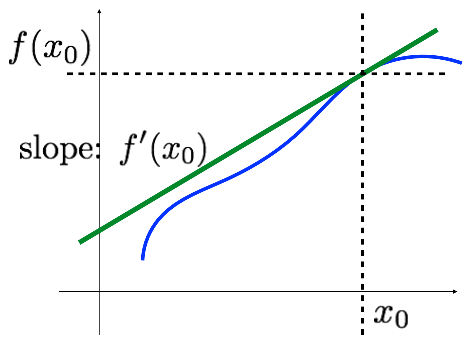
$$y = g(x)$$

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} = f'(y)g'(x) = f'(g(x))g'(x)$$

$$z = \sin(5x)$$

$$= \frac{d\sin(5x)}{d(5x)} \frac{d(5x)}{dx}$$

$$= 5\cos(5x)$$



29

Derivative Matrix

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

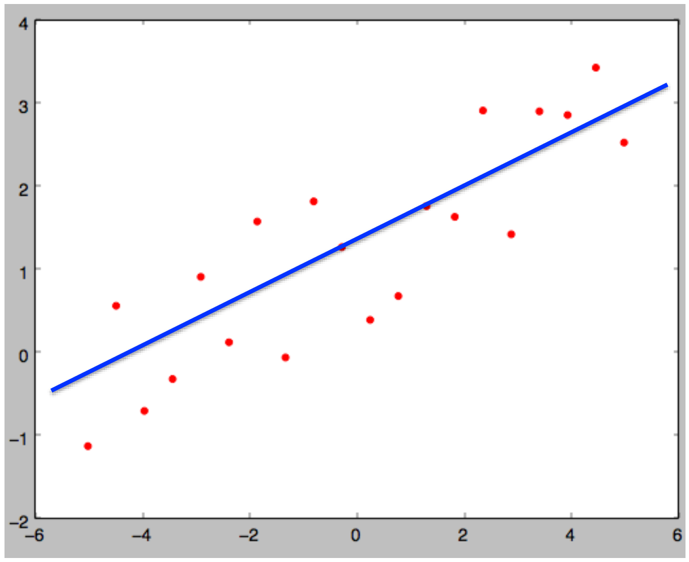
$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \qquad \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_j} = \begin{bmatrix} \frac{\partial \mathbf{f}_1(\mathbf{x})}{\partial x_j} \\ \vdots \\ \frac{\partial \mathbf{f}_m(\mathbf{x})}{\partial x_j} \end{bmatrix}_{m \times 1}$$

Jacobian matrix

$$\mathbf{L} = D\mathbf{f} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}_{m \times n}$$

30

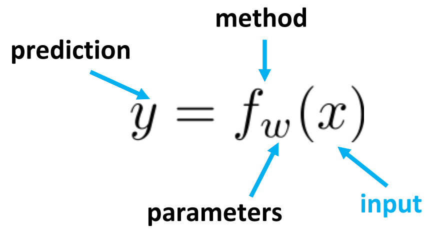
Regression: Continuous Output



Learning a Function

$$y = f_w(x)$$

Learning a Function



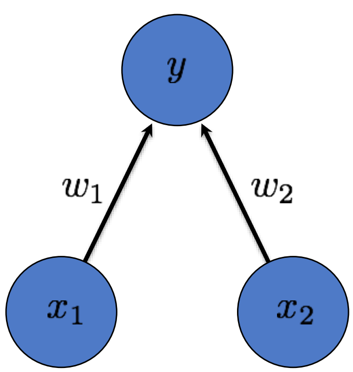
$$y = f_w(x)$$

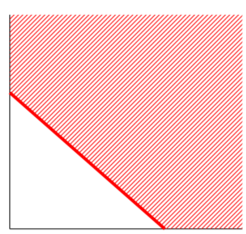
Calculus	$x \in \mathbb{R}$	**Classification:**	$y \in \{0, 1\}$
Vector calculus	$\mathbf{x} \in \mathbb{R}^d$	**Regression:**	$y \in \mathbb{R}$

Machine learning: can work also for discrete inputs, strings, images, meshes, animations, …

SIGGRAPH ASIA 2018 TOKYO 33

Learning a Simple Separator/Classifier



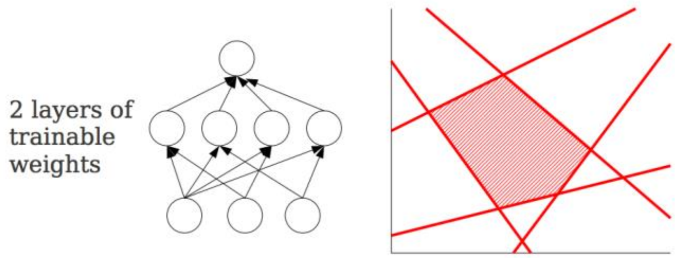


separating hyperplane

$$y = f(w_1 x_1 + w_2 x_2)$$

SIGGRAPH ASIA 2018 TOKYO 34

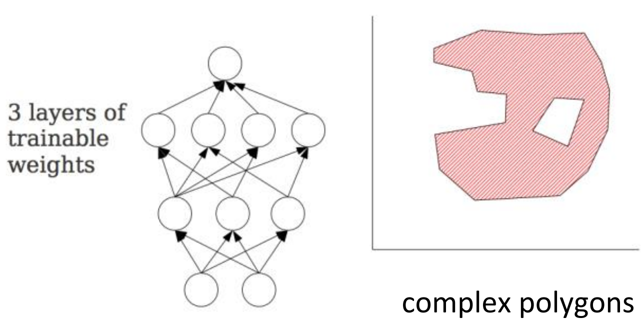
Combining Simple Functions/Classifiers



convex region

35

Combining Simple Functions/Classifiers



complex polygons

36

Learning a Function: Modeling

method
prediction

$$y = f_w(x) = f(x; w)$$

parameters
input

$$w \in \mathbb{R}$$

$$\mathbf{w} \in \mathbb{R}^K$$

37

Regression

1. Least Squares fitting
2. Nonlinear error function and gradient descent
3. Perceptron training

38

Regression

1. Least Squares fitting

2. Nonlinear error function and gradient descent

3. Perceptron training

SIGGRAPH ASIA 2018 TOKYO

39

Assumption: Linear Function

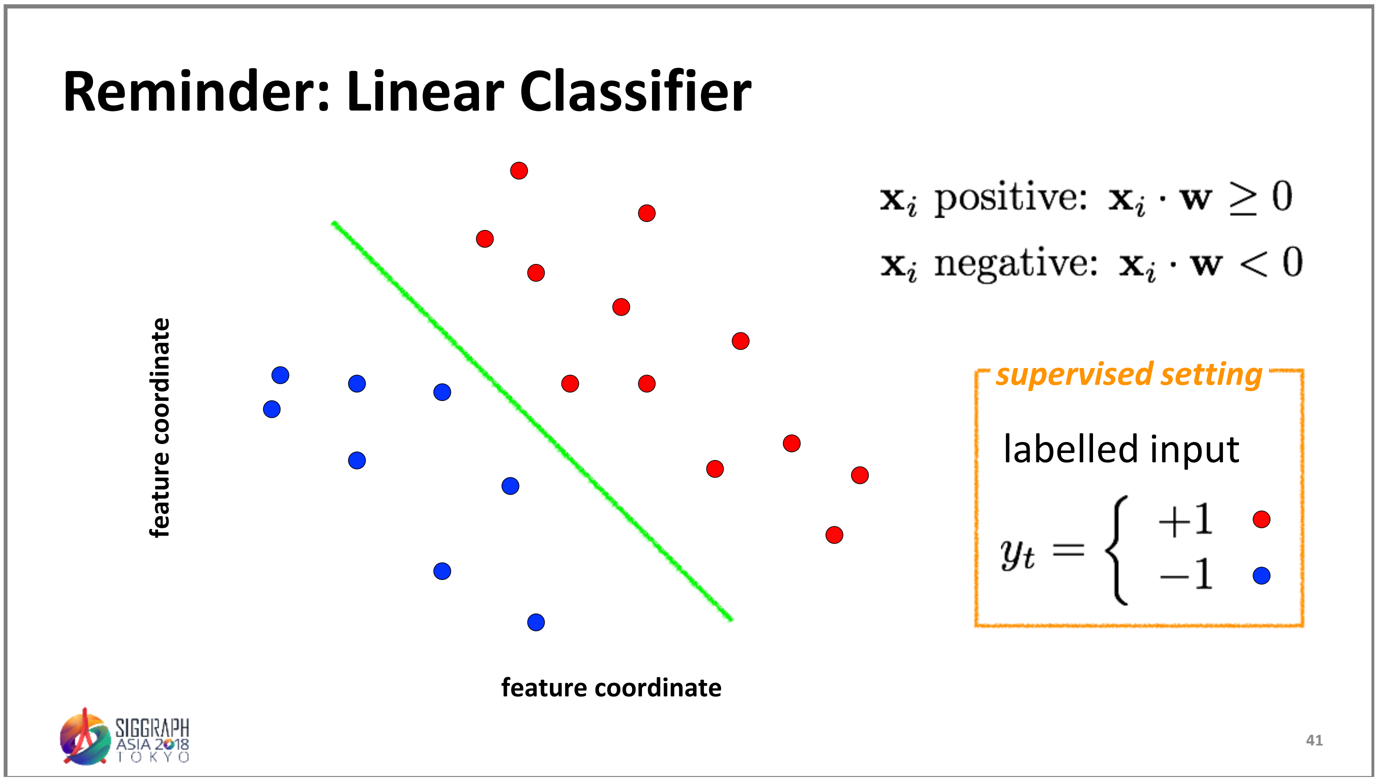
$$y = f_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

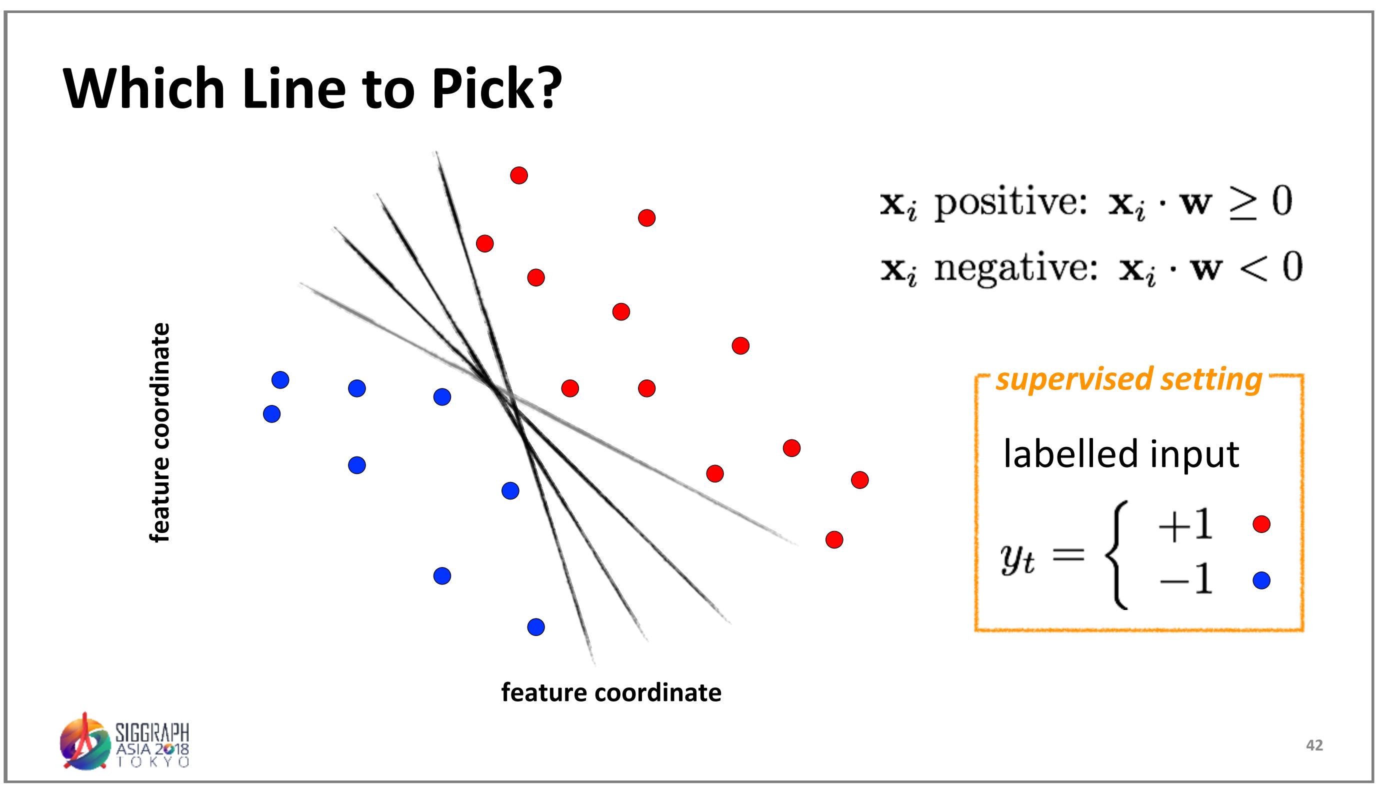
$$\mathbf{w}^T \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle = \sum_{d=1}^{D} \mathbf{w}_d \mathbf{x}_d$$

$$\mathbf{x} \in \mathbb{R}^D, \mathbf{w} \in \mathbb{R}^D$$

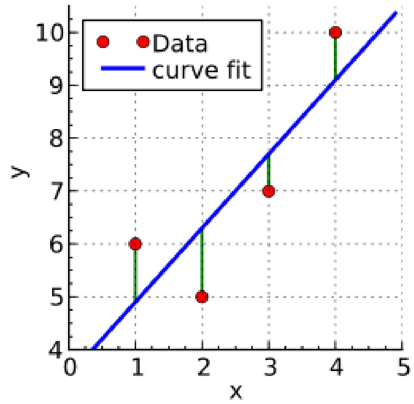
SIGGRAPH ASIA 2018 TOKYO

40


Reminder: Linear Classifier
$\mathbf{x}_i$ positive: $\mathbf{x}_i \cdot \mathbf{w} \geq 0$
$\mathbf{x}_i$ negative: $\mathbf{x}_i \cdot \mathbf{w} < 0$
supervised setting
labelled input
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
feature coordinate
feature coordinate
SIGGRAPH ASIA 2018 TOKYO
41


Which Line to Pick?
$\mathbf{x}_i$ positive: $\mathbf{x}_i \cdot \mathbf{w} \geq 0$
$\mathbf{x}_i$ negative: $\mathbf{x}_i \cdot \mathbf{w} < 0$
supervised setting
labelled input
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
feature coordinate
feature coordinate
SIGGRAPH ASIA 2018 TOKYO
42

Linear Regression in 1D



Training set: input–output pairs

$$\mathcal{S} = \{(x^i, y^i)\}, \quad i = 1 \ldots, N$$
$$x^i \in \mathbb{R}, \quad y^i \in \mathbb{R}$$

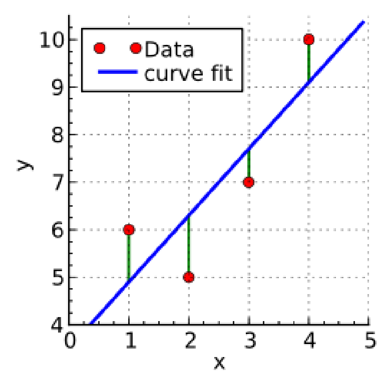
SIGGRAPH ASIA 2018 TOKYO

43

Linear regression in 1D

$$y^i = w_0 + w_1 x_1^i + \epsilon^i \qquad w_0 \text{ bias}$$
$$= w_0 x_0^i + w_1 x_1^i + \epsilon^i, \quad x_0^i = 1, \quad \forall i$$
$$= \mathbf{w}^T \mathbf{x}^i + \epsilon^i$$

noise



SIGGRAPH ASIA 2018 TOKYO

44

Sum of Square Errors *(MSE without the mean)*

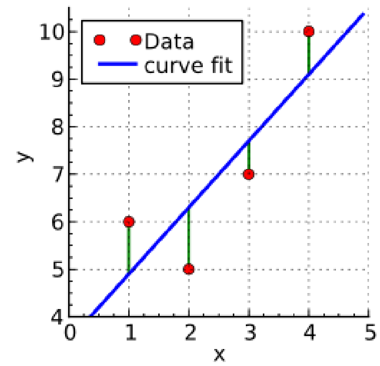
$$y^i = \mathbf{w}^T \mathbf{x}^i + \epsilon^i$$

Loss function: sum of squared errors

$$L(\mathbf{w}) = \sum_{i=1}^{N} (\epsilon^i)^2$$

In two variables:

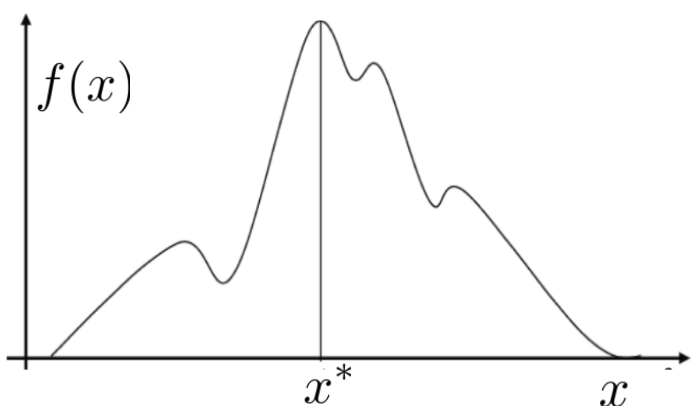
$$L(w_0, w_1) = \sum_{i=1}^{N} \left[y^i - \left(w_0 x_0^i + w_1 x_1^i \right) \right]^2$$



*Question: what is the best (or least bad) value of **w**?*

45

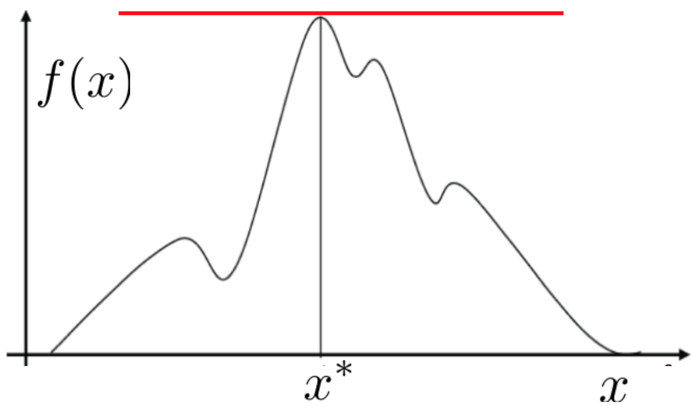
Calculus 101



$$x^* = \text{argmax}_x f(x)$$

46

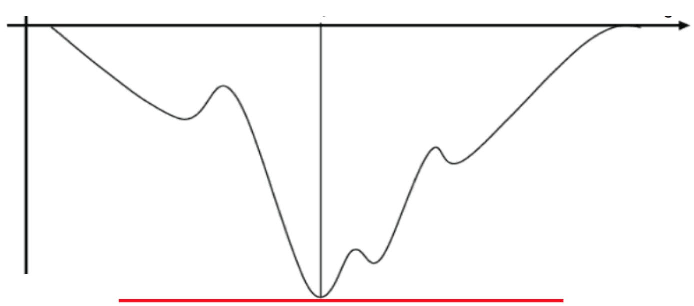
Local Extrema Condition



$$x^* = \text{argmax}_x f(x) \quad \rightarrow \quad f'(x^*) = 0$$

47

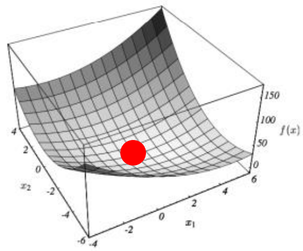
Local Extrema Condition



$$x^* = \text{argmax}_x f(x) \quad \rightarrow \quad f'(x^*) = 0$$

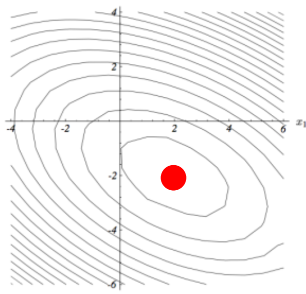
48

Vector Calculus 101



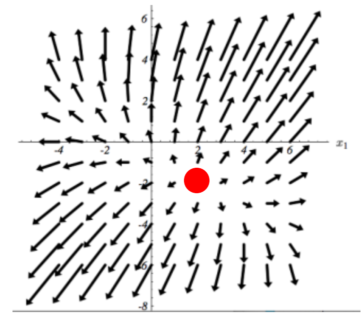
$f(\mathbf{x})$

2D function graph



$f(\mathbf{x}) = c$

isocontours



$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

gradient field

● at minimum of function: $\nabla f(\mathbf{x}) = \mathbf{0}$

SIGGRAPH ASIA 2018 TOKYO

49

Line Fitting

$$L(w_0, w_1) = \sum_{i=1}^{N} \left[y^i - \left(w_0 x_0^i + w_1 x_1^i\right)\right]^2$$

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = \sum_{i=1}^{N} \frac{\partial \left[y^i - \left(w_0 x_0^i + w_1 x_1^i\right)\right]^2}{\partial w_0}$$

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = \sum_{i=1}^{N} \frac{\partial [z^i]^2}{\partial z^i} \frac{\partial z^i}{\partial w_0} = \sum_{i=1}^{N} (2z^i)(-x_0^i)$$

$$= -2 \sum_{i=1}^{N} (y^i - (w_0 x_0^i + w_1 x_1^i)) x_0^i$$

$$z^i = y^i - (w_0 x_0^i + w_1 x_1^i)$$

SIGGRAPH ASIA 2018 TOKYO

50

Line Fitting (continued)

$$\frac{\partial L(w_0,w_1)}{\partial w_0} = \sum_{i=1}^{N} \frac{\partial \left[y^i - \left(w_0 x_0^i + w_1 x_1^i\right)\right]^2}{\partial w_0}$$
$$= -2\sum_{i=1}^{N} \left(y^i x_0^i - w_0 x_0^i x_0^i - w_1 x_1^i x_0^i\right)$$

$$\frac{\partial L(w_0,w_1)}{\partial w_0} = 0 \qquad \sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$

SIGGRAPH ASIA 2018 TOKYO

51

Line Fitting (continued)

$$\sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$
$$\sum_{i=1}^{N} y^i x_1^i = w_0 \sum_{i=1}^{N} x_0^i x_1^i + w_1 \sum_{i=1}^{N} x_1^i x_1^i$$

2x2 system of equations

$$\begin{bmatrix} \sum_{i=1}^{N} y^i x_0^i \\ \sum_{i=1}^{N} y^i x_1^i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} x_0^i x_0^i & \sum_{i=1}^{N} x_0^i x_1^i \\ \sum_{i=1}^{N} x_0^i x_1^i & \sum_{i=1}^{N} x_1^i x_1^i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

SIGGRAPH ASIA 2018 TOKYO

52

Line Fitting (continued)

$$\begin{bmatrix} \sum_{i=1}^{N} y^i x_0^i \\ \sum_{i=1}^{N} y^i x_1^i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} x_0^i x_0^i & \sum_{i=1}^{N} x_0^i x_1^i \\ \sum_{i=1}^{N} x_0^i x_1^i & \sum_{i=1}^{N} x_1^i x_1^i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_0^1 & x_1^1 \\ \vdots & \vdots \\ x_0^N & x_2^N \end{bmatrix}$$

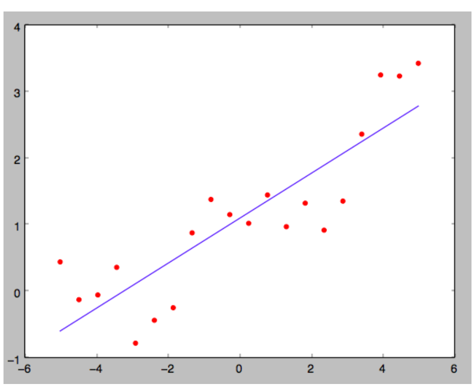
Normal Equation

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

SIGGRAPH ASIA 2018 TOKYO

53

Code Example



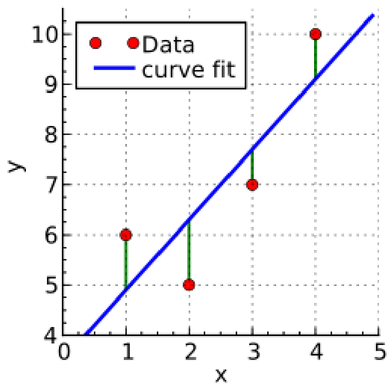

```
import numpy as np
from numpy import array
from numpy import matmul
from numpy.linalg import inv
from numpy.random import rand
from matplotlib import pyplot

# generate data on a line perturbed with some noise
noise_margin= 2
w = rand(2,1) # w[0] is random constant term (offset from origin), w[1] is random linear term (slope)
x  = np.linspace(-5,5,20)
y = w[0] + w[1]*x + noise_margin*rand(len(x))

# create the design matrix: the x data, and add a column of ones for the constant term
X = np.column_stack( [np.ones([len(x), 1]), x.reshape(-1, 1)] )

# These are the normal equations in matrix form: w = (X' X)^-1 X' y
w_est = matmul(inv(matmul(X.transpose(),X)),X.transpose()).dot(y)

# For ridge regression, use regularizer
#weight = 0.01
#w_est = matmul(inv(matmul(X.transpose(),X) + weight*np.identity(2)),X.transpose()).dot(y)


# evaluate the x values in the fitted model to get estimated y values
y_est = w_est[0] + w_est[1]*x

# visualize the fitted model
pyplot.scatter(x, y, color='red')
pyplot.plot(x, y_est, color='blue')
pyplot.show()
```

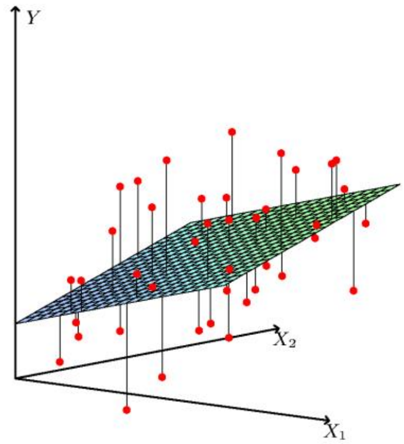
$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

SIGGRAPH ASIA 2018 TOKYO

54

Linear Regression (Line/Plane Fitting)





SIGGRAPH ASIA 2018 TOKYO

55

LS Solution for Regression

$$L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$$

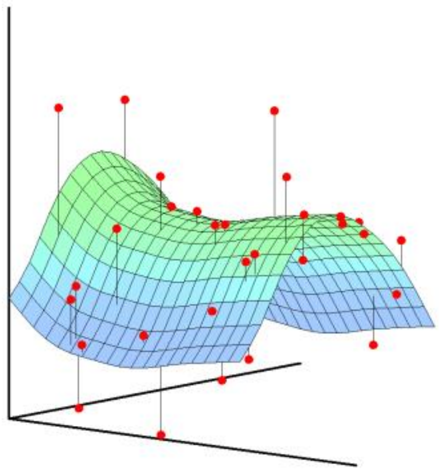
$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \ldots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \qquad \mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

SIGGRAPH ASIA 2018 TOKYO

56

Generalized Linear Regression



$$\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}$$

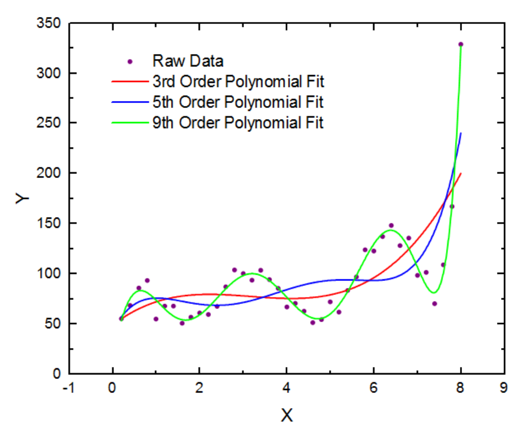
known nonlinearity

SIGGRAPH ASIA 2018 TOKYO

57

1D Example: k-th Degree Polynomial Fitting

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x \\ \vdots \\ (x)^K \end{bmatrix}$$



$$\langle \mathbf{w}, \boldsymbol{\phi}(x) \rangle = w_0 + w_1 x + \ldots + w_k (x)^K$$

SIGGRAPH ASIA 2018 TOKYO

58

Generalized Linear Regression

$$L(\mathbf{w}) = \sum_{i=1}^{N}(y^i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}^i))^T = \sum_{i=1}^{N}(\epsilon^i)^2$$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}^1)^T \\ \hline \boldsymbol{\phi}(\mathbf{x}^2)^T \\ \hline \vdots \\ \hline \boldsymbol{\phi}(\mathbf{x}^N)^T \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Nx1 **NxM** **Mx1** **Nx1**

$$\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^M$$

SIGGRAPH ASIA 2018 TOKYO

59

LS Solution for Linear Regression

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$$

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^1)^T \\ \hline (\mathbf{x}^2)^T \\ \hline \vdots \\ \hline (\mathbf{x}^N)^T \end{bmatrix}$$

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

SIGGRAPH ASIA 2018 TOKYO

60

LS Solution for Generalized Linear Regression

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w} + \boldsymbol{\epsilon}$$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

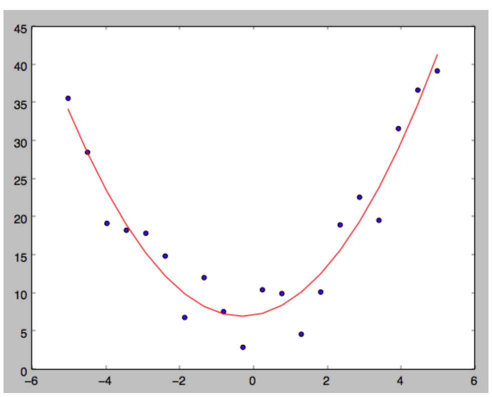
$$\mathbf{\Phi} = \begin{bmatrix} \phi(\mathbf{x}^1)^T \\ \hline \phi(\mathbf{x}^2)^T \\ \hline \vdots \\ \hline \phi(\mathbf{x}^N)^T \end{bmatrix}$$

$$\mathbf{w}^* = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi} \mathbf{y}$$

SIGGRAPH ASIA 2018 TOKYO

61

Code Example




```
import numpy as np
from numpy import array
from numpy import matmul
from numpy.linalg import inv
from numpy.random import rand
from matplotlib import pyplot

# generate data on a line perturbed with some noise
noise_margin= 3
w = 2*rand(3,1) # w[0] is random constant term (offset from origin), w[1] is random linear term, w[2] is random quadratic term
x  = np.linspace(-5,5,20)
y = w[0] + w[1]*x + w[2]*x**2 + noise_margin*rand(len(x))

# create the design matrix: the x data, and add a column of ones for the constant term
X = np.column_stack( [np.ones([len(x), 1]), x.reshape(-1, 1), (x**2).reshape(-1, 1)] )

# These are the normal equations in matrix form: w = (X' X)^-1 X' y
w_est = matmul(inv(matmul(X.transpose(),X)),X.transpose()).dot(y)


# evaluate the x values in the fitted model to get estimated y values
y_est = w_est[0] + w_est[1]*x + w_est[2]*x**2

# visualize the fitted model
pyplot.scatter(x, y)
pyplot.plot(x, y_est, color='red')
pyplot.show()
```

$$\mathbf{w}^* = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi} \mathbf{y}$$

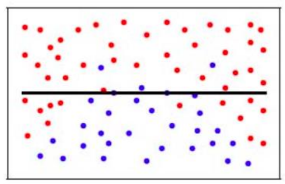
SIGGRAPH ASIA 2018 TOKYO

62

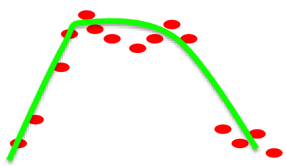
Underfitting vs. Overfitting

Underfitting

classification



regression



SIGGRAPH ASIA 2018 TOKYO

63

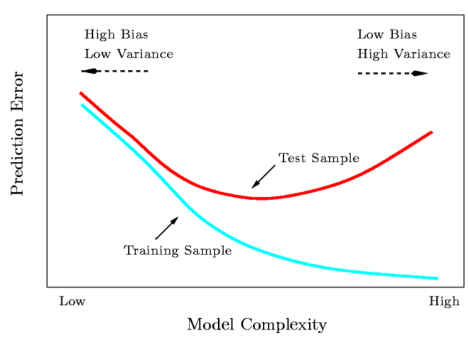
Tuning Model's Complexity

A ***flexible model*** approximates the target function well in the training set

but can "**overtrain**" and have poor performance on the test set ("variance").

A ***rigid model***'s performance is more predictable in the test set

but the model may not be good even on the training set ("bias").



SIGGRAPH ASIA 2018 TOKYO

64

Regularized Linear Regression

$$\boldsymbol{\epsilon} = \mathbf{y} - \boldsymbol{\Phi}\mathbf{w}$$ residual vector

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$ linear regression: minimize model error

Complexity term: (regularizer) $R(\mathbf{w}) \doteq \|\mathbf{w}\|_2^2 = \mathbf{w}^T\mathbf{w}$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \lambda \mathbf{w}^T \mathbf{w}$$

"data fidelity" complexity

minimum remains to be determined

scalar, remains to be determined

SIGGRAPH ASIA 2018 TOKYO

66

Least Squares Solution

$$\begin{aligned} L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} \end{aligned}$$

Condition for minimum:

$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

$$-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w}^* = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

SIGGRAPH ASIA 2018 TOKYO

67

Ridge regression: L2-regularized Linear Regression

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} + \lambda \mathbf{w}^T\mathbf{w}$$

$$= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda \mathbf{w}^T\mathbf{I}\mathbf{w}$$

as before, for linear regression

identity matrix

$$= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{w}$$

Condition for minimum:

$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

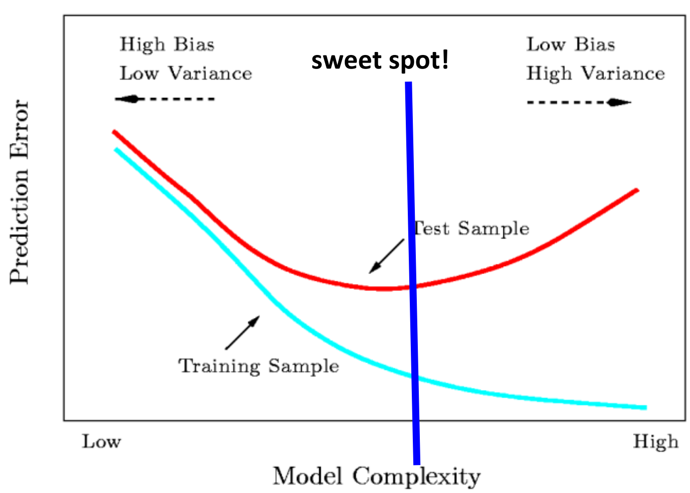
$$-2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X} + \lambda I)\mathbf{w}^* = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

λ: “hyperparameter”

68

Bias-Variance Tradeoff (function of λ)

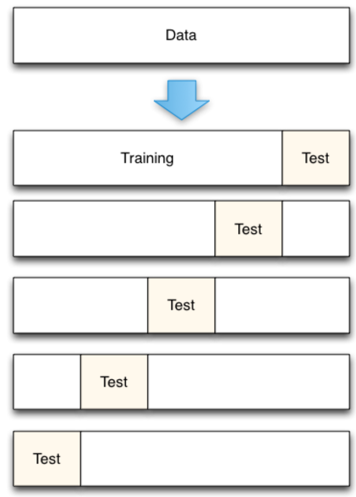


69

Selecting λ with Cross-validation

- Cross validation technique
  - Exclude part of the training data from parameter estimation
  - Use them only to predict the test error
- K-fold cross validation:
  - K splits, average K errors
- Use cross-validation for different values of λ
  - pick value that minimizes cross-validation error

Least glorious, most effective of all methods



SIGGRAPH ASIA 2018 TOKYO

70

Form of posterior distribution

Bernoulli-type conditional distribution

$$\left.\begin{aligned} P(Y=1|X=\mathbf{x};\mathbf{w}) &= \quad f(\mathbf{x},\mathbf{w}) \\ P(Y=0|X=\mathbf{x};\mathbf{w}) &= 1-f(\mathbf{x},\mathbf{w}) \end{aligned}\right\} \rightarrow$$

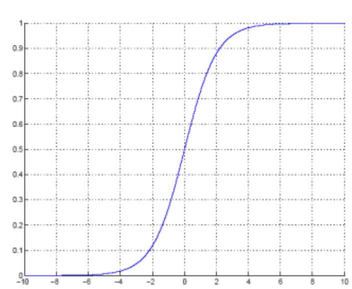
$$P(Y=y|X=\mathbf{x};\mathbf{w}) = f(\mathbf{x},\mathbf{w})^y(1-f(\mathbf{x},\mathbf{w}))^{1-y}$$

Particular choice of form of f:

$$P(Y=1|X=\mathbf{x};\mathbf{w}) = g(\mathbf{w}^T\mathbf{x})$$

Sigmoidal: $g(\alpha) = \dfrac{1}{1+\exp(-a)}$

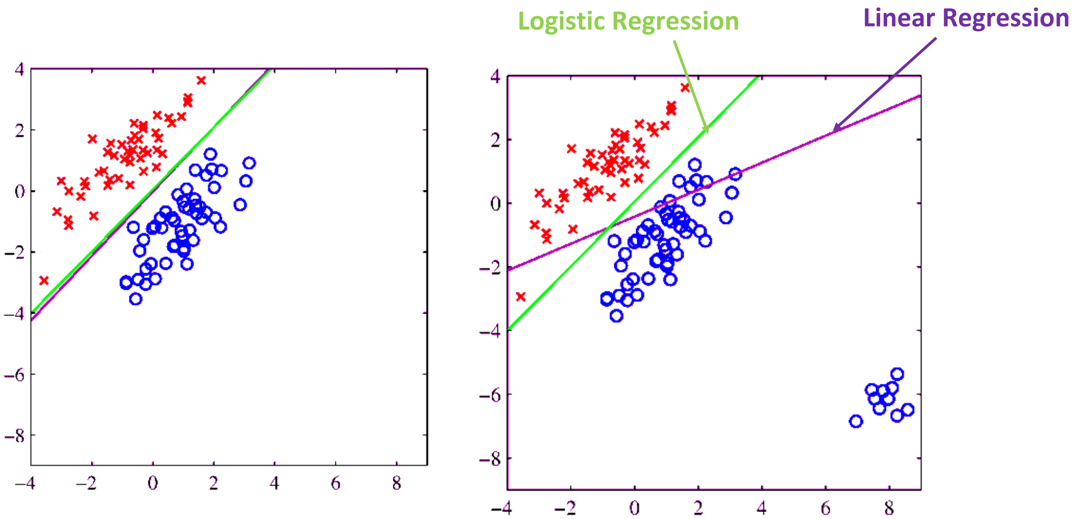
"squashing function": $-\infty \rightarrow 0$, $+\infty \rightarrow 1$



SIGGRAPH ASIA 2018 TOKYO

71

Logistic vs Linear Regression



SIGGRAPH ASIA 2018 TOKYO

72

From Two to Many

- How about multi-class classification?

SIGGRAPH ASIA 2018 TOKYO

73

Multiple Classes & Linear Regression

C classes: one-of-c coding (or one-hot encoding)

4 classes, i-th sample is in 3rd class: $\mathbf{y}^i = (0, 0, 1, 0)$

Matrix notation:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}^1 \\ \vdots \\ \mathbf{y}^N \end{bmatrix} = [\ \mathbf{y}_1 \mid \ldots \mid \mathbf{y}_C\ ] \quad \text{where} \quad \mathbf{y}_c = \begin{bmatrix} y_c^1 \\ \vdots \\ y_c^N \end{bmatrix}$$

$$\mathbf{W} = [\ \mathbf{w}_1 \mid \ldots \mid \mathbf{w}_C\ ]$$

Loss function:

$$L(\mathbf{W}) = \sum_{c=1}^{C} (\mathbf{y}_c - \mathbf{X}\mathbf{w}_c)^T (\mathbf{y}_c - \mathbf{X}\mathbf{w}_c)$$

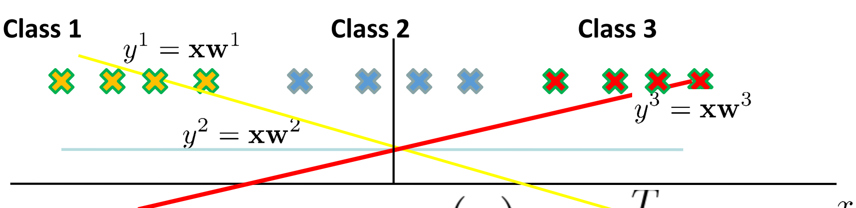
Least squares fit (decouples per class):

$$\mathbf{w}_c^* = (\mathbf{X}^T\mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}_c$$

SIGGRAPH ASIA 2018 TOKYO

74

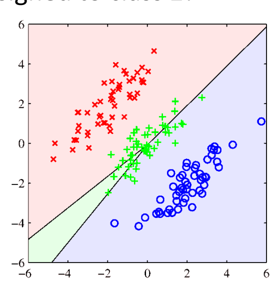
Linear Regression Masking Problem



One linear discriminant per class: $s_C(\mathbf{x}) = \mathbf{w}_C^T \mathbf{x}$

Nothing ever gets assigned to class 2!

2D version:



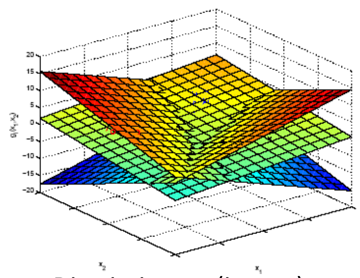
SIGGRAPH ASIA 2018 TOKYO

75

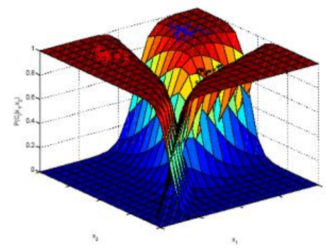
Multiple classes & Logistic regression

Soft maximum (softmax) of competing classes:

$$P(y = c|\mathbf{x}; \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T \mathbf{x})} \doteq g_c(\mathbf{x}, \mathbf{W})$$



Discriminants (inputs)

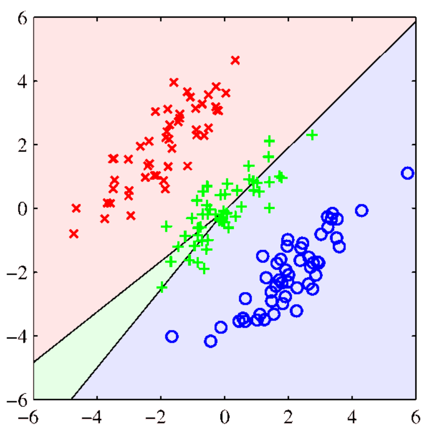


Softmax (outputs)

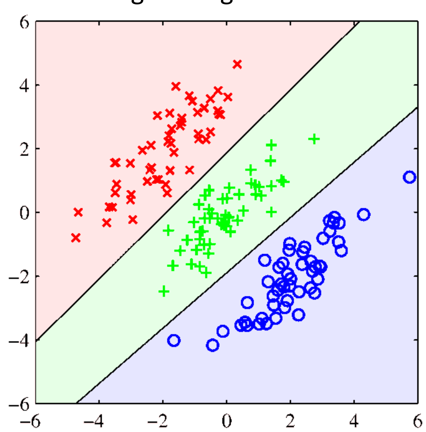
76

Logistic vs Linear Regression, n>2 classes

Linear regression



Logistic regression



Logistic regression does not exhibit the masking problem

77

LS Solution (in vector form)

$$\begin{aligned} L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned}$$

Condition for minimum:

$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

$$-2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w}^* = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

SIGGRAPH ASIA 2018 TOKYO

78

Gradient of Cross-entropy Loss

$$L(\mathbf{w}) = -\sum_{i=1}^{N} y^i \log g(\mathbf{w}^T \mathbf{x}^i) + (1 - y^i) \log(1 - g(\mathbf{w}^T \mathbf{x}^i))$$

$$\frac{\partial L(\mathbf{w})}{\partial w_k} = -\sum_{i=1}^{N} \left[y^i \frac{1}{g(\mathbf{w}^T \mathbf{x}^i)} \frac{\partial g(\mathbf{w}^T \mathbf{x}^i)}{\partial w_k} + (1 - y^i) \frac{1}{1 - g(\mathbf{w}^T \mathbf{x}^i)} (-\frac{\partial g(\mathbf{w}^T \mathbf{x}^i)}{\partial w_k}) \right]$$

using

$$g(x) = \frac{1}{1 + \exp(-x)} \rightarrow \frac{dg}{dx} = g(x)(1 - g(x))$$

$$\begin{aligned} &= -\sum_{i=1}^{N} \left[y^i \frac{1}{g(\mathbf{w}^T \mathbf{x}^i)} - (1 - y^i) \frac{1}{1 - g(\mathbf{w}^T \mathbf{x}^i)} \right] g(\mathbf{w}^T \mathbf{x}^i)(1 - g(\mathbf{w}^T \mathbf{x}^i)) \frac{\partial \mathbf{w}^T \mathbf{x}^i}{\partial w_k} \\ &= -\sum_{i=1}^{N} \left[y^i (1 - g(\mathbf{w}^T \mathbf{x}^i)) - (1 - y^i) g(\mathbf{w}^T \mathbf{x}^i) \right] x_k^i \\ &= -\sum_{i=1}^{N} \left[y^i - g(\mathbf{w}^T \mathbf{x}^i) \right] \mathbf{x}_k^i \end{aligned}$$

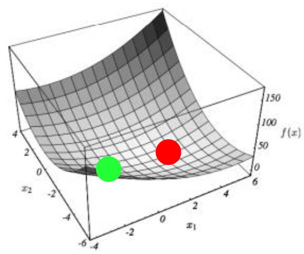
$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

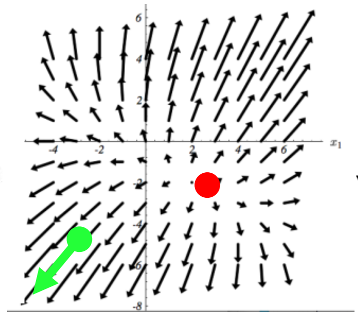
nonlinear system of equations!!

SIGGRAPH ASIA 2018 TOKYO

79

Gradient Descent Minimization



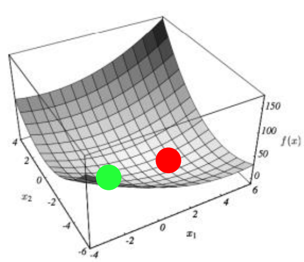


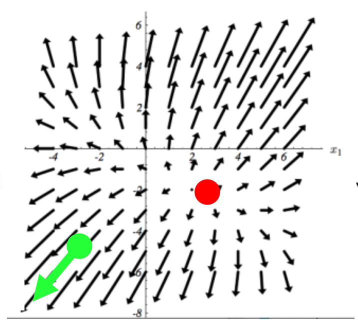
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

80

Gradient Descent Minimization





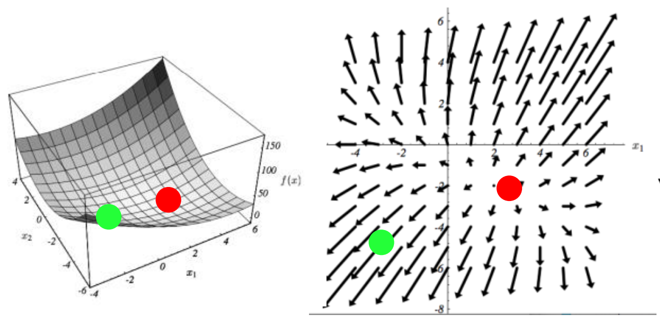
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

81

Gradient Descent Minimization



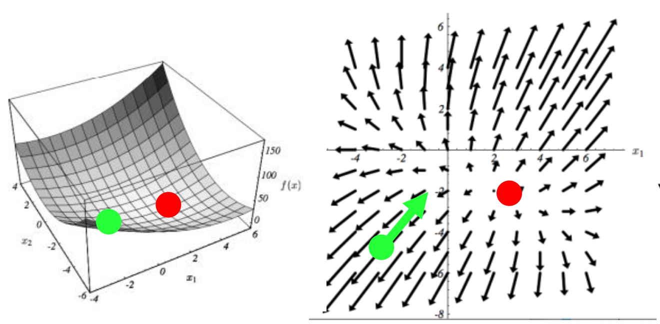
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

SIGGRAPH ASIA 2018 TOKYO 82

Gradient Descent Minimization



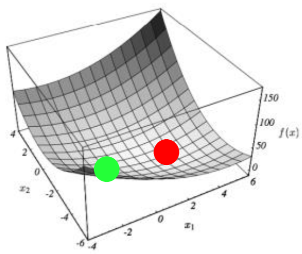
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

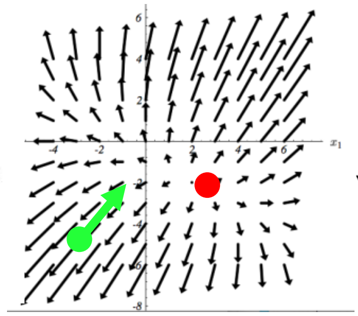
Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

SIGGRAPH ASIA 2018 TOKYO 83

Gradient Descent Minimization





$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

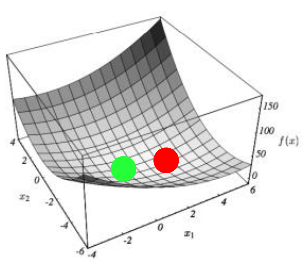
Initialize: $\mathbf{x}_0$

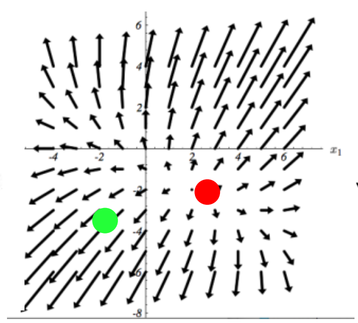
Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$ **i=0**

SIGGRAPH ASIA 2018 TOKYO

84

Gradient Descent Minimization





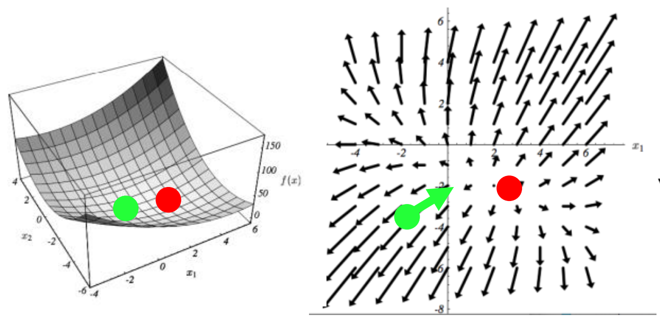
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$ **i=1**

SIGGRAPH ASIA 2018 TOKYO

85

Gradient Descent Minimization

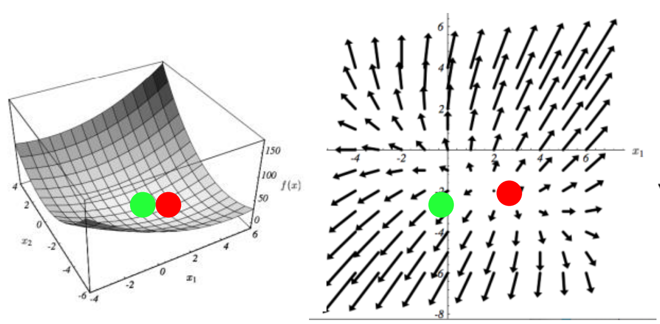


$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$ **i=1**

86

Gradient Descent Minimization

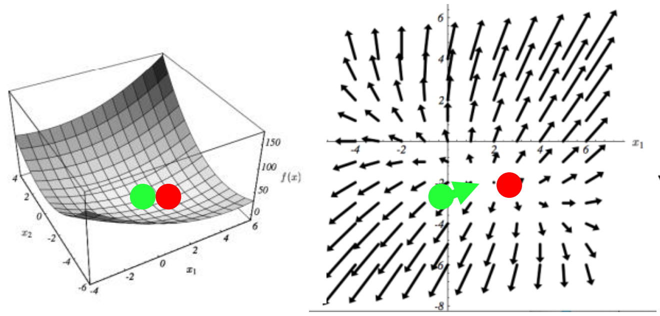


$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$ **i=2**

87

Gradient Descent Minimization



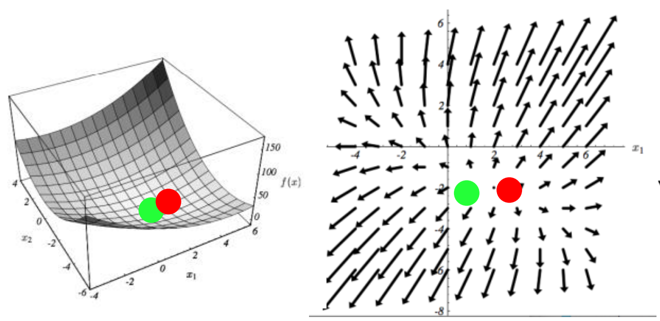
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$ **i=2**

SIGGRAPH ASIA 2018 TOKYO

88

Gradient Descent Minimization



$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$ **i=3**

SIGGRAPH ASIA 2018 TOKYO

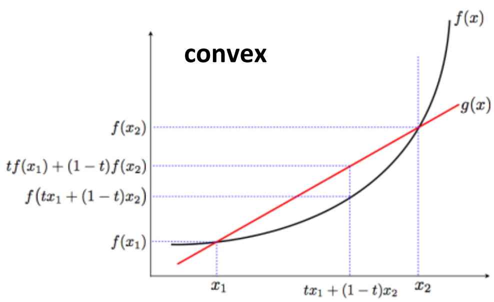
89

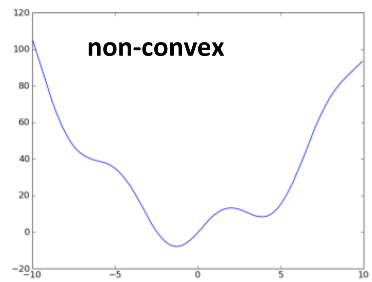
Gradient Descent Minimization

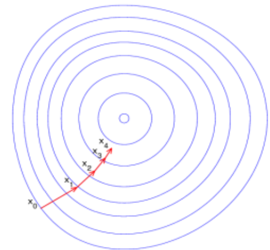
Initialize: $\mathbf{x}_0$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$

We can always make it converge for a convex function.





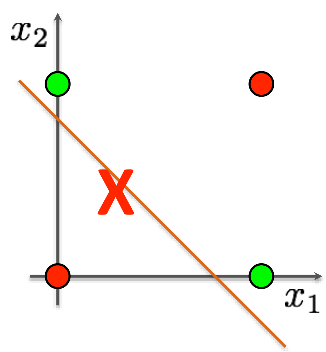


90

XOR Problem

$$y = f(x_1, x_2)$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$y = f(w_0, w_1, w_2) = \mathcal{H}(w_0 + w_1 x_1 + w_2 x_2)$$

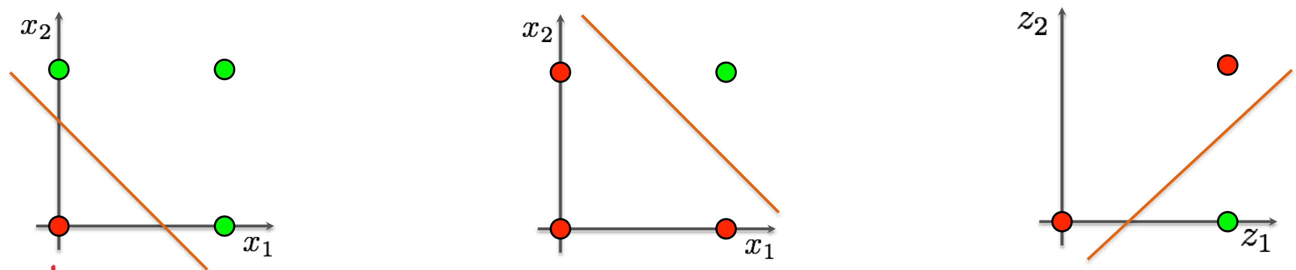
91

XOR Problem $y = f(z_1, z_2) = f(g_1(x_1, x_2), g_2(x_1, x_2))$

x_1	x_2	z_1
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	z_2
0	0	0
0	1	0
1	0	0
1	1	1

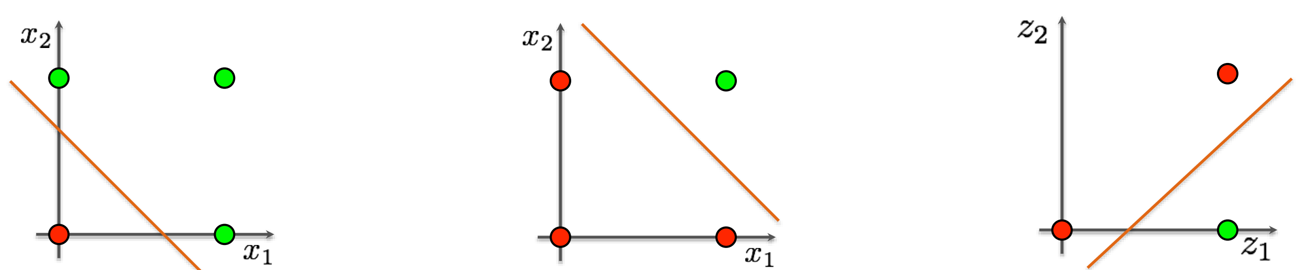
z_1	z_2	y
0	0	0
1	0	1
1	0	1
1	1	0



92

XOR Problem

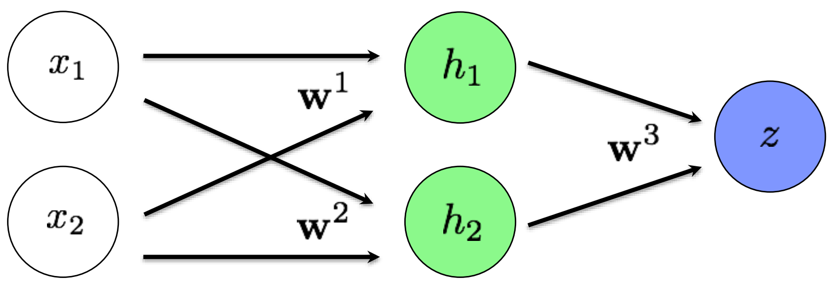
$$y = f(z_1, z_2) = f(g_1(x_1, x_2), g_2(x_1, x_2))$$
$$= f\left(\mathcal{H}(\mathbf{w}^1, x_1, x_2), \mathcal{H}(\mathbf{w}^2, x_1, x_2)\right)$$
$$= \mathcal{H}(\mathbf{w}^3, \mathcal{H}(g_1(\mathbf{w}^1, x_1, x_2)), \mathcal{H}(g_2(\mathbf{w}^2, x_1, x_2)))$$



93

XOR Problem

$$
\begin{aligned}
y = f(z_1, z_2) &= f(g_1(x_1, x_2), g_2(x_1, x_2)) \\
&= f\left(\mathcal{H}(\mathbf{w}^1, x_1, x_2), \mathcal{H}(\mathbf{w}^2, x_1, x_2)\right) \\
&= \mathcal{H}(\mathbf{w}^3, \mathcal{H}(g_1(\mathbf{w}^1, x_1, x_2)), \mathcal{H}(g_2(\mathbf{w}^2, x_1, x_2)))
\end{aligned}
$$



SIGGRAPH ASIA 2018 TOKYO 94

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course CreativeAI: Deep Learning for Graphics

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

Neural Network Basics

Niloy Mitra	Iasonas Kokkinos	Paul Guerrero	Nils Thuerey	Tobias Ritschel
UCL	UCL/Facebook	UCL	TU Munich	UCL

UCL

facebook Artificial Intelligence Research

TUM Technische Universität München

Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

SIGGRAPH Asia Course CreativeAI: Deep Learning for Graphics 2

Introduction to Neural Networks

SIGGRAPH ASIA 2018 TOKYO

Goal: Learn a Parametric Function

$$f_\theta : \mathbb{X} \longrightarrow \mathbb{Y}$$

θ : function parameters, these are learned

$\mathbb{X}$: source domain

$\mathbb{Y}$: target domain

Examples:

Image Classification: $f_\theta : \mathbb{R}^{w \times h \times c} \longrightarrow \{0, 1, \ldots, k-1\}$

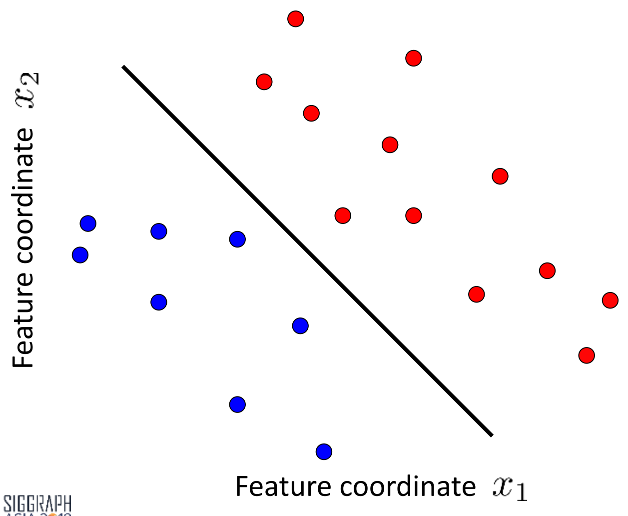
$w \times h \times c$: image dimensions k : class count

Image Synthesis: $f_\theta : \mathbb{R}^{n} \longrightarrow \mathbb{R}^{w \times h \times c}$

n : latent variable count $w \times h \times c$: image dimensions

SIGGRAPH ASIA 2018 TOKYO

Machine Learning 101: Linear Classifier



$$f_\theta : \mathbb{R}^n \longrightarrow \{0,1\}$$

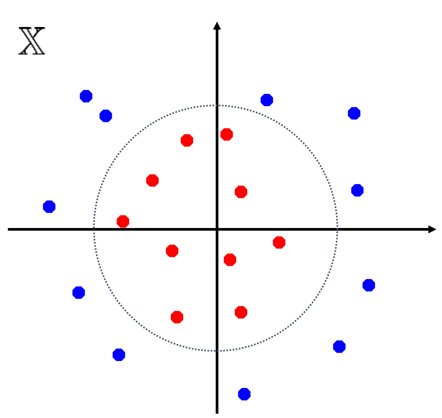
$$f_\theta(x) = \begin{cases} 1 & \text{if } wx + b \geq 0 \\ 0 & \text{if } wx + b < 0 \end{cases}$$

$$\theta = \{w, b\}$$

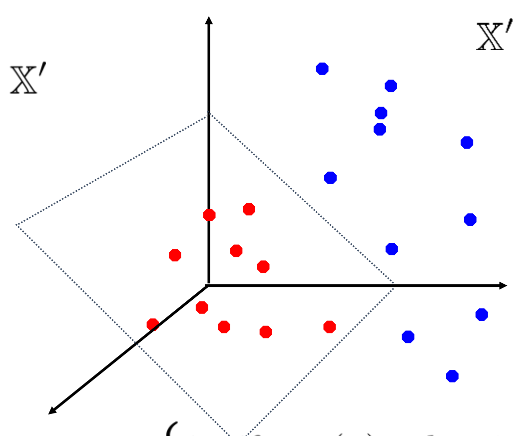
Each data point has a class label:

$$y^i = \begin{cases} 1 & (\text{red}) \\ 0 & (\text{blue}) \end{cases}$$

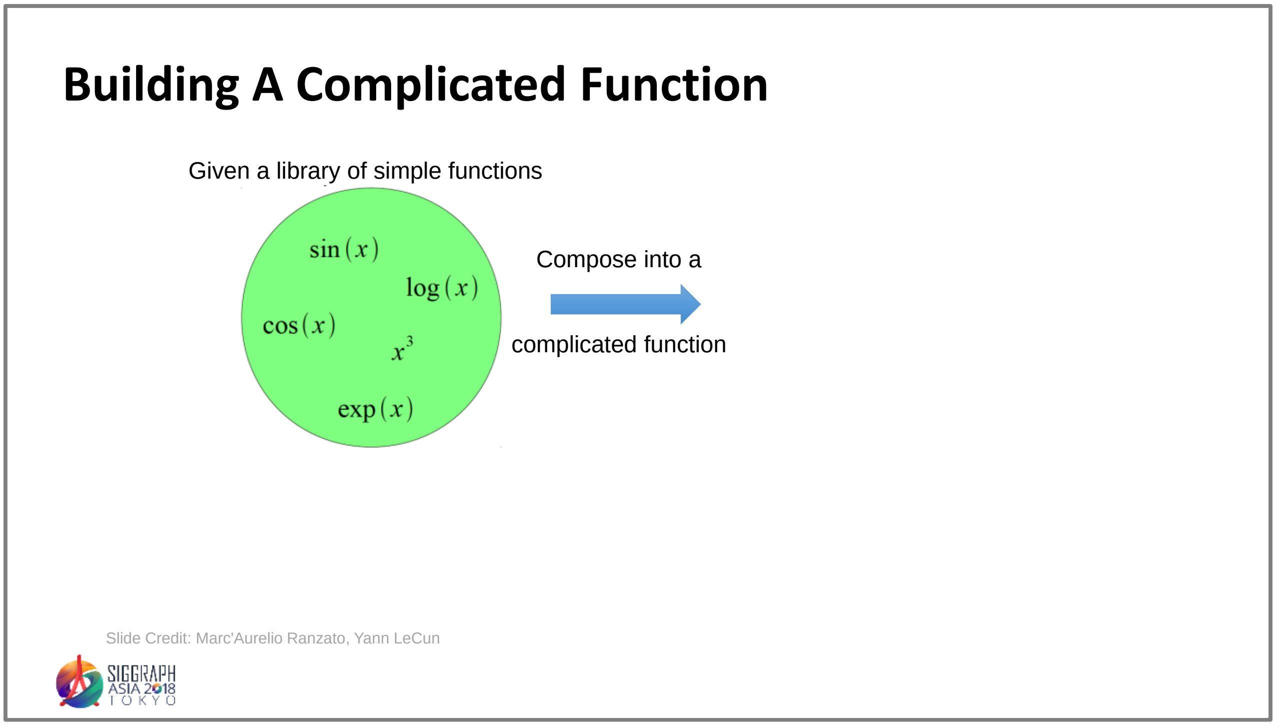
Nonlinear decision boundaries

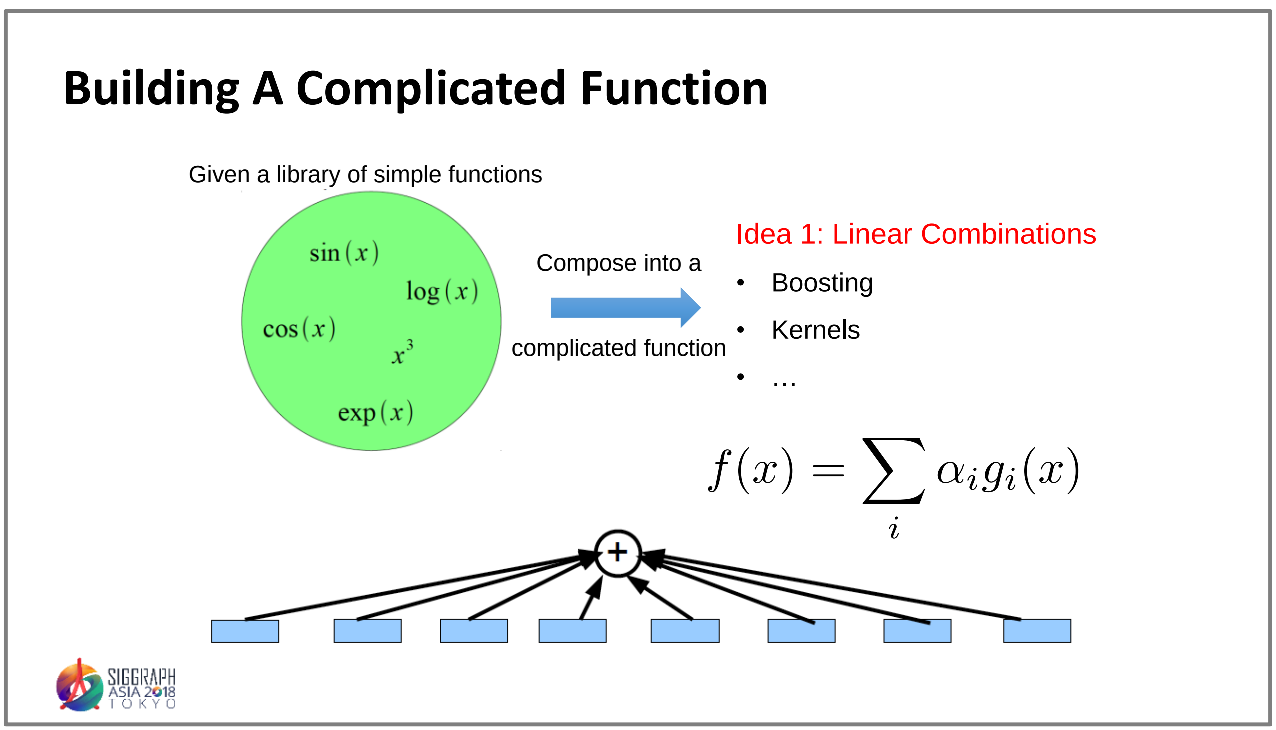


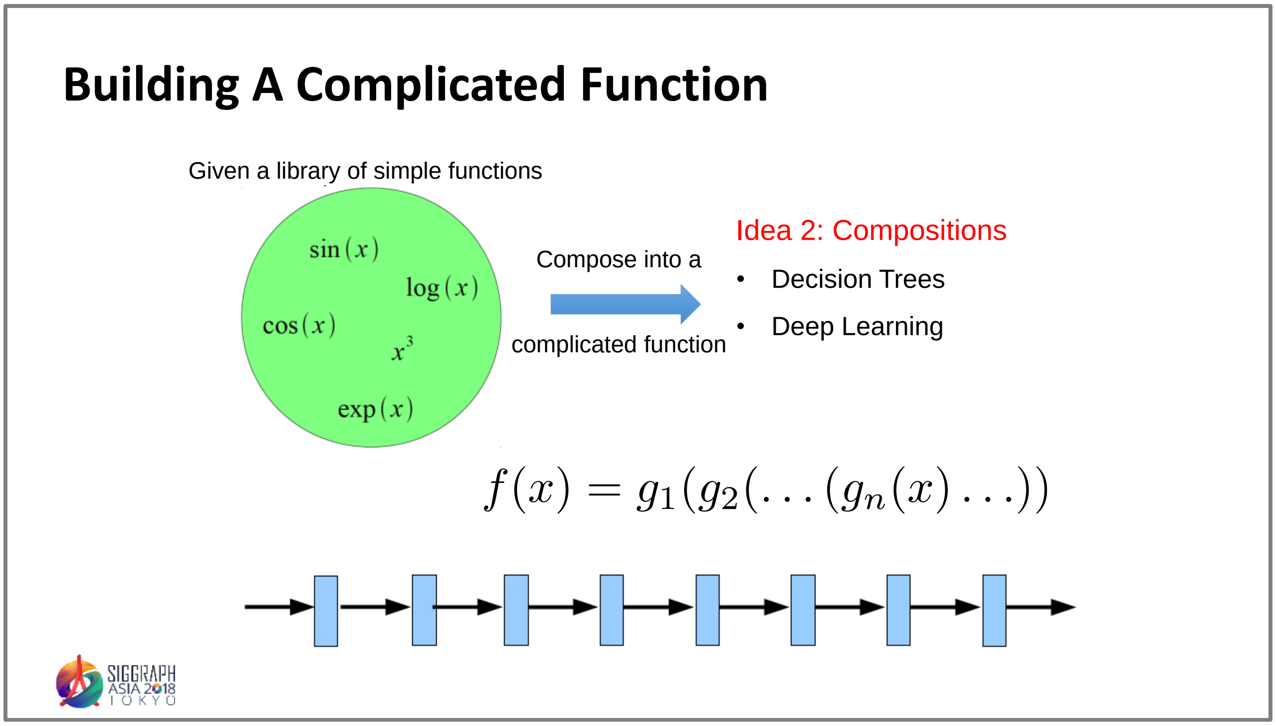
$$g : \mathbb{X} \longrightarrow \mathbb{X}'$$

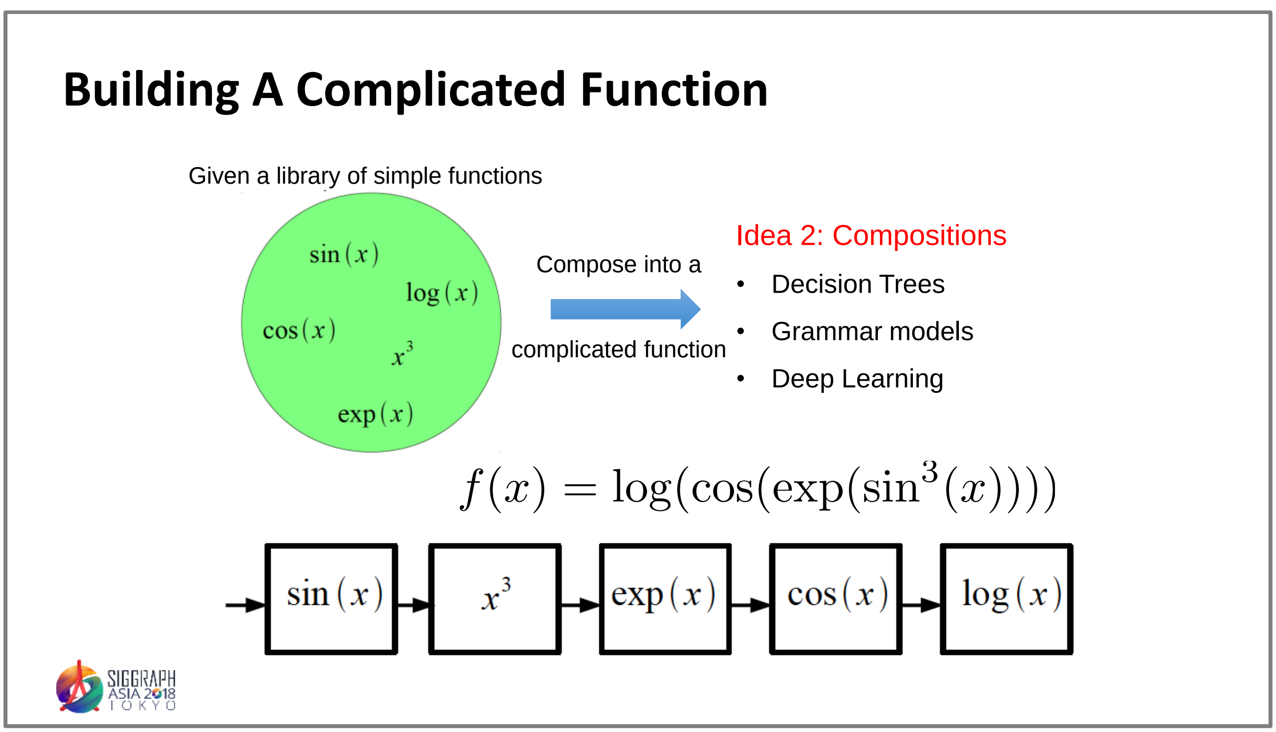


$$f_\theta(x) = \begin{cases} 1 & \text{if } w\ g(x) + b \geq 0 \\ 0 & \text{if } w\ g(x) + b < 0 \end{cases}$$


Building A Complicated Function
Given a library of simple functions
$\sin(x)$
$\log(x)$
$\cos(x)$
x^3
$\exp(x)$
Compose into a
complicated function
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun
SIGGRAPH ASIA 2018 TOKYO

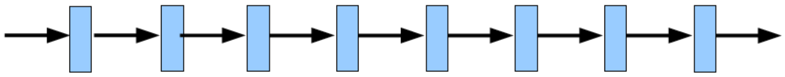

Building A Complicated Function
Given a library of simple functions
$\sin(x)$
$\log(x)$
$\cos(x)$
x^3
$\exp(x)$
Compose into a
complicated function
Idea 1: Linear Combinations
• Boosting
• Kernels
• …
$f(x) = \sum_i \alpha_i g_i(x)$
SIGGRAPH ASIA 2018 TOKYO

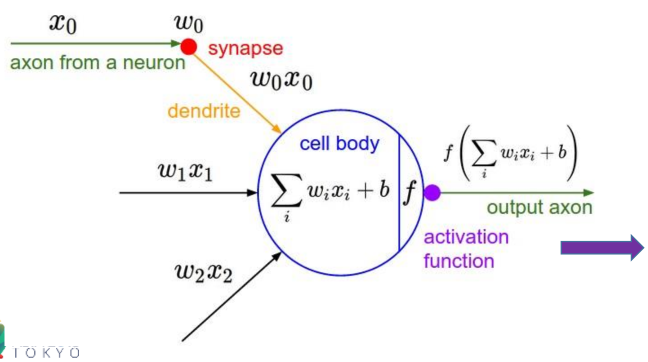

Building A Complicated Function
Given a library of simple functions
sin(x)
log(x)
cos(x)
x^3
exp(x)
Compose into a
complicated function
Idea 2: Compositions
• Decision Trees
• Deep Learning
$f(x) = g_1(g_2(\ldots(g_n(x)\ldots))$

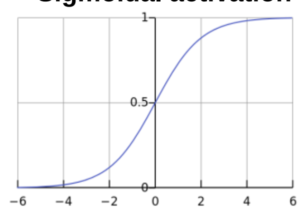

Building A Complicated Function
Given a library of simple functions
sin(x)
log(x)
cos(x)
x^3
exp(x)
Compose into a
complicated function
Idea 2: Compositions
• Decision Trees
• Grammar models
• Deep Learning
$f(x) = \log(\cos(\exp(\sin^3(x))))$
sin(x)
x^3
exp(x)
cos(x)
log(x)

'Neuron': Cascade of Linear and Nonlinear Function

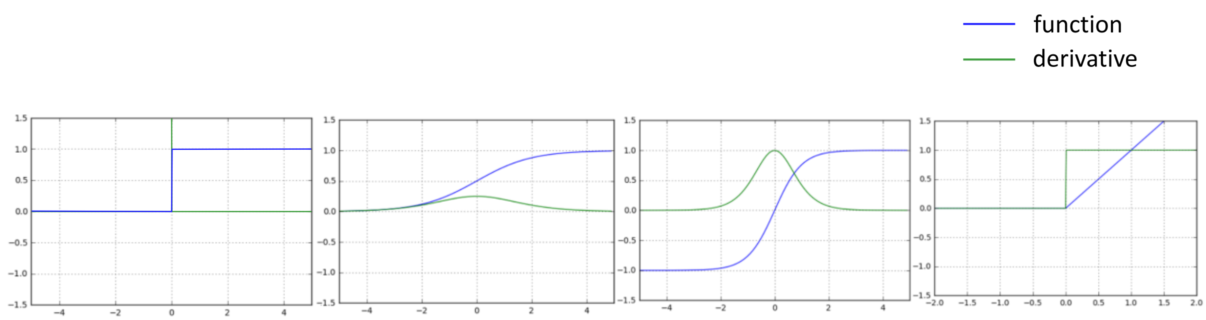
basic building block







Activation functions



Step ("perceptron")

$$g(a) = \begin{cases} 0 & a < 0 \\ 1 & a \geq 0 \end{cases}$$

Sigmoidal ("logistic")

$$g(a) = \frac{1}{1+\exp(-a)}$$

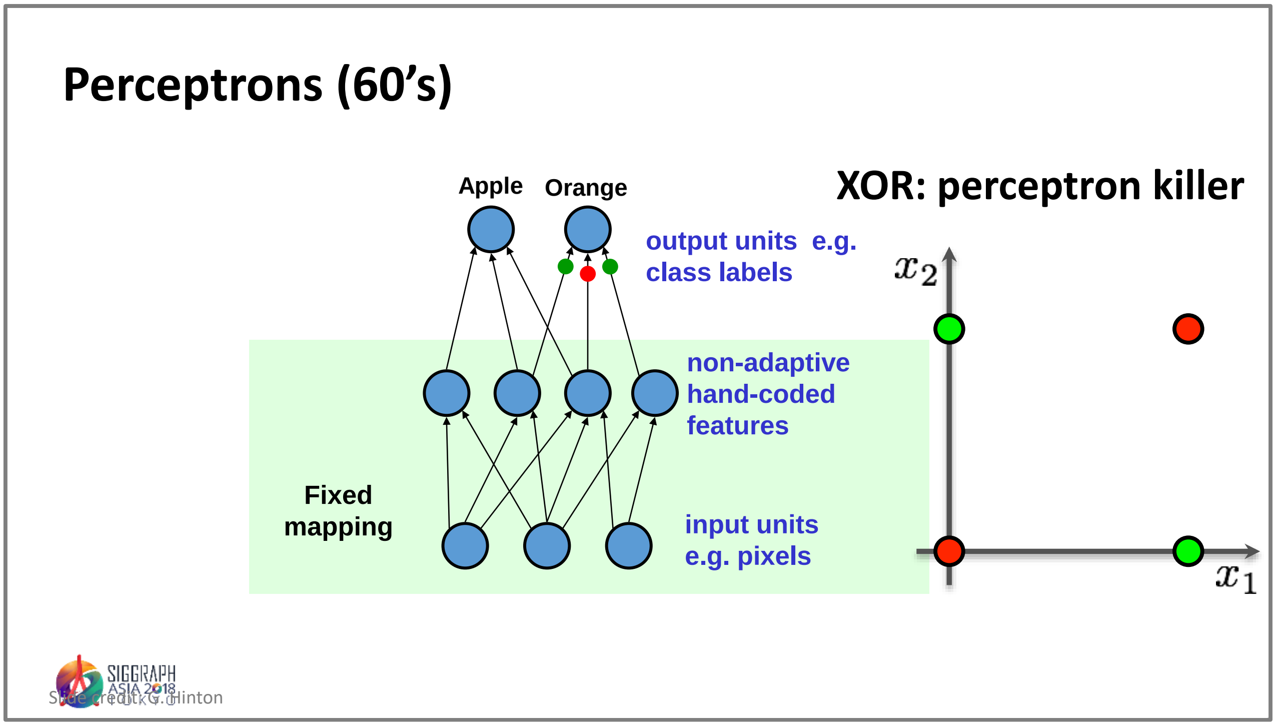
Hyperbolic tangent

$$g(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$$

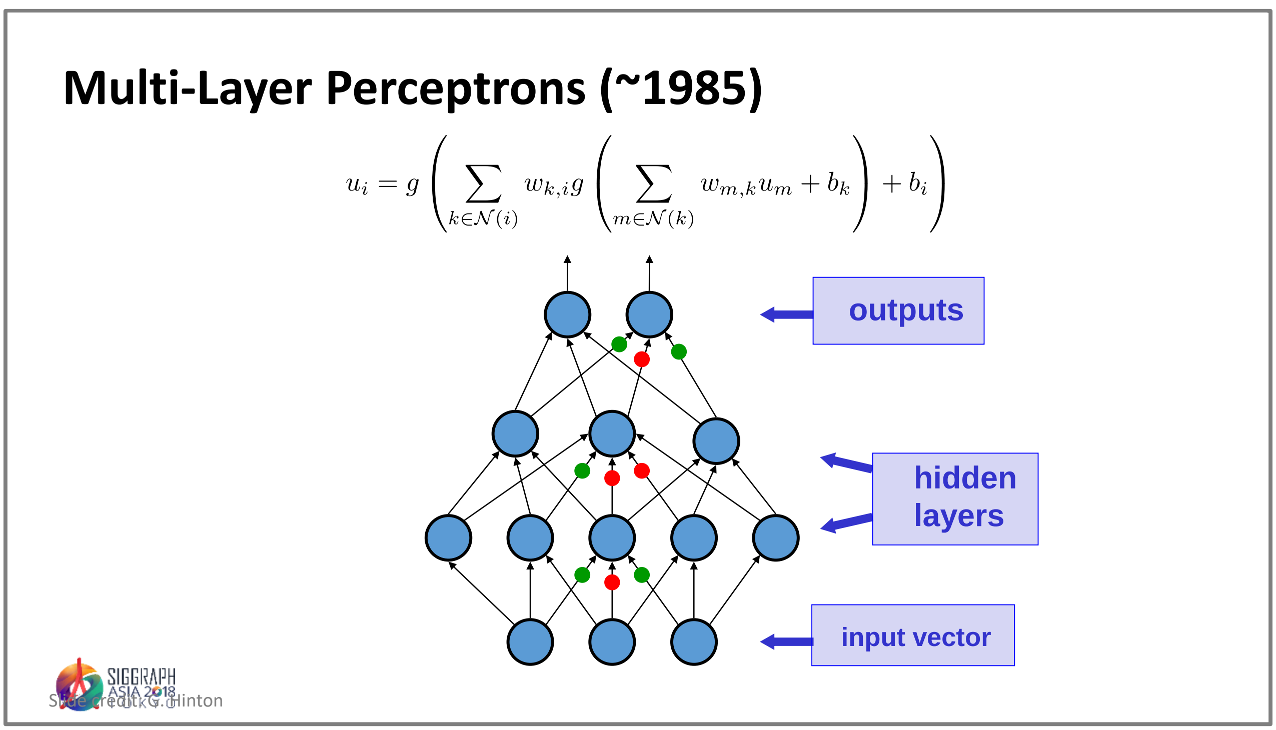
Rectified Linear Unit (RELU)

$$g(a) = \max(0, a)$$

Image Credit: Olivier Grisel and Charles Ollion



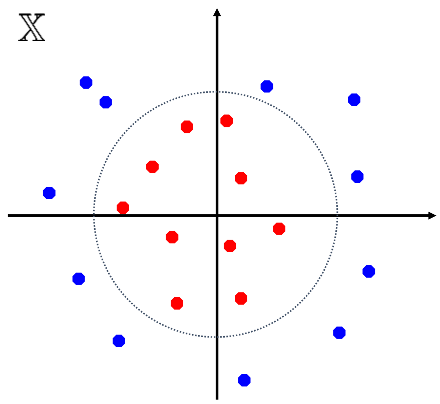
Perceptrons (60's)
Apple
Orange
output units e.g. class labels
non-adaptive hand-coded features
Fixed mapping
input units e.g. pixels
XOR: perceptron killer
x_2
x_1
Slide credit: G. Hinton



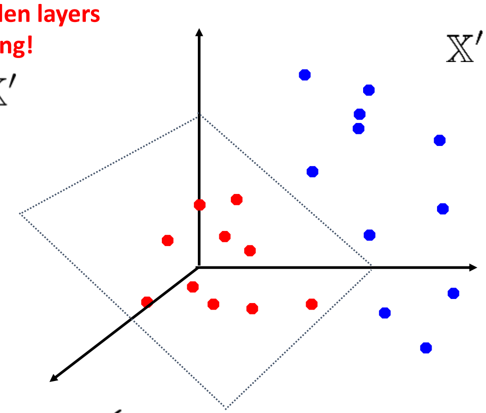
Multi-Layer Perceptrons (~1985)
$$u_i = g\left(\sum_{k\in\mathcal{N}(i)} w_{k,i}\, g\left(\sum_{m\in\mathcal{N}(k)} w_{m,k}u_m + b_k\right) + b_i\right)$$
outputs
hidden layers
input vector
Slide credit: G. Hinton

Reminder: Non-linear decision boundaries

This is what the hidden layers should be doing!



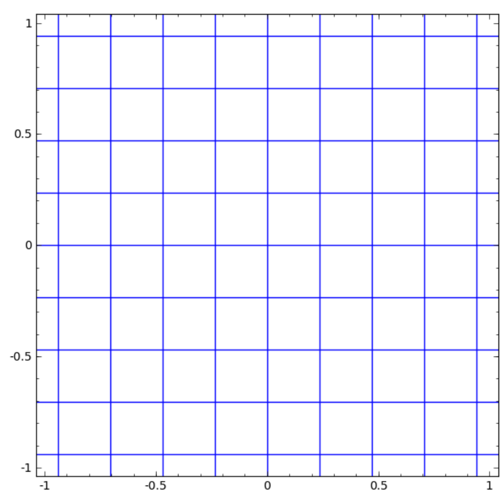
$$g : \mathbb{X} \longrightarrow \mathbb{X}'$$



$$f_\theta(x) = \begin{cases} 1 & \text{if } w \; g(x) + b \geq 0 \\ 0 & \text{if } w \; g(x) + b < 0 \end{cases}$$

Nonlinear mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^2$

Evolution of isocontours as parameters change



$$y_1 = g(w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3})$$
$$y_2 = g(w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3})$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = g(\mathbf{W}\mathbf{x})$$

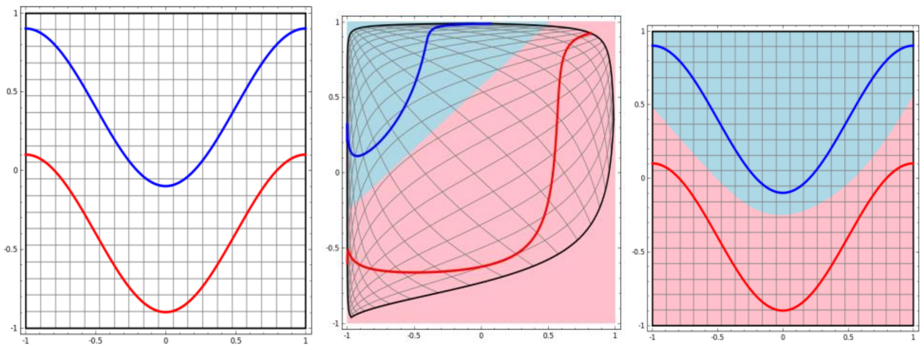
http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

From non-separable to linearly separable

Non-linearly separable data

Data mapped to learned space

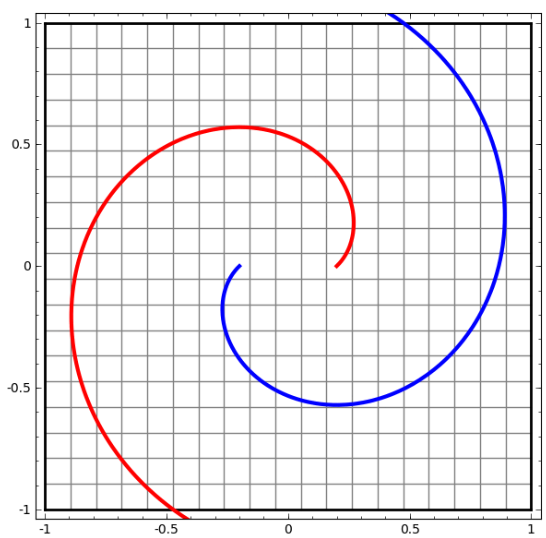
Decision function



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

SIGGRAPH ASIA 2018 TOKYO

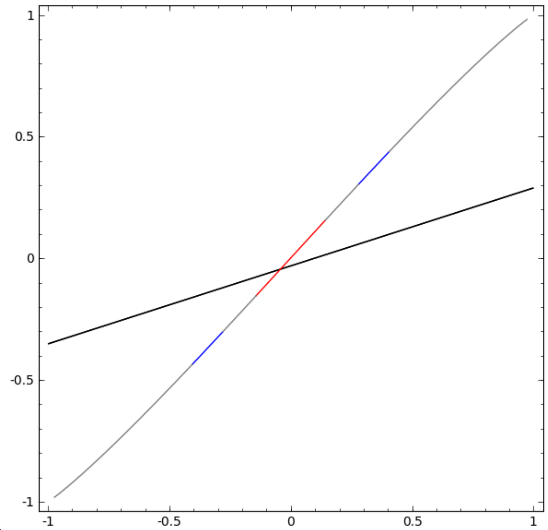
Linearizing a 2D classification task (4 hidden layers)



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

SIGGRAPH ASIA 2018 TOKYO

Linearization: may need higher dimensions

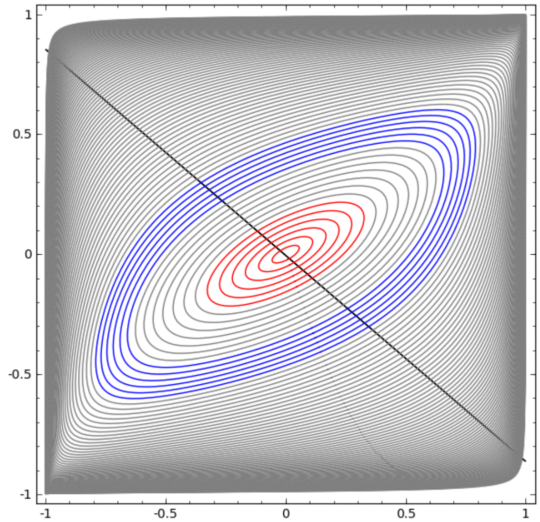


Points in 1D,
Decision in 2D

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

SIGGRAPH ASIA 2018 TOKYO

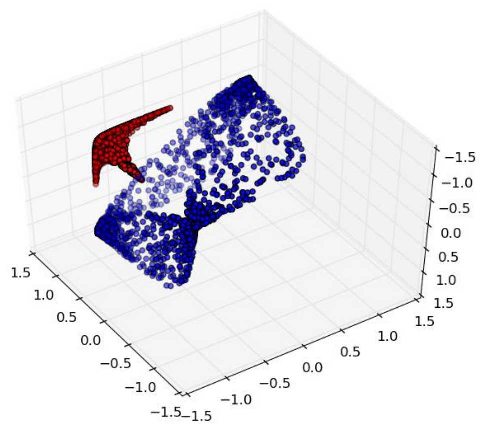
Linearization: may need higher dimensions



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

SIGGRAPH ASIA 2018 TOKYO

Linearization: may need higher dimensions



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

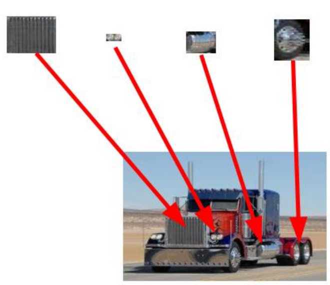
SIGGRAPH ASIA 2018 TOKYO

Hidden Layers: intuitively, what do they do?

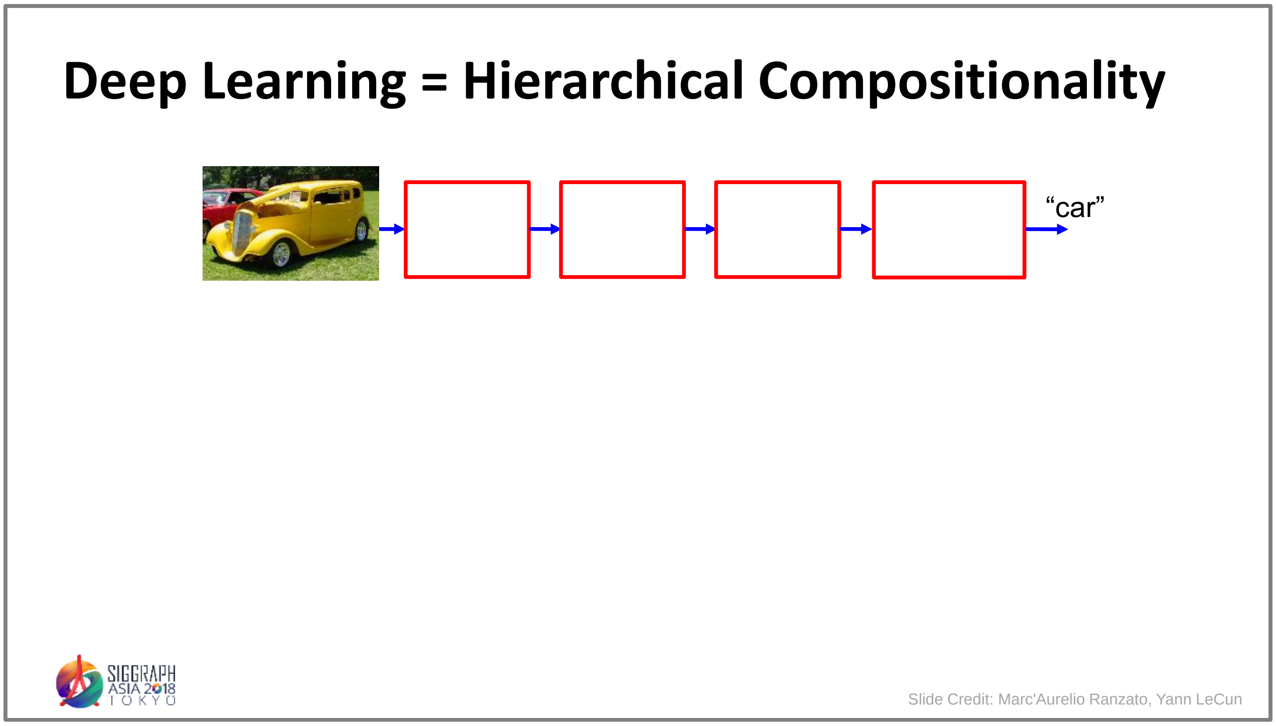
Intuition: learn “dictionary” for objects

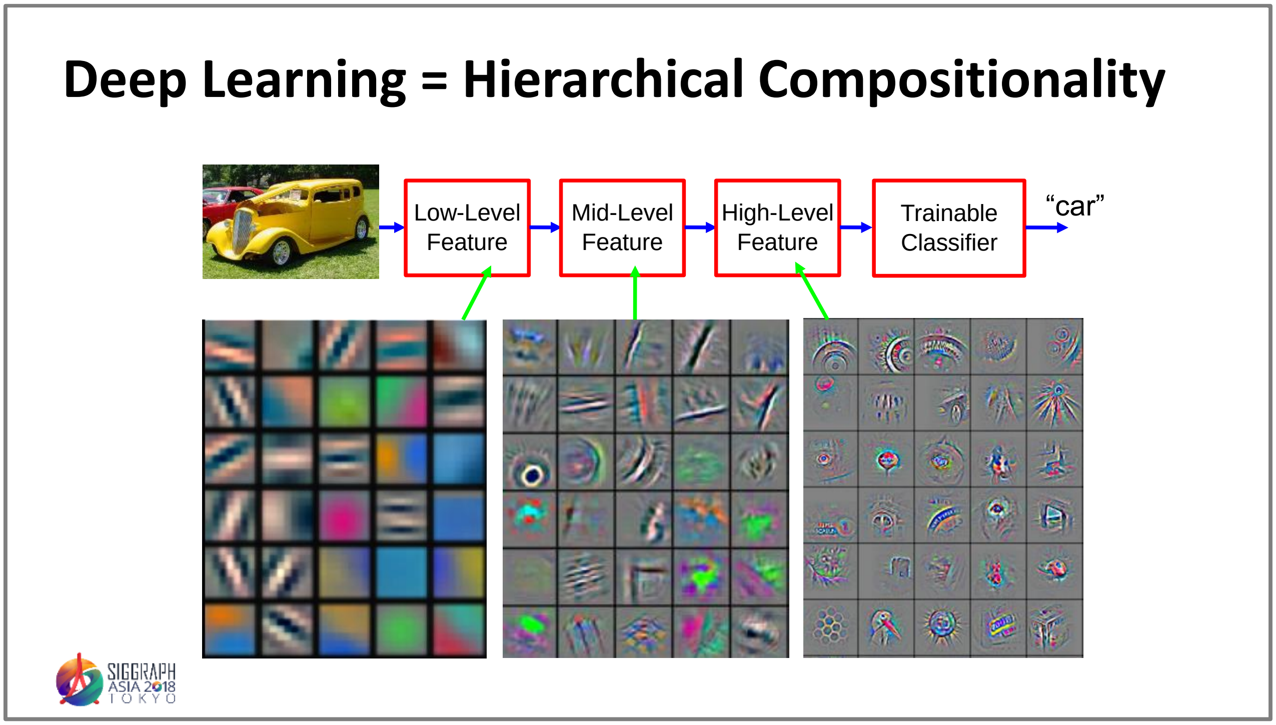
“Distributed representation”:
represent (and classify) objects by mixing & mashing reusable parts

[0 0 **1** 0 0 0 0 **1** 0 0 **1** **1** 0 0 **1** 0 …] truck feature

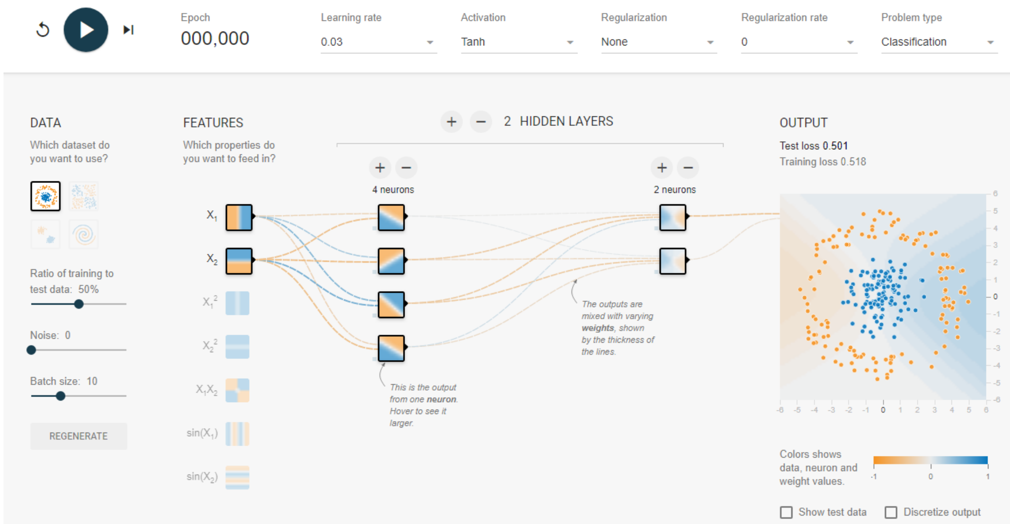


Slide Credit: Marc'Aurelio Ranzato, Yann LeCun


Deep Learning = Hierarchical Compositionality
"car"
SIGGRAPH ASIA 2018 TOKYO
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun


Deep Learning = Hierarchical Compositionality
Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier
"car"
SIGGRAPH ASIA 2018 TOKYO

MLP Demo: playground.tensorflow.org



Training and Optimization

Neural Network Training: Old & New Tricks

Old:

- **Back-propagation algorithm**
- **Stochastic Gradient Descent, Momentum, "weight decay"**

New: (last 5-6 years)

- **Dropout**
- **ReLUs**
- **Batch Normalization**
- **Residual Networks**

Training Goal

Our network implements a parametric function:

$$f_\theta : \mathbb{X} \longrightarrow \mathbb{Y} \qquad \hat{y} = f(x;\theta)$$

During training, we search for parameters that minimize a loss:

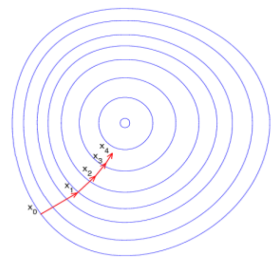
$$\min_\theta L_f(\theta)$$

Example: L2 regression loss given target (x^i, y^i) pairs :

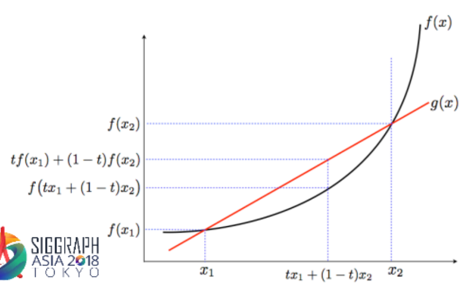
$$L_f(\theta) = \sum_i \|f(x^i;\theta) - y^i\|_2^2$$

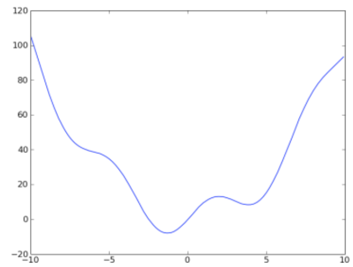
Gradient Descent Minimization Method

Initialize: θ_0
Update: $\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i)$

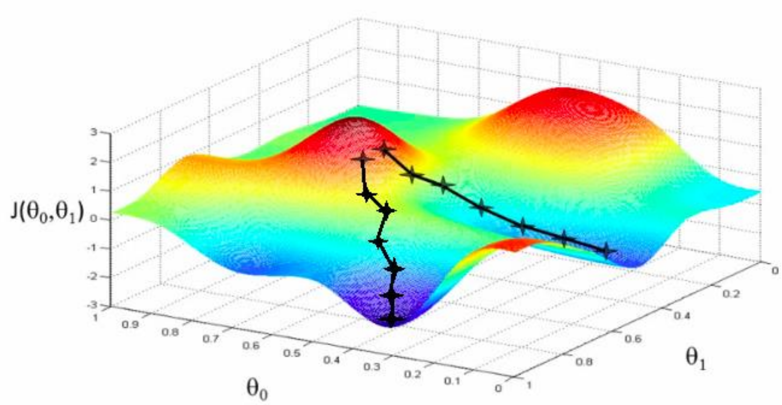


We can always make it converge for a **convex** function





Multiple Local Minima, based on initialization

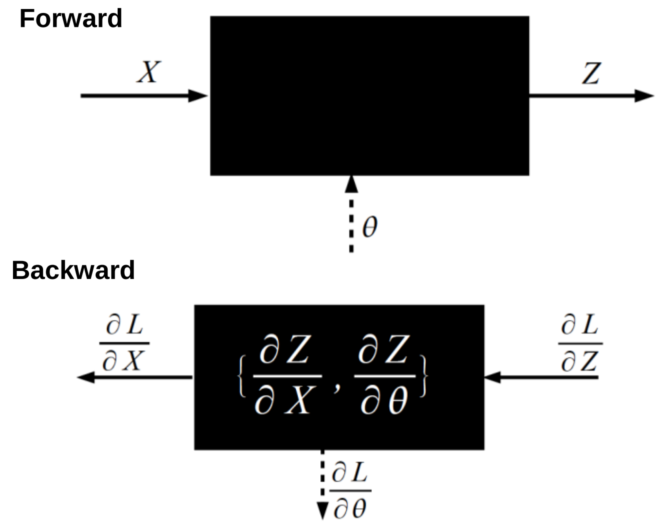


Empirically all are almost equally good

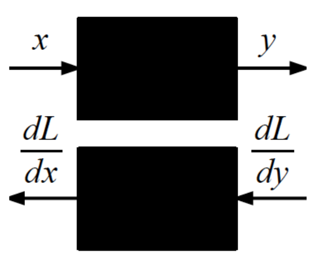
Central research topic: how can this happen?

On to the gradients!

All you need is gradients



Chain Rule

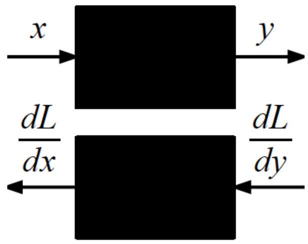


Given $y(x)$ and dL/dy,

What is dL/dx ?

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Chain Rule

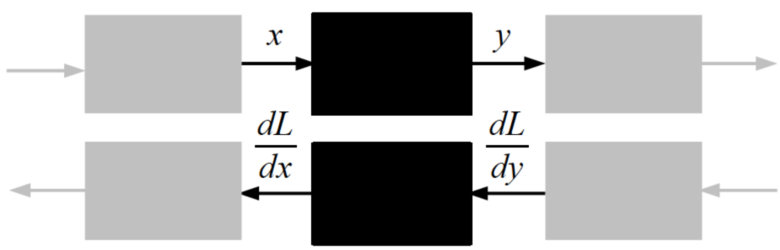


Given $y(x)$ and dL/dy,
What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

‘Another Brick in the Wall’



Given $y(x)$ and dL/dy,
What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

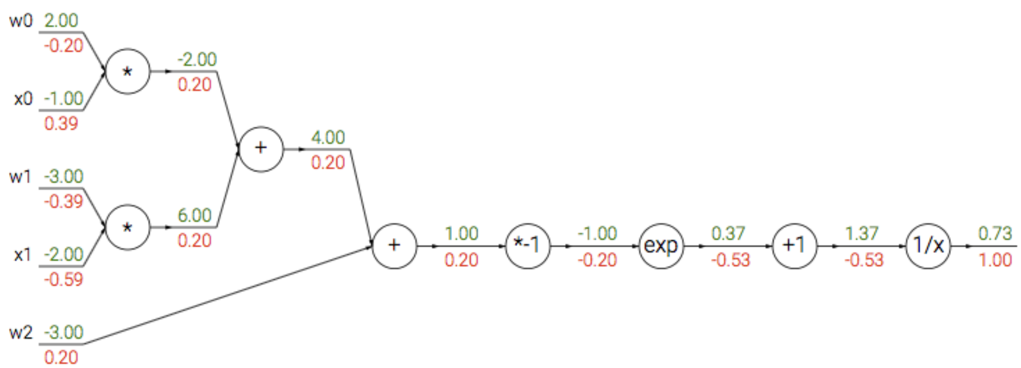
Toy example: single sigmoidal unit

$$f(w,x) = \frac{1}{1+\exp(-(w_0x_0+w_1x_1+w_2))}$$

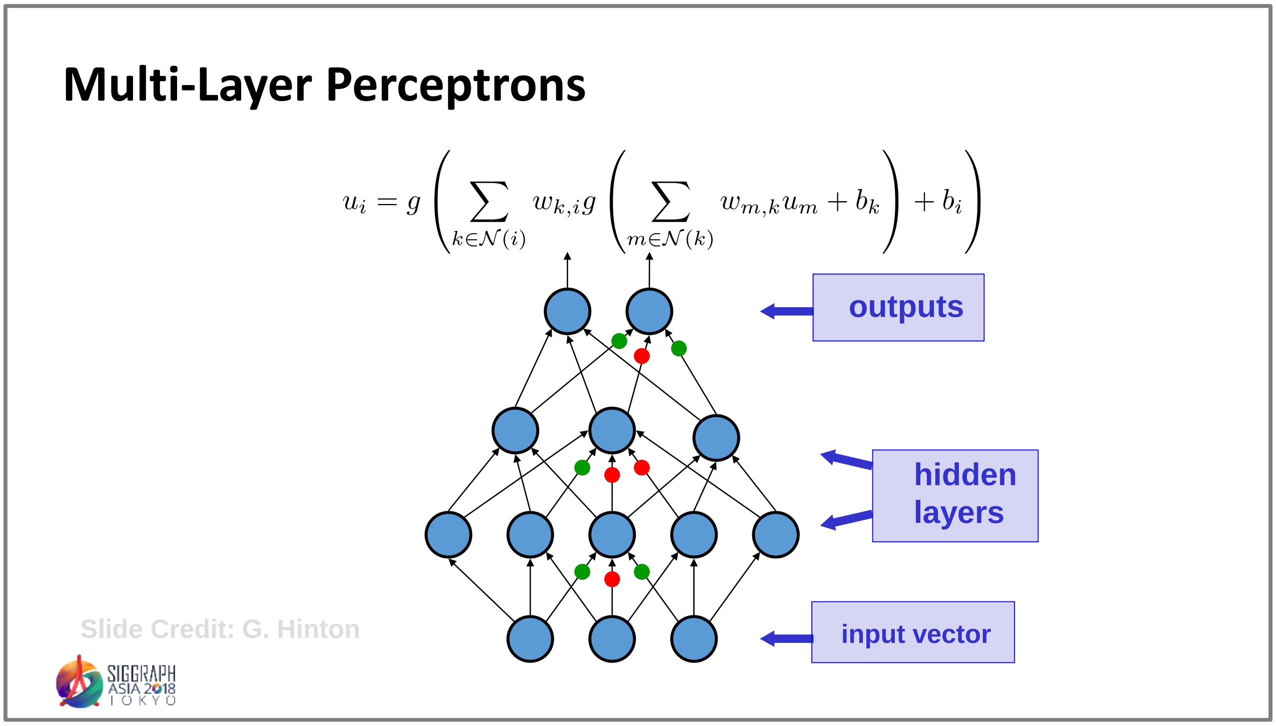
Composition of differentiable blocks:

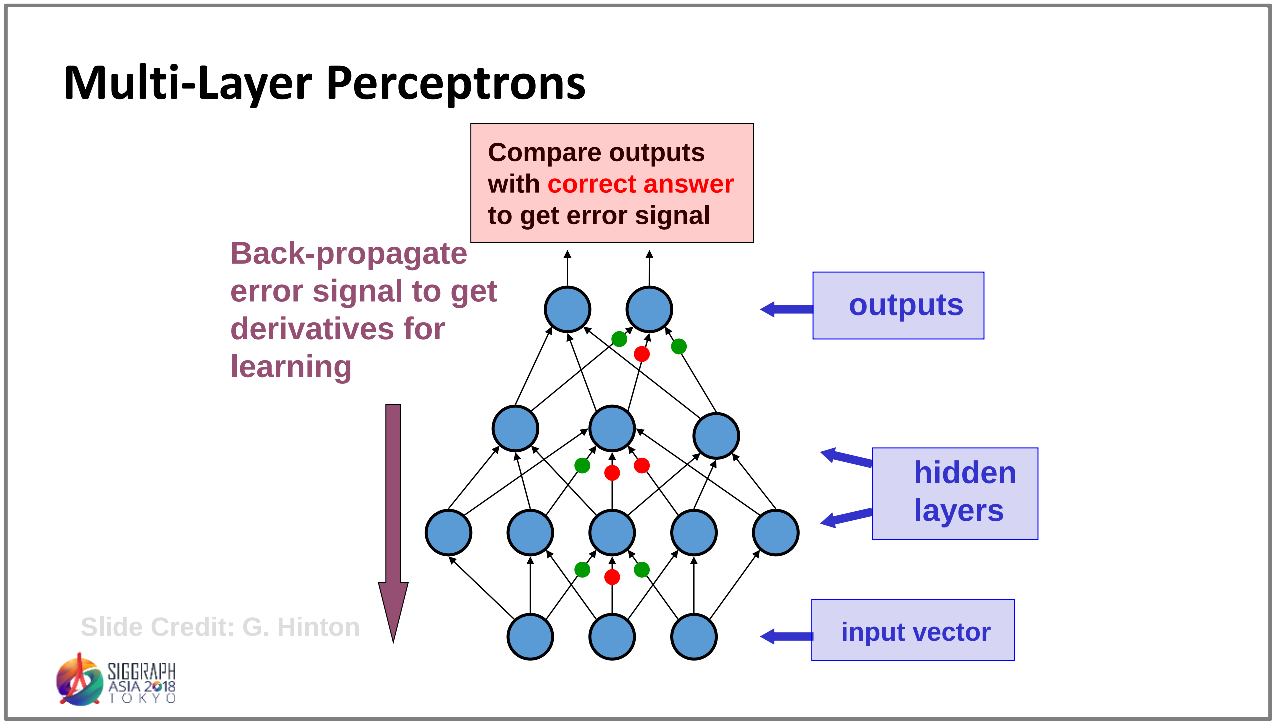
$$f(x) = \frac{1}{x} \quad \rightarrow \quad f'(x) = -\frac{1}{x^2}$$
$$f_c(x) = c + x \quad \rightarrow \quad f'(x) = 1$$
$$f(x) = e^x \quad \rightarrow \quad f'(x) = e^x$$
$$f_a(x) = ax \quad \rightarrow \quad f'(x) = a$$

Computation graph & automatic differentiation



Slide Credit: Justin Johnson


Multi-Layer Perceptrons
$u_i = g\left(\sum_{k\in\mathcal{N}(i)} w_{k,i}\, g\left(\sum_{m\in\mathcal{N}(k)} w_{m,k} u_m + b_k\right) + b_i\right)$
outputs
hidden layers
input vector
Slide Credit: G. Hinton


Multi-Layer Perceptrons
Compare outputs with correct answer to get error signal
Back-propagate error signal to get derivatives for learning
outputs
hidden layers
input vector
Slide Credit: G. Hinton

Back-propagation Algorithm

Training Goal

Our network implements a parametric function:

$$f_\theta : \mathbb{X} \longrightarrow \mathbb{Y} \qquad \hat{y} = f(x;\theta)$$

During training, we search for parameters that minimize a loss:

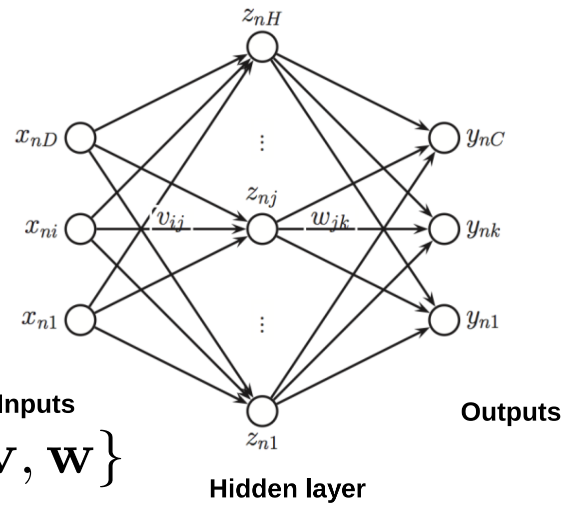
$$\min_\theta L_f(\theta)$$

Example: L2 regression loss given target (x^i, y^i) pairs :

$$L_f(\theta) = \sum_i \|f(x^i;\theta) - y^i\|_2^2$$

A Neural Network for Multi-way Classification

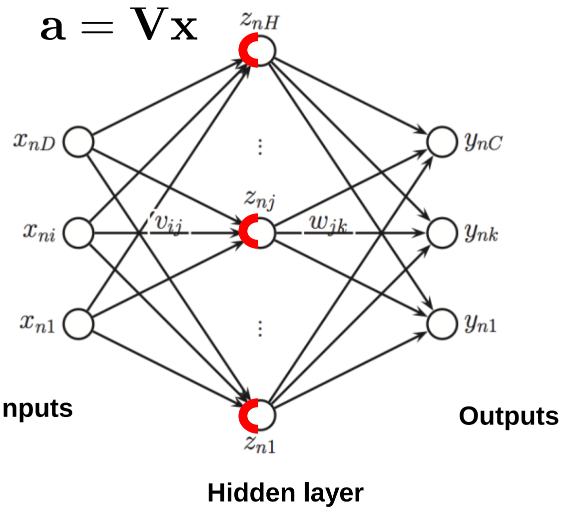
$$\mathbf{x}_n \xrightarrow{\mathbf{V}} \mathbf{a}_n \xrightarrow{g} \mathbf{z}_n \xrightarrow{\mathbf{W}} \mathbf{b}_n \xrightarrow{h} \hat{\mathbf{y}}_n$$

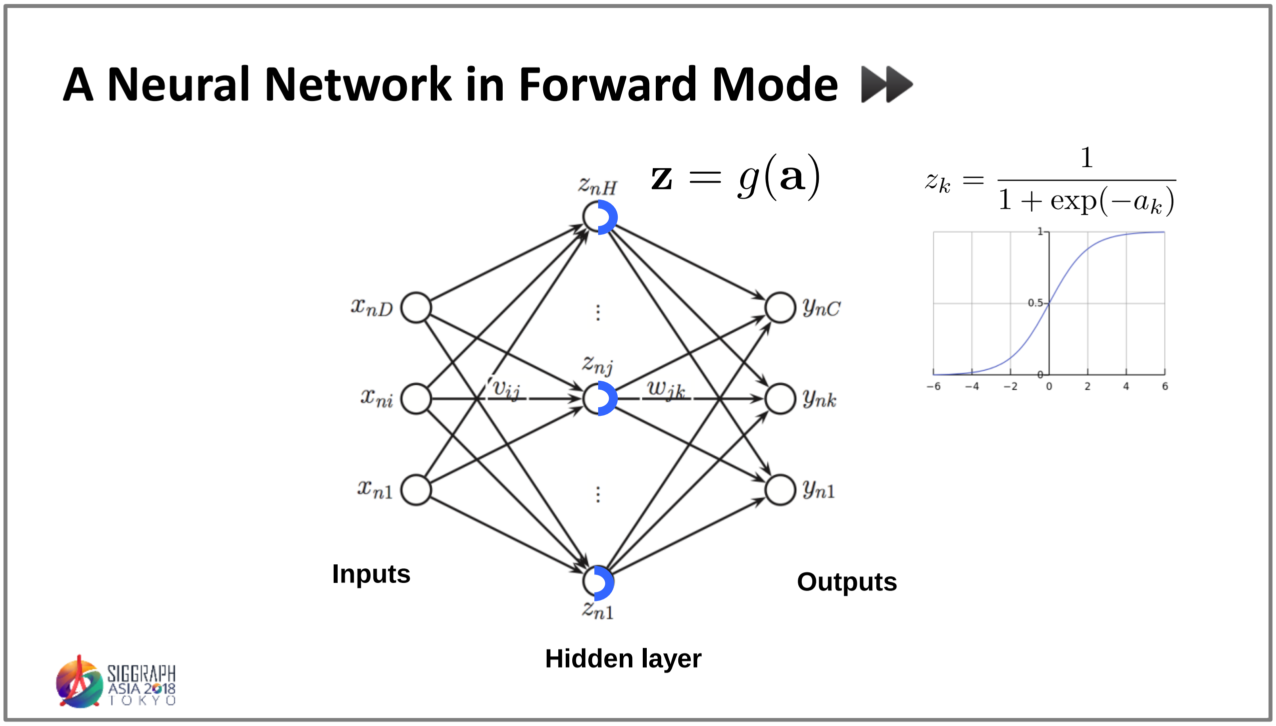


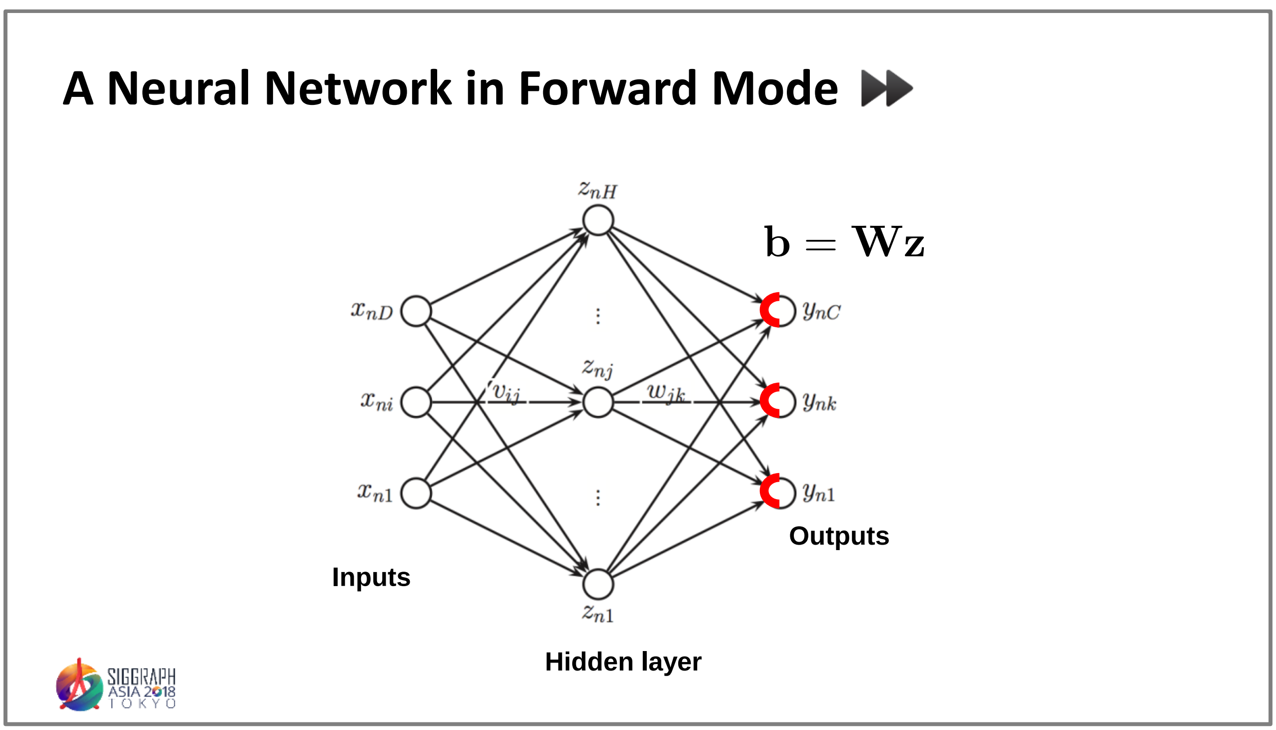
Parameters: $\theta = \{\mathbf{v}, \mathbf{w}\}$

A Neural Network in Forward Mode

$$\mathbf{a} = \mathbf{V}\mathbf{x}$$

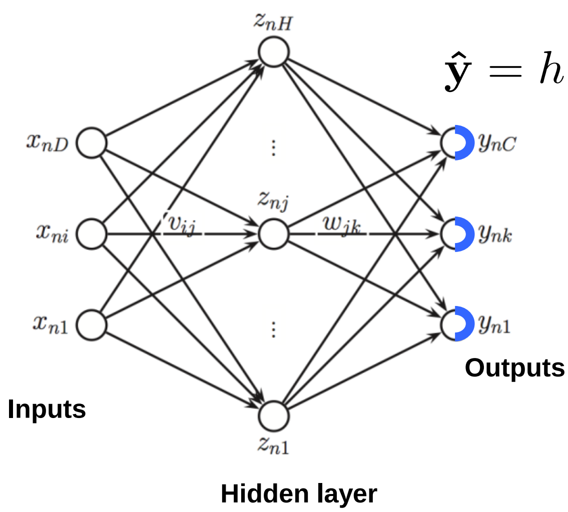



A Neural Network in Forward Mode
$\mathbf{z} = g(\mathbf{a})$
$z_k = \frac{1}{1 + \exp(-a_k)}$
z_{nH}
z_{nj}
z_{n1}
x_{nD}
x_{ni}
x_{n1}
v_{ij}
w_{jk}
y_{nC}
y_{nk}
y_{n1}
Inputs
Outputs
Hidden layer
SIGGRAPH ASIA 2018 TOKYO


A Neural Network in Forward Mode
$\mathrm{b} = \mathbf{W}\mathrm{z}$
z_{nH}
z_{nj}
z_{n1}
x_{nD}
x_{ni}
x_{n1}
v_{ij}
w_{jk}
y_{nC}
y_{nk}
y_{n1}
Inputs
Outputs
Hidden layer
SIGGRAPH ASIA 2018 TOKYO

A Neural Network in Forward Mode ▶▶

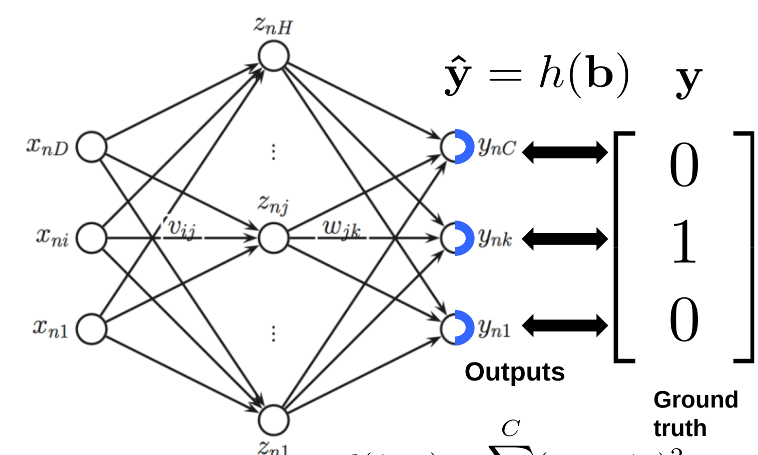
$$\hat{\mathbf{y}} = h(\mathbf{b})$$



Objective for linear regression

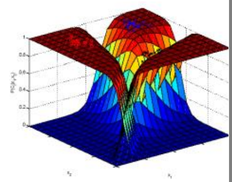
$$h(\mathbf{b}) = \mathbf{b}$$

$$\hat{\mathbf{y}} = h(\mathbf{b}) \quad \mathbf{y}$$



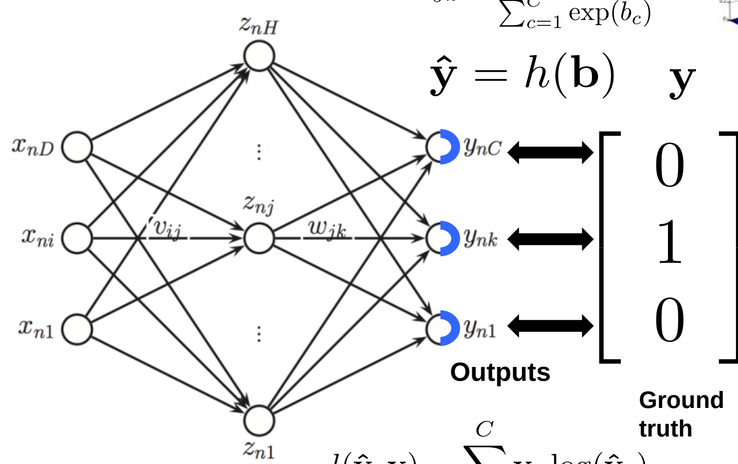
$$l(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{c=1}^{C} (\mathbf{y}_c - \hat{\mathbf{y}}_c)^2$$

Objective for multi-class classification



Softmax unit $\hat{y}_k = \frac{\exp(b_k)}{\sum_{c=1}^{C} \exp(b_c)}$

$\hat{\mathbf{y}} = h(\mathbf{b})$ $\mathbf{y}$



$$l(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{c=1}^{C} \mathbf{y}_c \log(\hat{\mathbf{y}}_c)$$ `Cross-entropy' loss

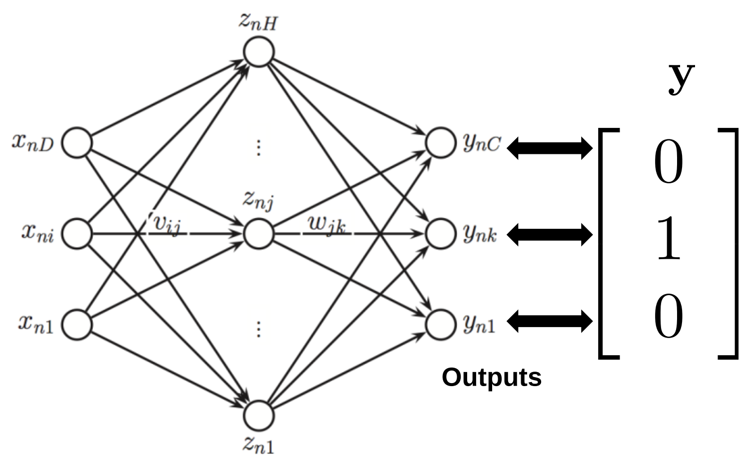
Neural network in forward mode: recap

Network output: $\hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{v}, \mathbf{w})$

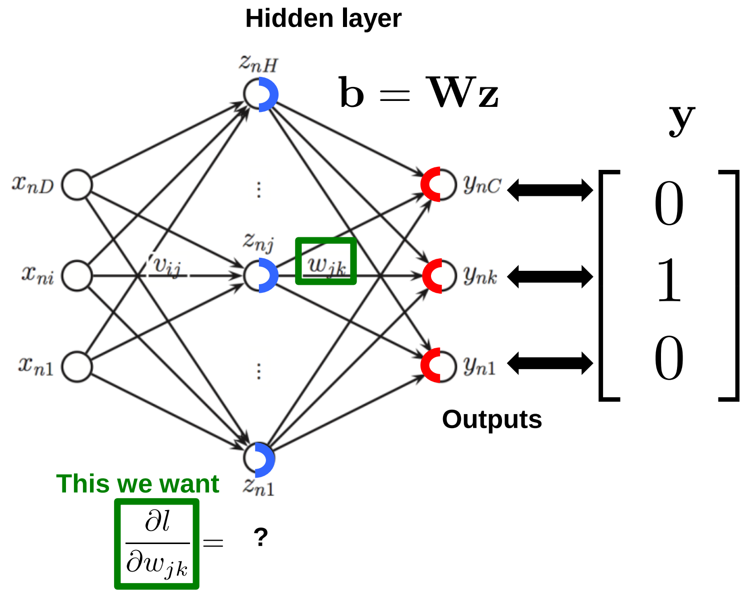
Loss (prediction error): $l(\hat{\mathbf{y}}, \mathbf{y})$

What we need to compute for gradient descent: $\frac{\partial l(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{v}_i}$ $\frac{\partial l(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{w}_j}$

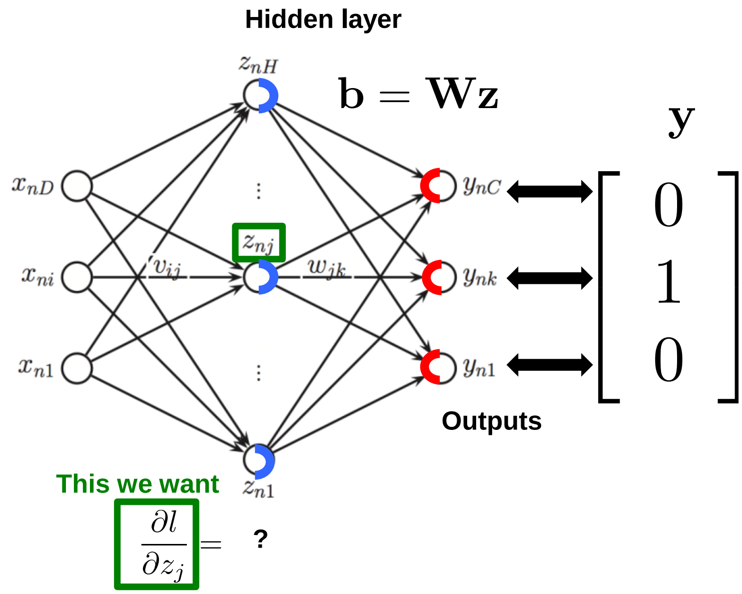
A Neural Network in Backward Mode



A Neural Network in Backward Mode

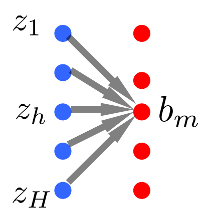


A Neural Network in Backward Mode



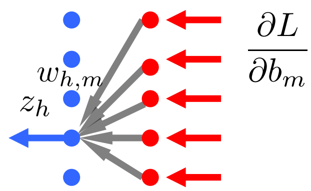
Linear Layer in Forward Mode: All For One

$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$



Linear Layer in Backward Mode: One From All

$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$

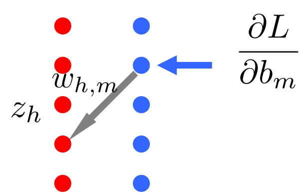


$$\frac{\partial L}{\partial z_h} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} \cdot \frac{\partial b_c}{\partial z_h} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} w_{h,c}$$

SIGGRAPH ASIA 2018 TOKYO

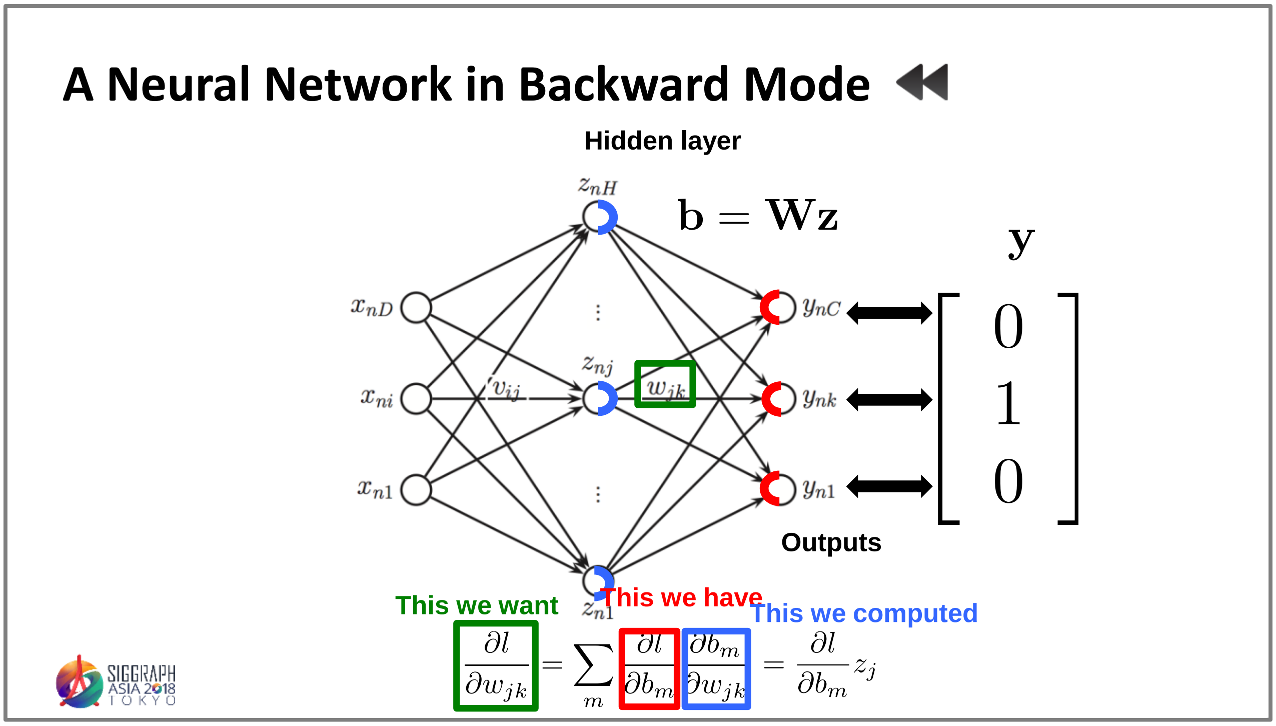
Linear Layer Parameters in Backward: 1-to-1

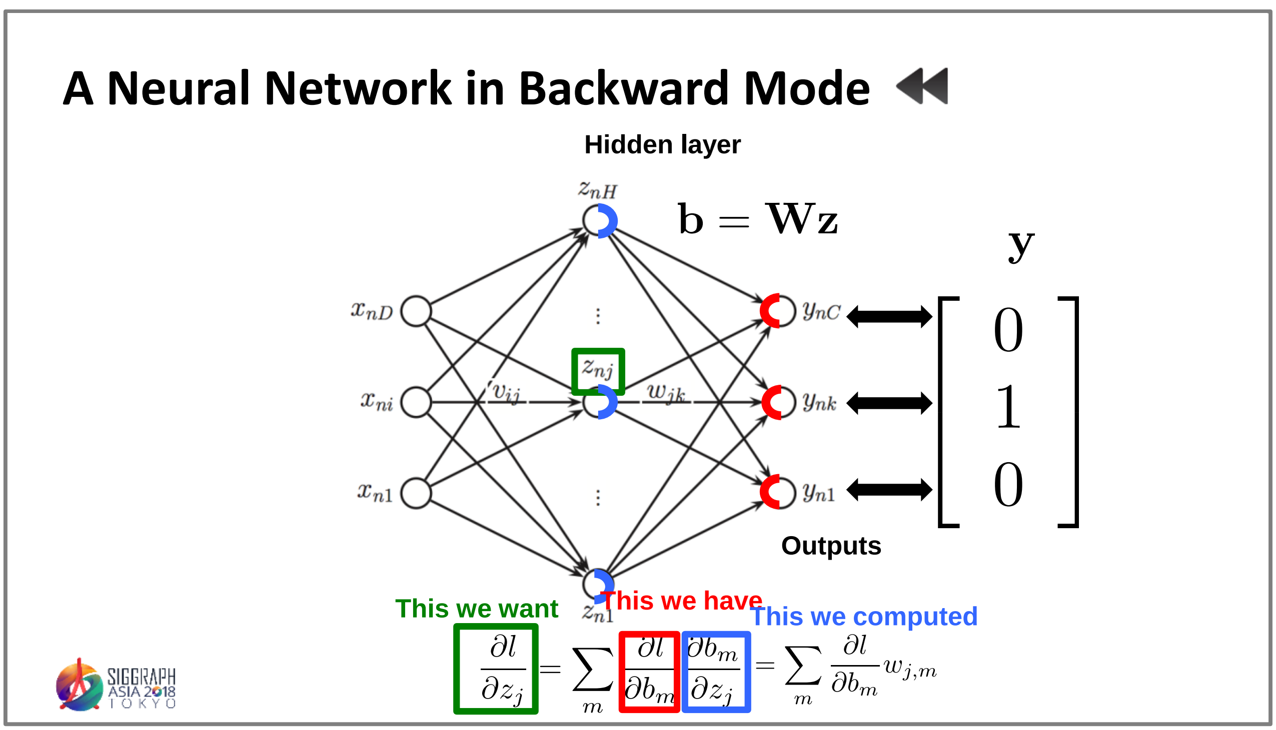
$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$

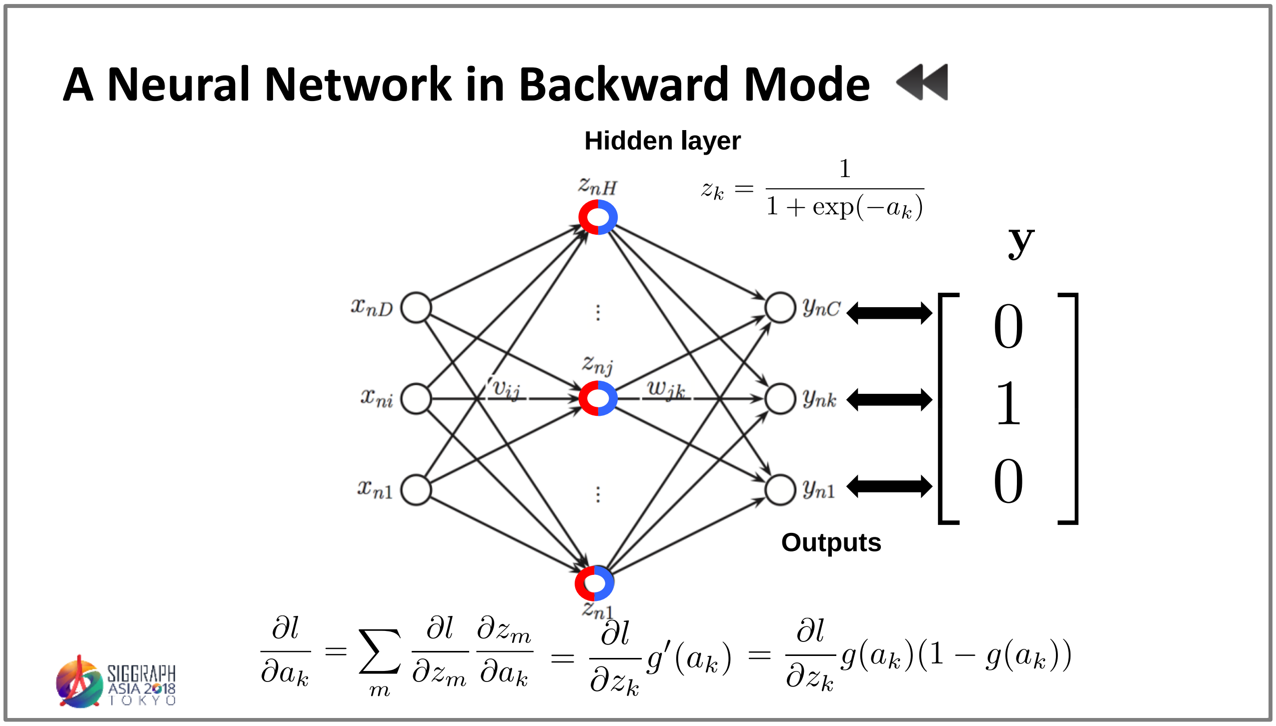


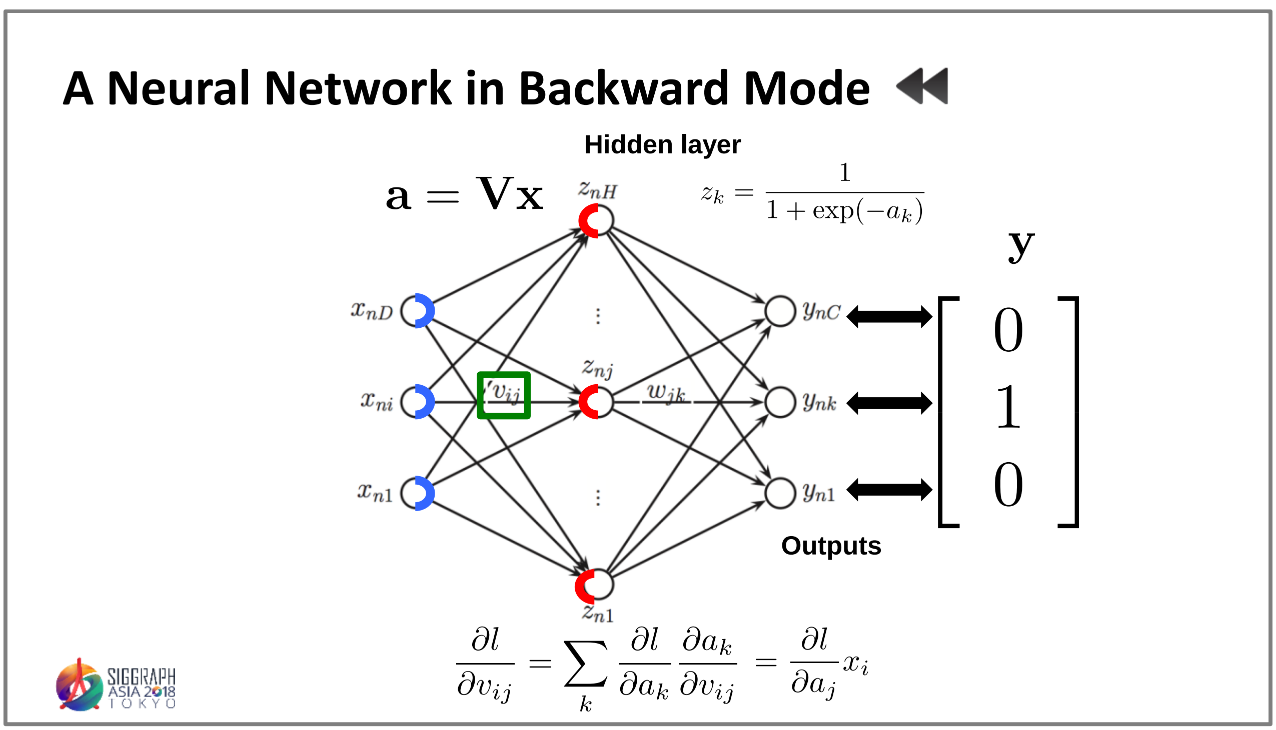
$$\frac{\partial L}{\partial w_{h,m}} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} \cdot \frac{\partial b_c}{\partial w_{h,m}} = \frac{\partial L}{\partial b_m} z_h$$

SIGGRAPH ASIA 2018 TOKYO


A Neural Network in Backward Mode
Hidden layer
z_{nH}
$\mathbf{b} = \mathbf{W}\mathbf{z}$
$\mathbf{y}$
x_{nD}
x_{ni}
x_{n1}
v_{ij}
z_{nj}
w_{jk}
y_{nC}
y_{nk}
y_{n1}
0
1
0
Outputs
z_{n1}
This we want
This we have
This we computed
$\frac{\partial l}{\partial w_{jk}} = \sum_m \frac{\partial l}{\partial b_m} \frac{\partial b_m}{\partial w_{jk}} = \frac{\partial l}{\partial b_m} z_j$
SIGGRAPH ASIA 2018 TOKYO


A Neural Network in Backward Mode
Hidden layer
z_{nH}
$\mathbf{b} = \mathbf{W}\mathbf{z}$
$\mathbf{y}$
x_{nD}
x_{ni}
x_{n1}
v_{ij}
z_{nj}
w_{jk}
y_{nC}
y_{nk}
y_{n1}
0
1
0
Outputs
z_{n1}
This we want
This we have
This we computed
$\frac{\partial l}{\partial z_j} = \sum_m \frac{\partial l}{\partial b_m} \frac{\partial b_m}{\partial z_j} = \sum_m \frac{\partial l}{\partial b_m} w_{j,m}$
SIGGRAPH ASIA 2018 TOKYO


A Neural Network in Backward Mode
Hidden layer
$z_k = \frac{1}{1+\exp(-a_k)}$
$\mathbf{y}$
z_{nH}
z_{nj}
z_{n1}
x_{nD}
x_{ni}
x_{n1}
v_{ij}
w_{jk}
y_{nC}
y_{nk}
y_{n1}
0
1
0
Outputs
$\frac{\partial l}{\partial a_k} = \sum_m \frac{\partial l}{\partial z_m} \frac{\partial z_m}{\partial a_k} = \frac{\partial l}{\partial z_k} g'(a_k) = \frac{\partial l}{\partial z_k} g(a_k)(1-g(a_k))$


A Neural Network in Backward Mode
Hidden layer
$\mathbf{a} = \mathbf{V}\mathbf{x}$
$z_k = \frac{1}{1+\exp(-a_k)}$
$\mathbf{y}$
z_{nH}
z_{nj}
z_{n1}
x_{nD}
x_{ni}
x_{n1}
v_{ij}
w_{jk}
y_{nC}
y_{nk}
y_{n1}
0
1
0
Outputs
$\frac{\partial l}{\partial v_{ij}} = \sum_k \frac{\partial l}{\partial a_k} \frac{\partial a_k}{\partial v_{ij}} = \frac{\partial l}{\partial a_j} x_i$

Neural Network Training: Old & New Tricks

Old:

- **Back-propagation algorithm**
- **Stochastic Gradient Descent, Momentum, "weight decay"**

New: (last 5-6 years)

- **Dropout**
- **ReLUs**
- **Batch Normalization**

Training Objective for N training samples

$$L(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{y}^i, \hat{\mathbf{y}}^i) + \sum_l \lambda_l \sum_{k,m} (\mathbf{W}^l_{k,m})^2$$

Per-sample loss; Per-layer regularization

Gradient descent: $\mathbf{W}_{t+1} = \mathbf{W}_t - \epsilon \nabla_{\mathbf{W}} L(\mathbf{W}_t)$

(l,k,m) element of gradient vector:

$$\frac{\partial L}{\partial \mathbf{W}^l_{k,m}} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial l(\mathbf{y}^i, \hat{\mathbf{y}}^i)}{\partial \mathbf{W}^l_{k,m}} + 2\lambda_l \mathbf{W}^l_{k,m}$$

Back-prop for i-th example

If $N=10^6$, we will need to run back-prop 10^6 times to update **W** once!

Stochastic Gradient Descent (SGD)

Gradient: **Batch:** [1..N]

$$\frac{\partial L}{\partial \mathbf{W}^l_{k,m}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial l(\mathbf{y}^i, \hat{\mathbf{y}}^i)}{\partial \mathbf{W}^l_{k,m}} + 2\lambda_l \mathbf{W}^l_{k,m}$$

Noisy ('Stochastic') Gradient: **Minibatch:** B elements b(1), b(2),…, b(B): sampled from [1,N]

$$\frac{\partial L}{\partial \mathbf{W}^l_{k,m}} \simeq \frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\mathbf{y}^{b(i)}, \hat{\mathbf{y}}^{b(i)})}{\partial \mathbf{W}^l_{k,m}} + 2\lambda_l \mathbf{W}^l_{k,m}$$

Epoch: N samples, N/B batches

SIGGRAPH ASIA 2018 TOKYO

Regularization in SGD: Weight Decay

Gradient: **Batch:** [1..N]

$$\frac{\partial L}{\partial \mathbf{W}^l_{k,m}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial l(\mathbf{y}^i, \hat{\mathbf{y}}^i)}{\partial \mathbf{W}^l_{k,m}} + 2\lambda_l \mathbf{W}^l_{k,m}$$

Noisy ('Stochastic') Gradient: **Minibatch:** B elements b(1), b(2),…, b(B): sampled from [1,N]

$$\frac{\partial L}{\partial \mathbf{W}^l_{k,m}} \simeq \underbrace{\frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\mathbf{y}^{b(i)}, \hat{\mathbf{y}}^{b(i)})}{\partial \mathbf{W}^l_{k,m}}}_{\text{Back-prop on minibatch}} + \underbrace{2\lambda_l \mathbf{W}^l_{k,m}}_{\text{"Weight decay"}}$$

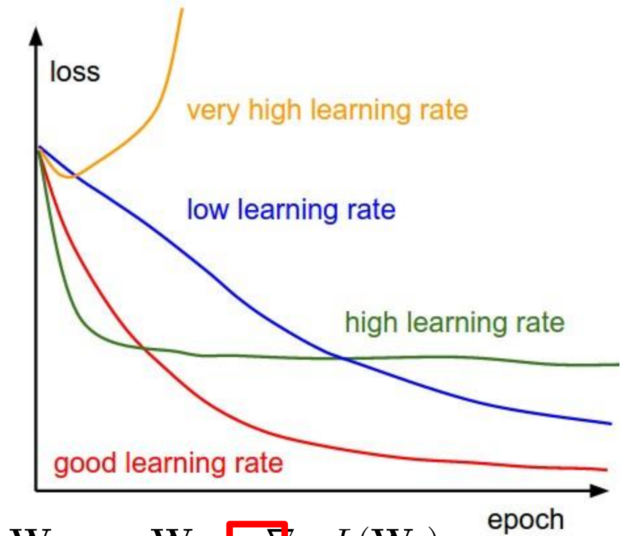
Back-prop on minibatch

"Weight decay"

Epoch: N samples, N/B batches

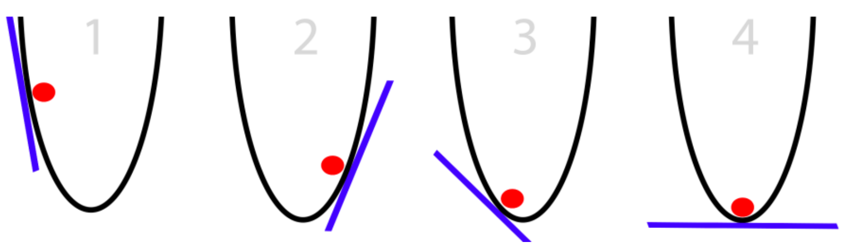
SIGGRAPH ASIA 2018 TOKYO

Learning rate

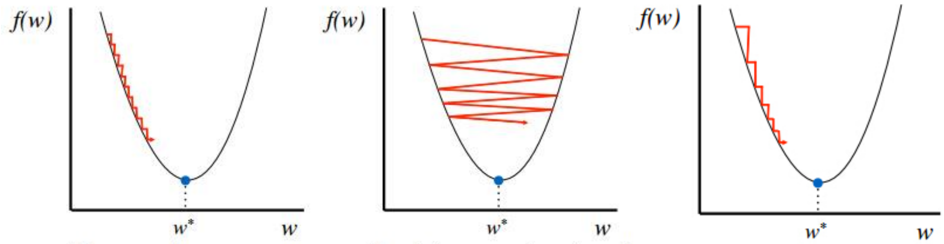


$$\mathbf{W}_{t+1} = \mathbf{W}_t - \epsilon \nabla_{\mathbf{W}} L(\mathbf{W}_t)$$

Gradient Descent



(S)GD with adaptable stepsize



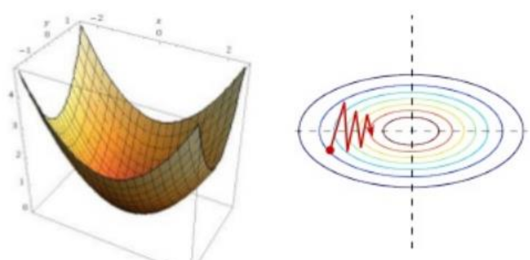
Too small: converge very slowly

Too big: overshoot and even diverge

Reduce size over time

e.g. $\epsilon_t = \frac{c}{t}$

(S)GD with momentum



Main idea: retain long-term trend of updates, drop oscillations

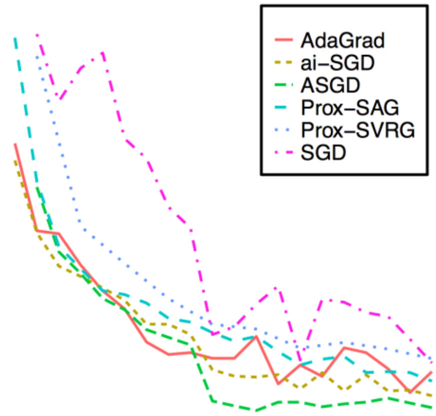
(S)GD $\mathbf{W}_{t+1} = \mathbf{W}_t - \epsilon_t \nabla_{\mathbf{W}} L(\mathbf{W})$

(S)GD + momentum

$$\mathbf{V}_{t+1} = \mu \mathbf{V}_t + (1-\mu) \nabla_{\mathbf{W}} L(\mathbf{W}_t)$$
$$\mathbf{W}_{t+1} = \mathbf{W}_t - \epsilon_t \mathbf{V}_{t+1}$$

Step-size Selection & Optimizers: research problem

- Nesterov's Accelerated Gradient (NAG)
- R-prop
- AdaGrad
- RMSProp
- AdaDelta
- Adam
- …



SIGGRAPH ASIA 2018 TOKYO

Code example

Multi-layer perceptron classification

68

Neural Network Training: Old & New Tricks

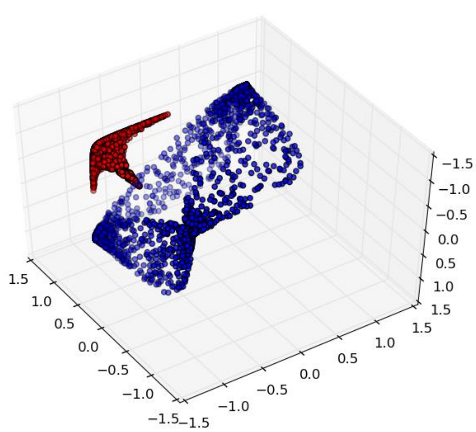
Old: (80's)

Stochastic Gradient Descent, Momentum, "weight decay"

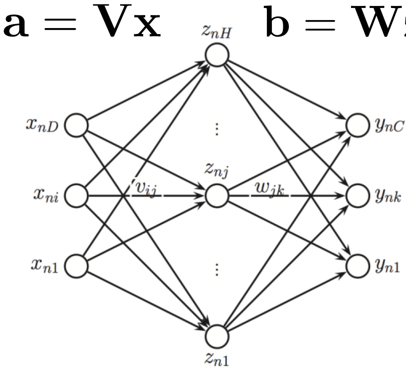
New: (last 5-6 years)

- **Dropout**
- **ReLUs**
- **Batch Normalization**

Linearization: may need higher dimensions

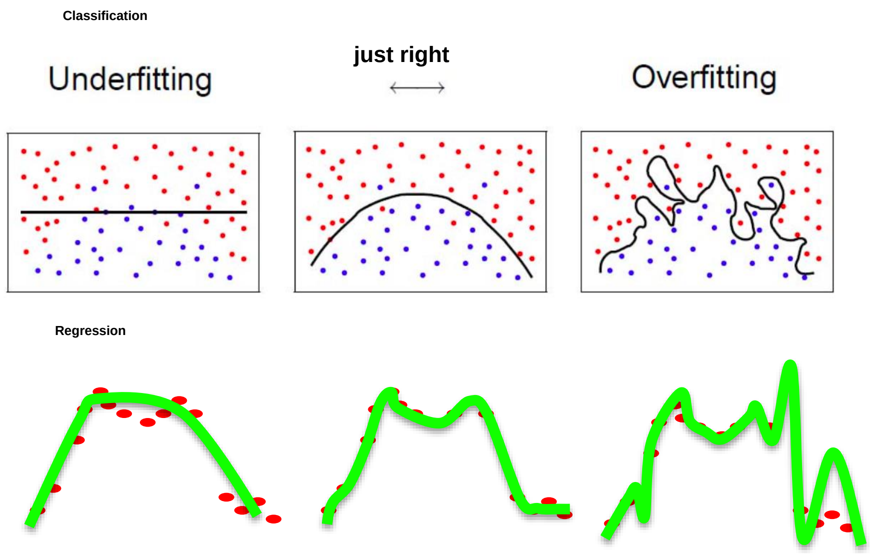


$\mathbf{a} = \mathbf{V}\mathbf{x}$ $\mathbf{b} = \mathbf{W}\mathbf{z}$



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Reminder: Overfitting, in images

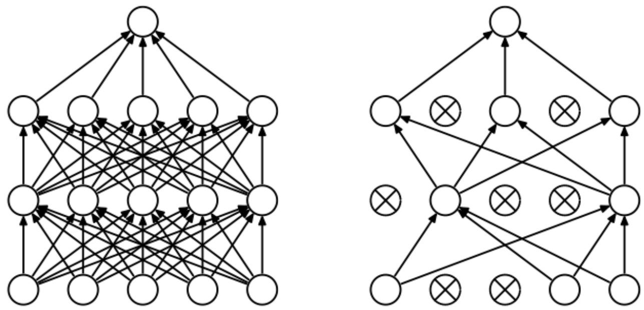


Previously: l2 Regularization

$$L(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{y}^i, \hat{\mathbf{y}}^i) + \sum_{l} \lambda_l \sum_{k,m} (\mathbf{W}^l_{k,m})^2$$

Per-sample loss Per-layer regularization

Dropout



(a) Standard Neural Net (b) After applying dropout.

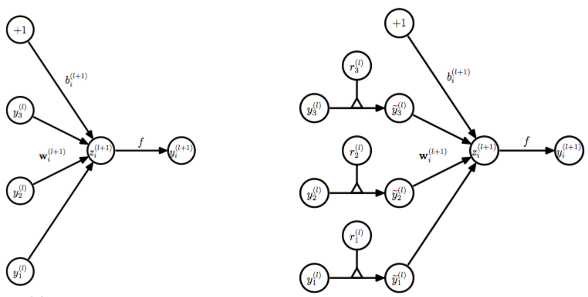
Each sample is processed by a 'decimated' neural net

Decimated nets: distinct classifiers

But: they should all do the same job

SIGGRAPH ASIA 2018 TOKYO

Dropout block



(a) Standard network (b) Dropout network

Figure 3: Comparison of the basic operations of a standard and dropout network.

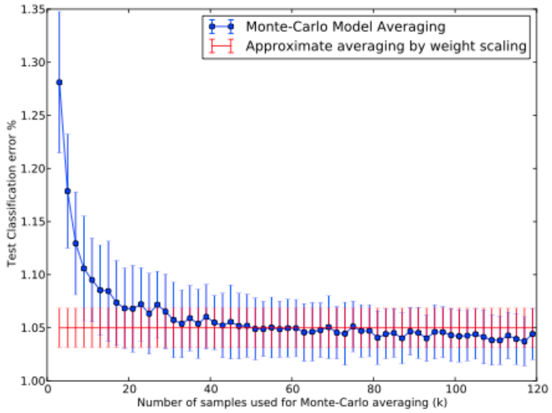
$$
\begin{aligned}
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}),
\end{aligned}
$$

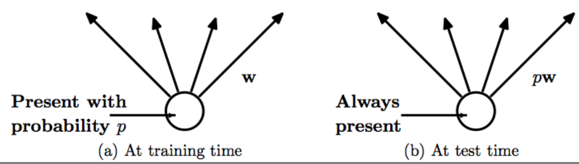
$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

'Feature noising'

SIGGRAPH ASIA 2018 TOKYO

Test time: Deterministic Approximation





Dropout Performance

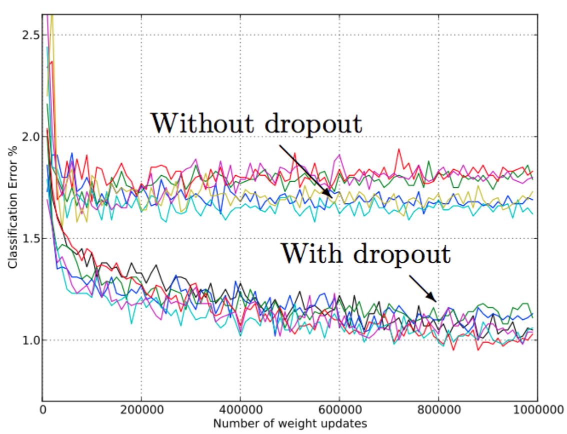


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Neural Network Training: Old & New Tricks

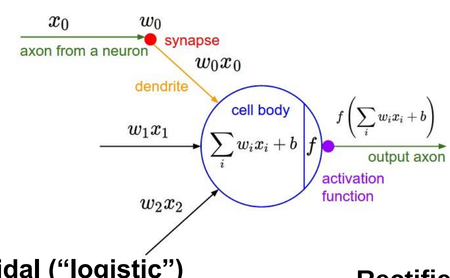
Old: (80's)

Stochastic Gradient Descent, Momentum, "weight decay"

New: (last 5-6 years)

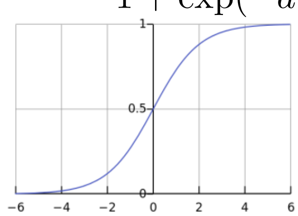
- **Dropout**
- **ReLUs**
- **Batch Normalization**

'Neuron': Cascade of Linear and Nonlinear Function



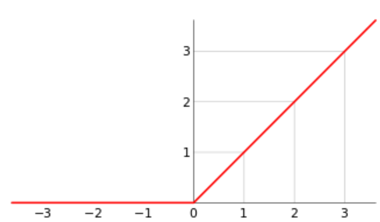
Sigmoidal ("logistic")

$$g(a) = \frac{1}{1+\exp(-a)}$$

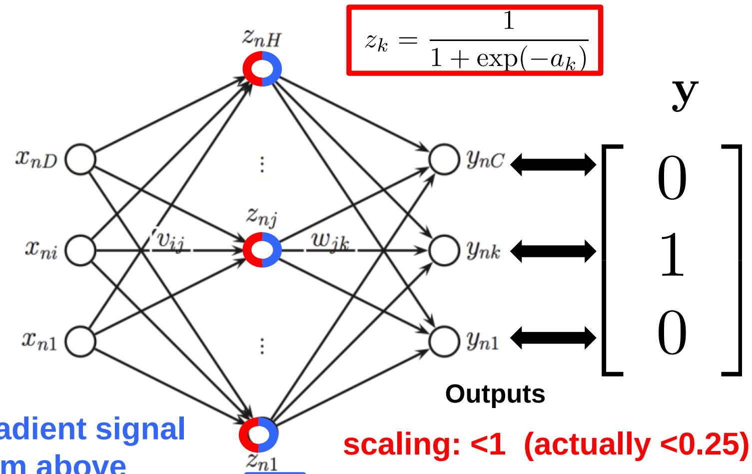


Rectified Linear Unit (RELU)

$$g(a) = \max(0, a)$$



Reminder: a network in backward mode



Gradient signal from above

z_{n1}

scaling: <1 (actually <0.25)

$$\frac{\partial l}{\partial a_k} = \sum_m \frac{\partial l}{\partial z_m}\frac{\partial z_m}{\partial a_k} = \frac{\partial l}{\partial z_k} g'(a_k) = \frac{\partial l}{\partial z_k} g(a_k)(1-g(a_k))$$

Vanishing Gradients Problem

Gradient signal from above

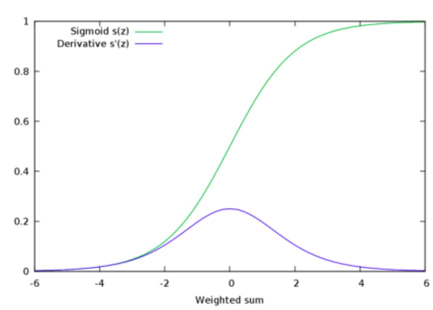
scaling: <1 (actually <0.25)

$$\frac{\partial l}{\partial a_k} = \sum_m \frac{\partial l}{\partial z_m}\frac{\partial z_m}{\partial a_k} = \frac{\partial l}{\partial z_k} g'(a_k) = \frac{\partial l}{\partial z_k} g(a_k)(1-g(a_k))$$

Do this 10 times: updates in the first layers get minimal

Top layer knows what to do, lower layers "don't get it"

Sigmoidal Unit: Signal is not getting through!



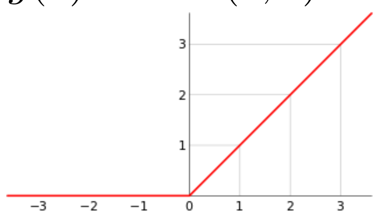
Vanishing Gradients Problem: ReLU Solves It

Gradient signal from above

Scaling: {0,1}

$$\frac{\partial l}{\partial a_k} = \sum_m \frac{\partial l}{\partial z_m}\frac{\partial z_m}{\partial a_k} = \frac{\partial l}{\partial z_k} g'(a_k)$$

$$g(a) = \max(0, a)$$



$$g'(a) = \begin{cases} 1 & a > 0 \\ 0 & a < 0 \end{cases}$$

SIGGRAPH ASIA 2018 TOKYO

Neural Network Training: Old & New Tricks

Old: (80's)

Stochastic Gradient Descent, Momentum, "weight decay"

New: (last 5-6 years)

- **Dropout**
- **ReLUs**
- **Batch Normalization**

SIGGRAPH ASIA 2018 TOKYO

External Covariate Shift: your input changes

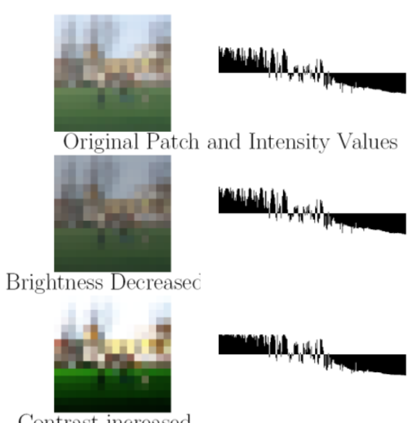
10 am

2pm

7pm

"Whitening": Set Mean = 0, Variance = 1

Photometric transformation: $I \to a\,I + b$



- Make each patch have zero mean:

$$\mu = \frac{1}{N}\sum_{x,y} I(x,y)$$

$$Z(x,y) = I(x,y) - \mu$$

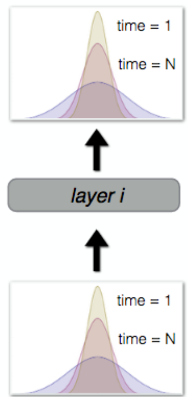
- Then make it have unit variance:

$$\sigma^2 = \frac{1}{N}\sum_{x,y} Z(x,y)^2$$

$$ZN(x,y) = \frac{Z(x,y)}{\sigma}$$

Internal Covariate Shift

Neural network activations during training: moving target



Batch Normalization

Whiten-as-you-go:

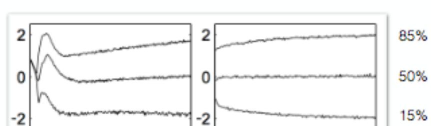
- Normalize the activations in each layer within a mini-batch.
- Learn the mean and variance (γ, β) of each layer as parameters

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

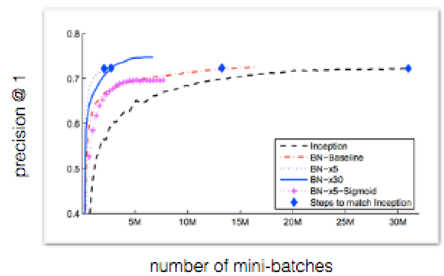


(b) Without BN (c) With BN

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
S Ioffe and C Szegedy (2015)

Batch Normalization: used in all current systems

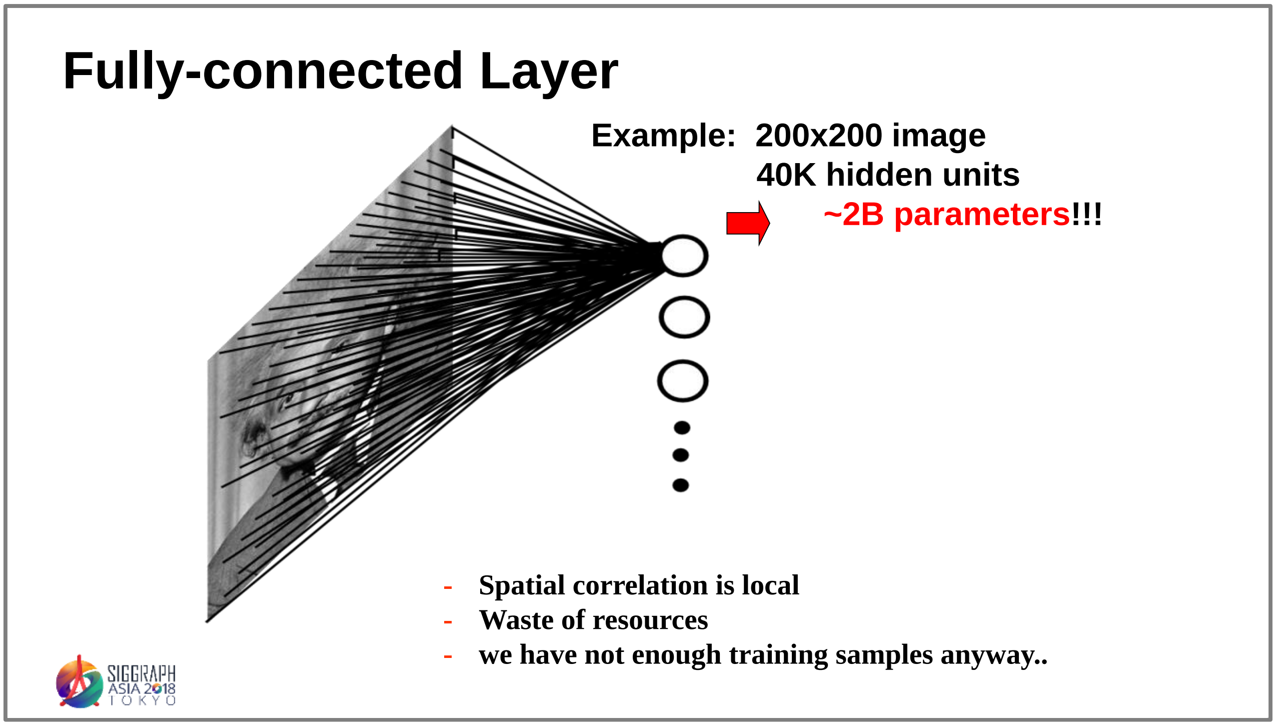
- Multi-layer CNN's train faster with fewer data samples (15x).
- Employ faster learning rates and less network regularizations.
- Achieves state of the art results on ImageNet.

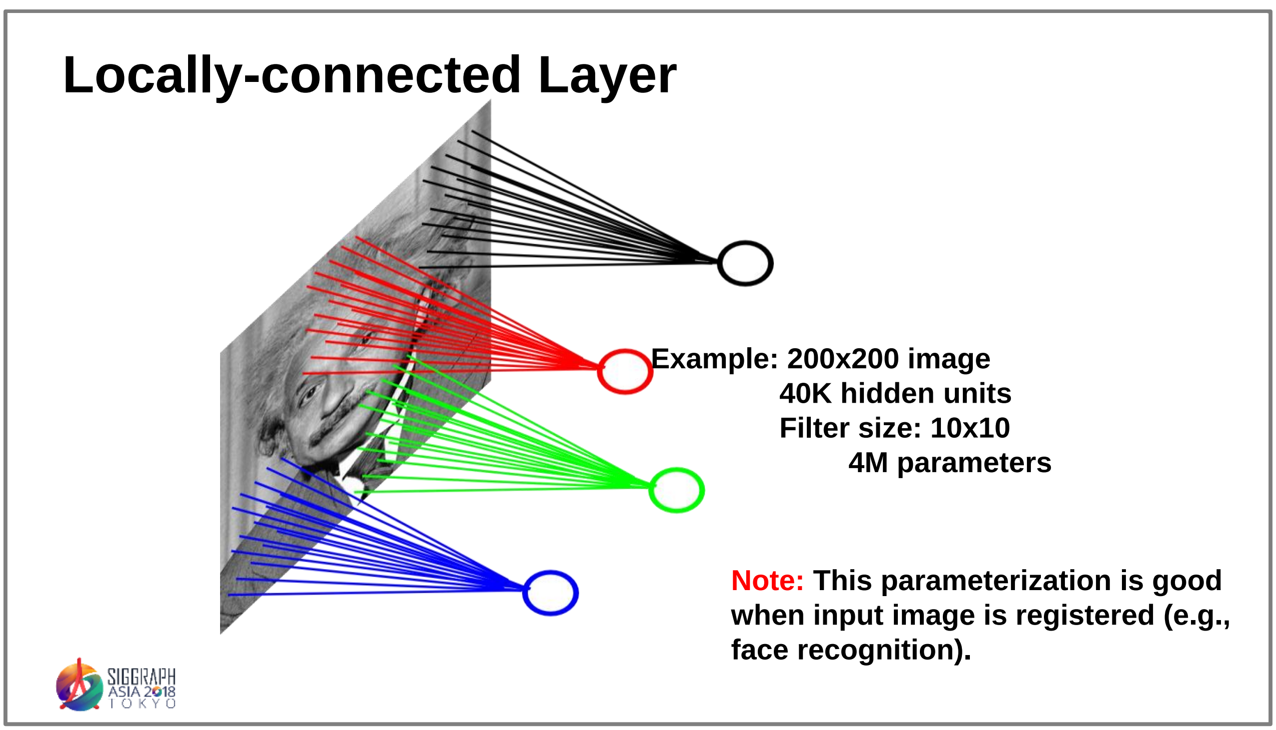


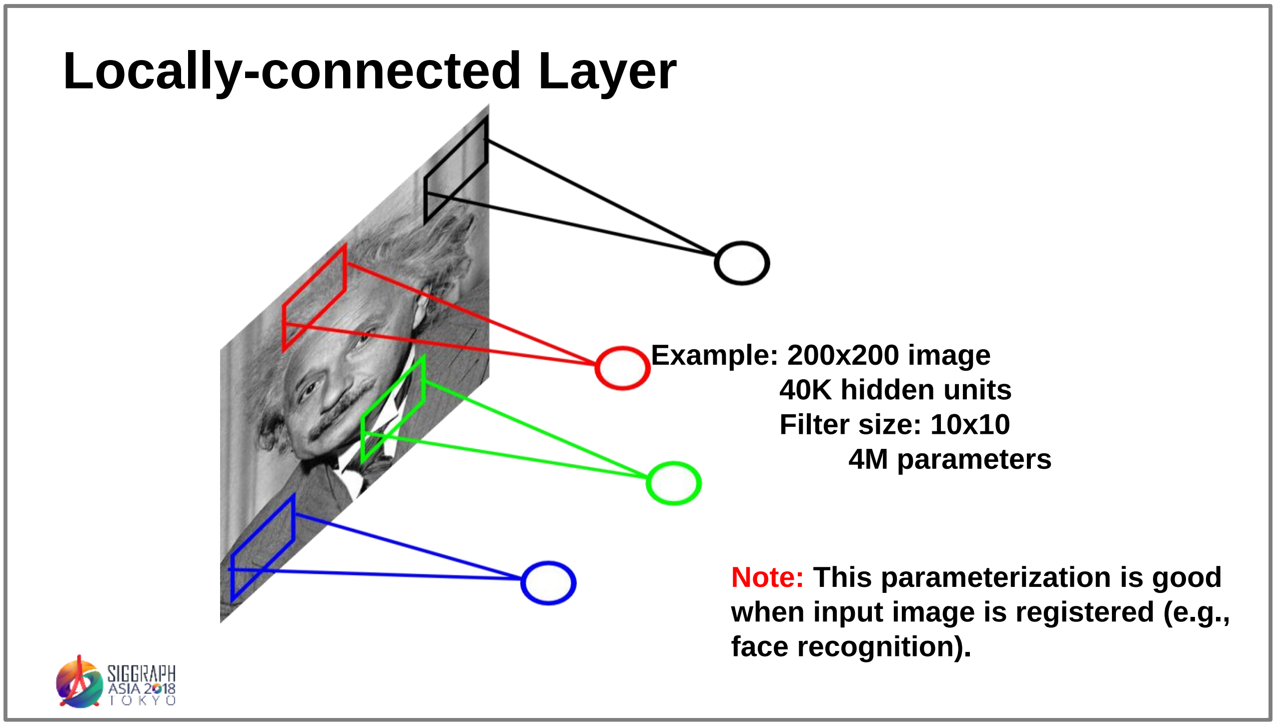
SIGGRAPH ASIA 2018 TOKYO

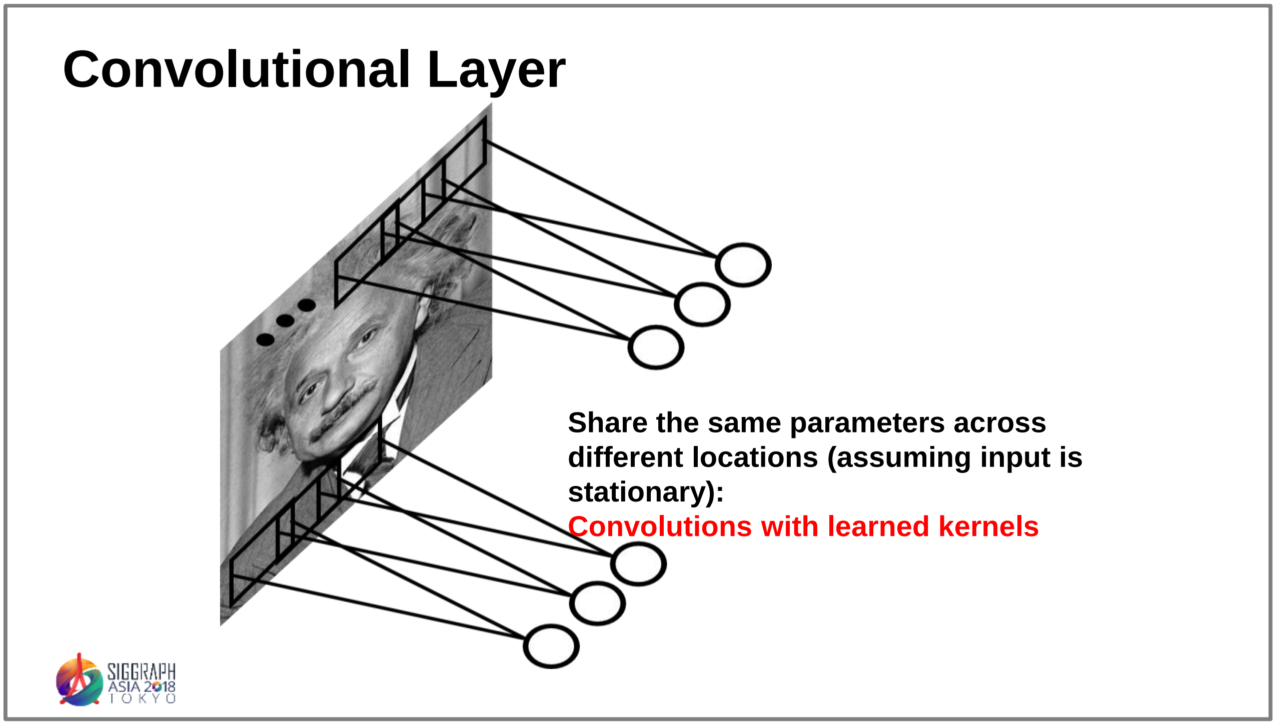
Convolutional Neural Networks

SIGGRAPH ASIA 2018 TOKYO

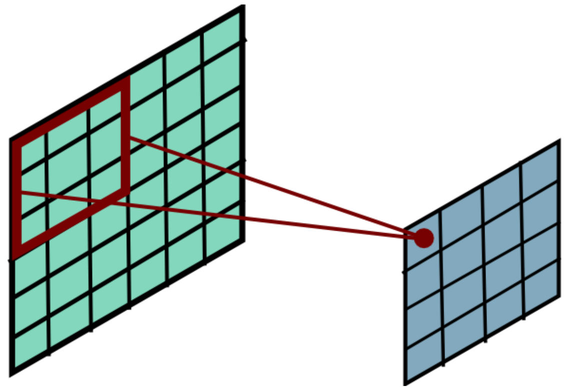

Fully-connected Layer
Example: 200x200 image
40K hidden units
~2B parameters!!!
- Spatial correlation is local
- Waste of resources
- we have not enough training samples anyway..
SIGGRAPH
ASIA 2018
TOKYO


Locally-connected Layer
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters
Note: This parameterization is good when input image is registered (e.g., face recognition).
SIGGRAPH
ASIA 2018
TOKYO

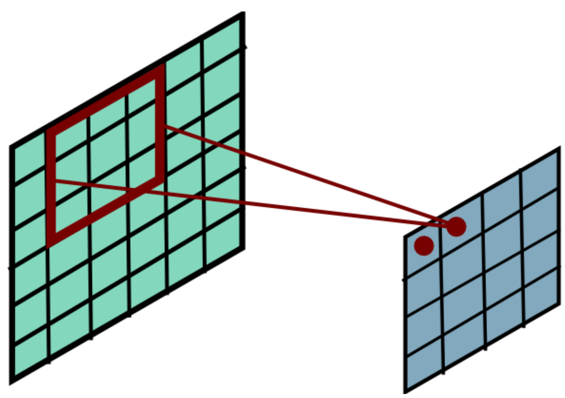

Locally-connected Layer
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters
Note: This parameterization is good when input image is registered (e.g., face recognition).
SIGGRAPH ASIA 2018 TOKYO


Convolutional Layer
Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels
SIGGRAPH ASIA 2018 TOKYO

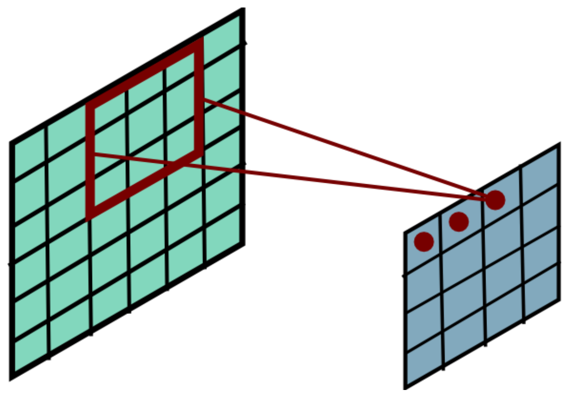
Convolutional Layer



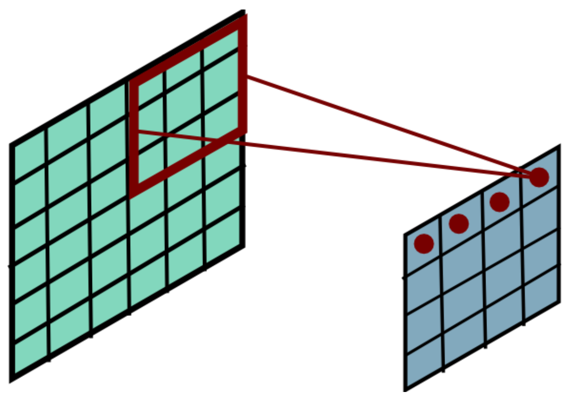
Convolutional Layer



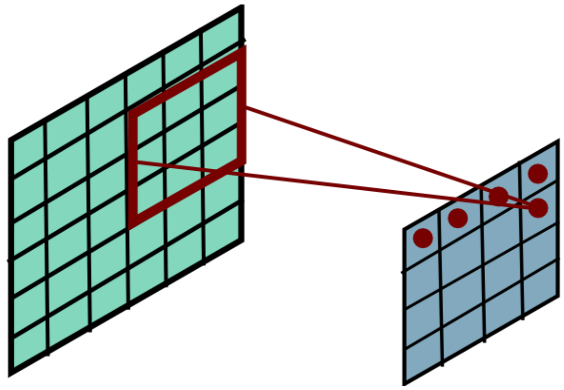
Convolutional Layer



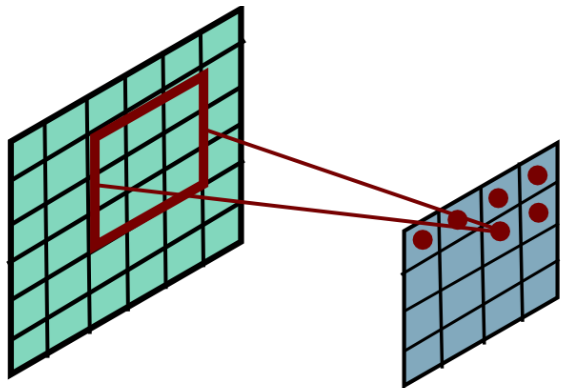
Convolutional Layer



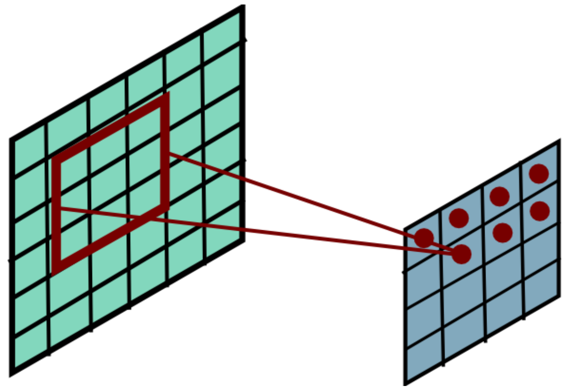
Convolutional Layer



Convolutional Layer

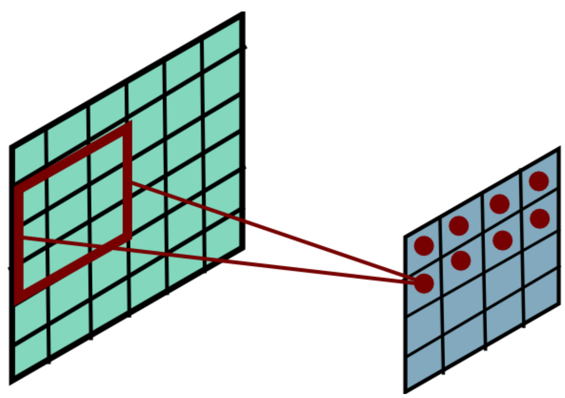


Convolutional Layer



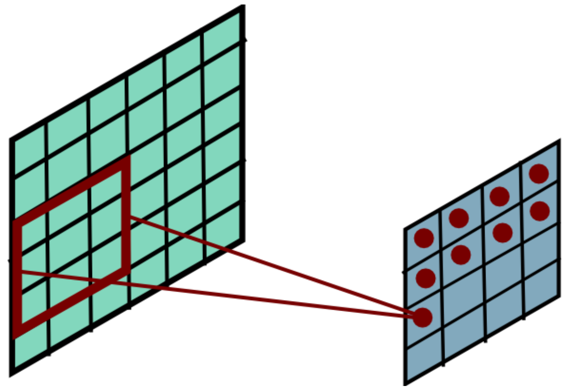
SIGGRAPH ASIA 2018 TOKYO

Convolutional Layer

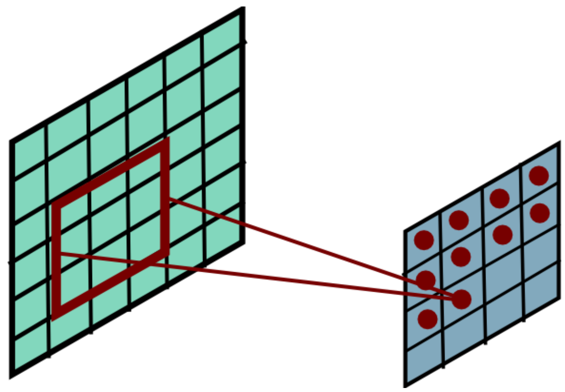


SIGGRAPH ASIA 2018 TOKYO

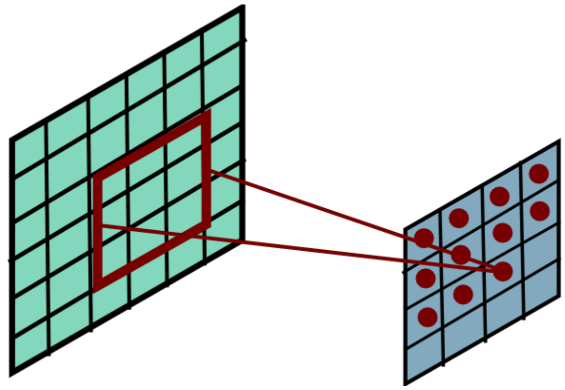
Convolutional Layer



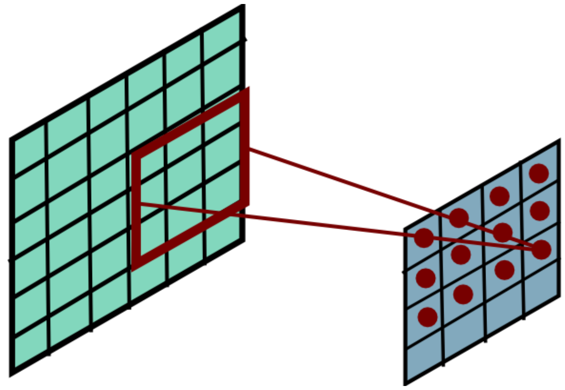
Convolutional Layer



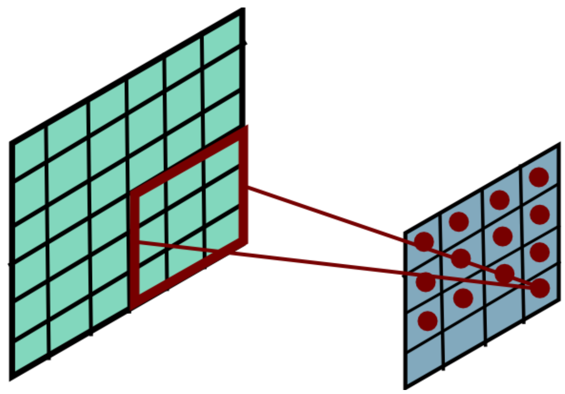
Convolutional Layer



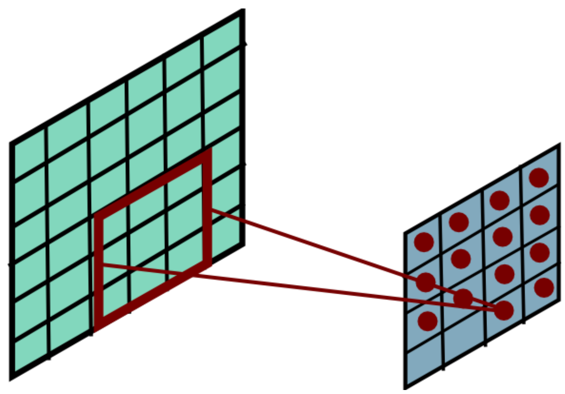
Convolutional Layer



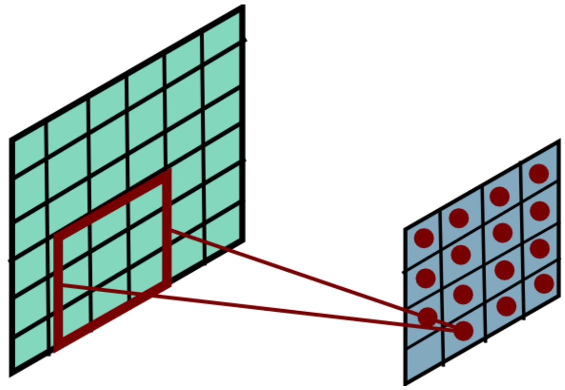
Convolutional Layer



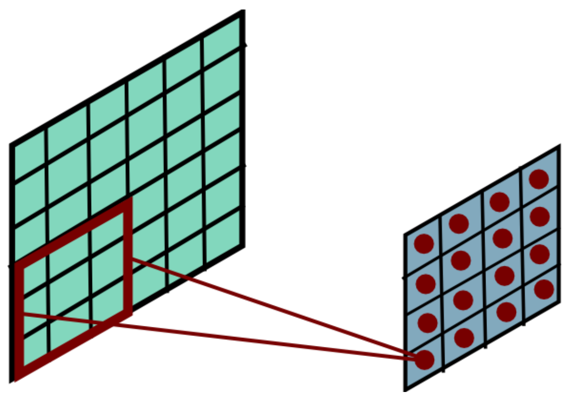
Convolutional Layer



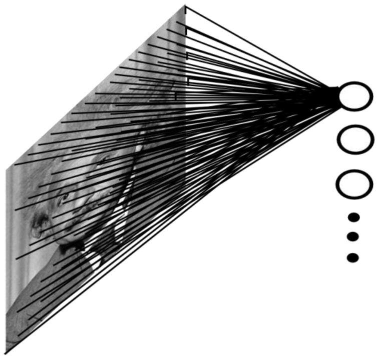
Convolutional Layer



Convolutional Layer



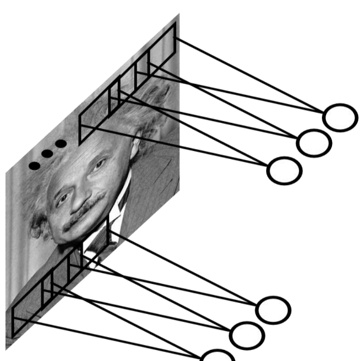
Fully-connected layer



#of parameters: K²

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & \dots & w_{1,K} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & \dots & w_{2,K} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & \dots & w_{3,K} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & \dots & w_{4,K} \\ & & \vdots & & & \\ w_{K,1} & w_{K,2} & w_{K,3} & w_{K,4} & \dots & w_{K,K} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_K \end{bmatrix}$$

Convolutional layer

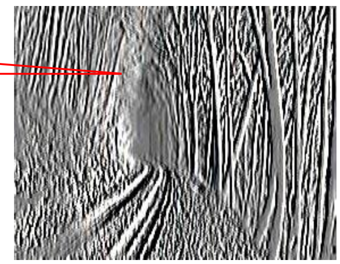


#of parameters: size of window

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} w_0 & w_1 & w_2 & 0 & \dots & 0 \\ 0 & w_0 & w_1 & w_2 & \dots & 0 \\ 0 & 0 & w_0 & w_1 & \dots & 0 \\ 0 & 0 & 0 & w_0 & \dots & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & 0 & \dots & w_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_K \end{bmatrix}$$

Convolutional layer

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$



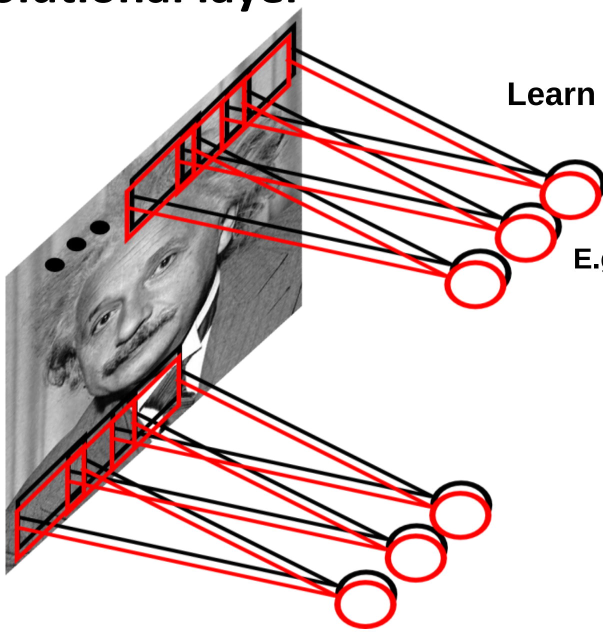
SIGGRAPH ASIA 2018 TOKYO

Code example

Learning an edge filter

112

Convolutional layer



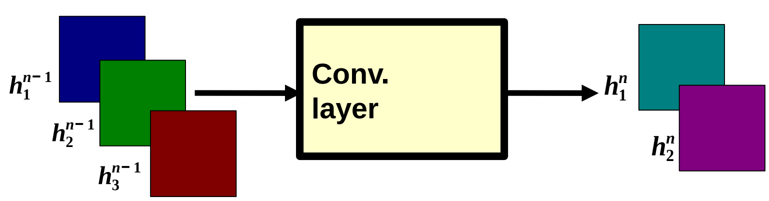
Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Convolutional layer

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

output feature map — input feature map — kernel



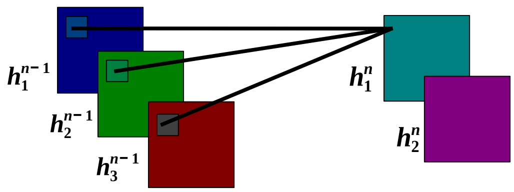
Convolutional layer

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

output feature map

input feature map

kernel



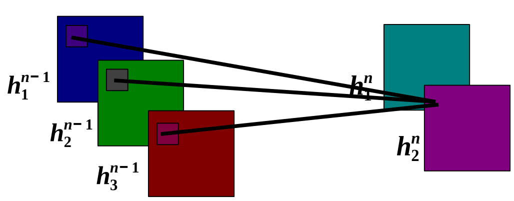
Convolutional layer

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

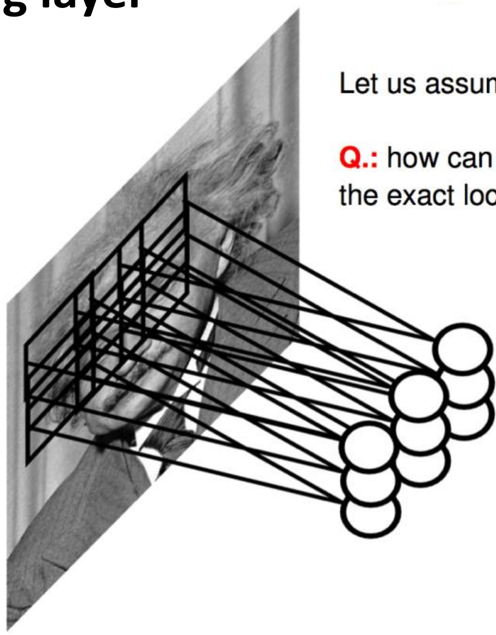
output feature map

input feature map

kernel



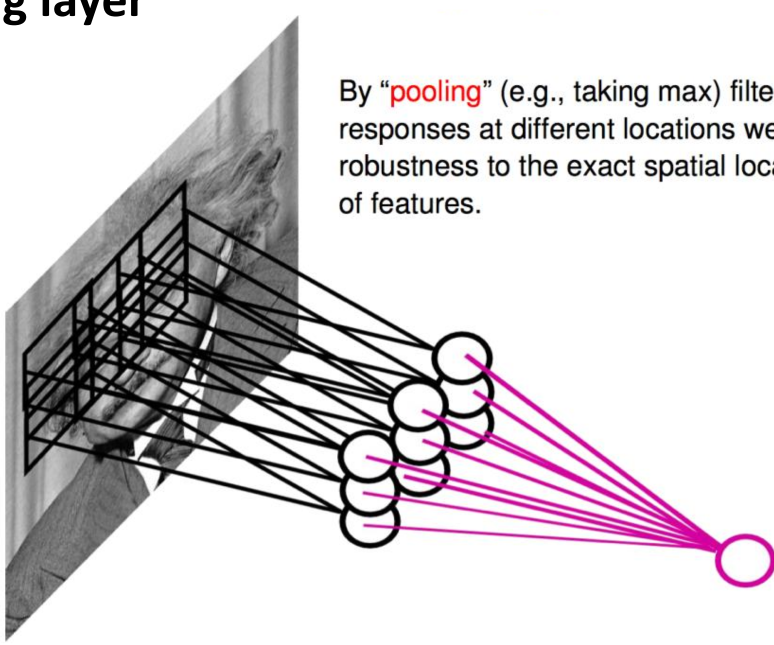
Pooling layer



Let us assume filter is an “eye” detector.

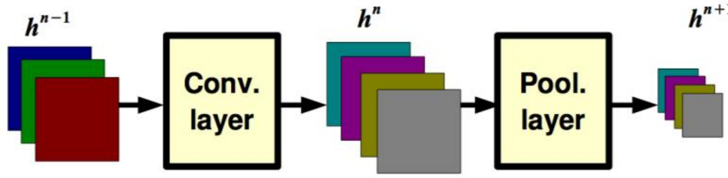
Q.: how can we make the detection robust to the exact location of the eye?

Pooling layer

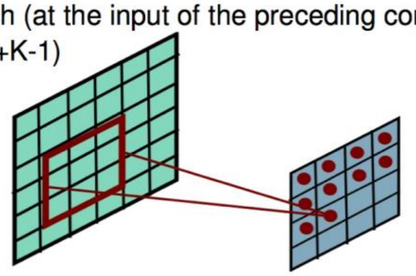


By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

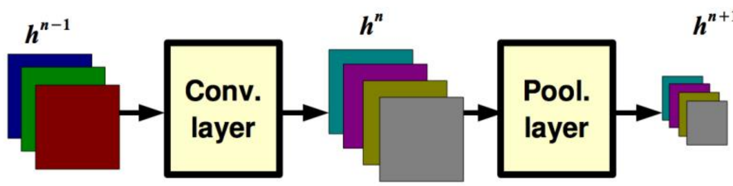
Pooling layer: receptive field size



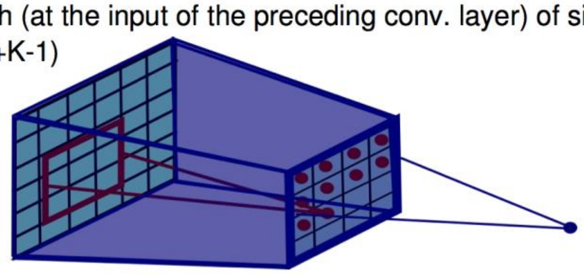
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



Pooling layer: receptive field size



If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)

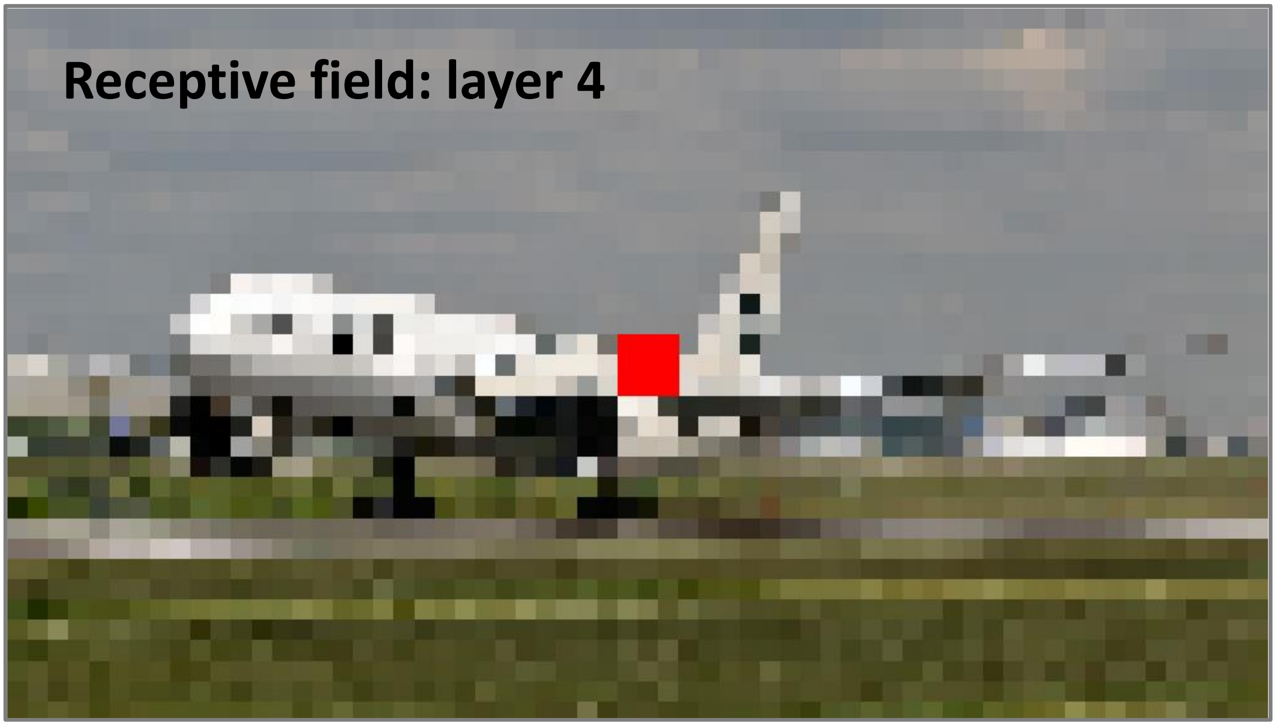


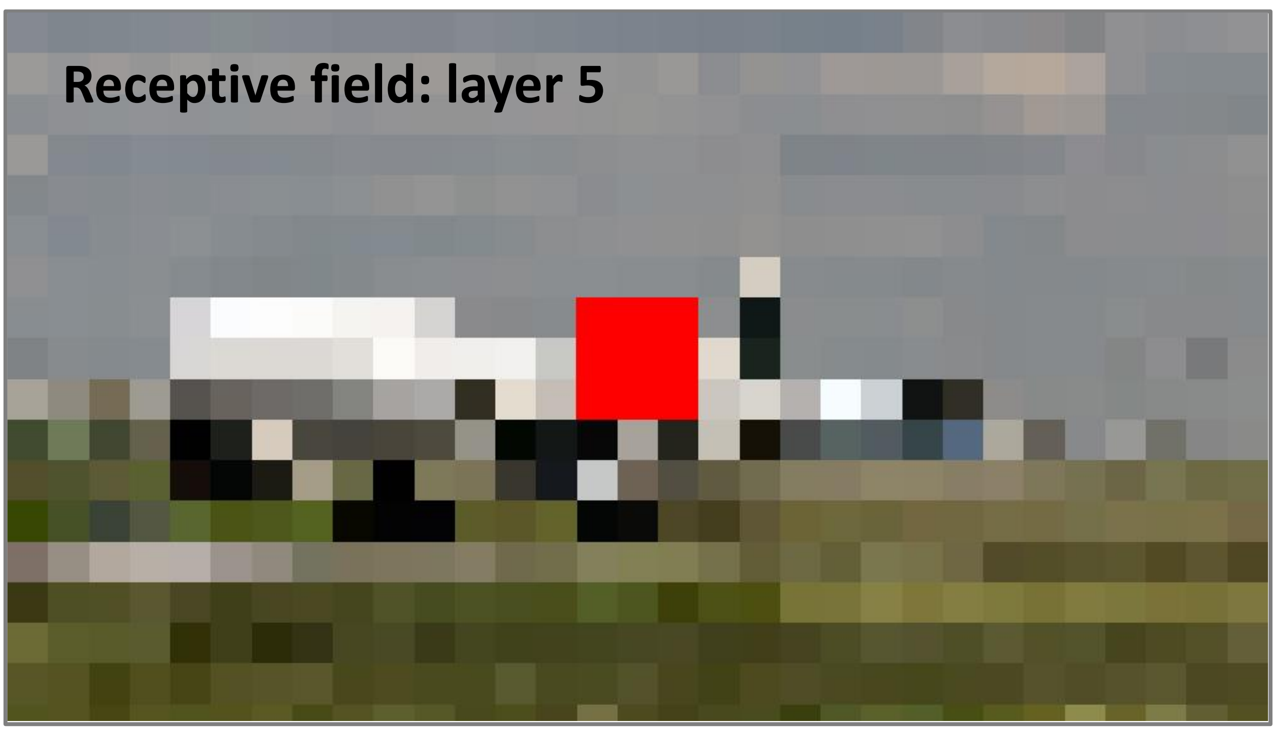

Receptive field
PIA

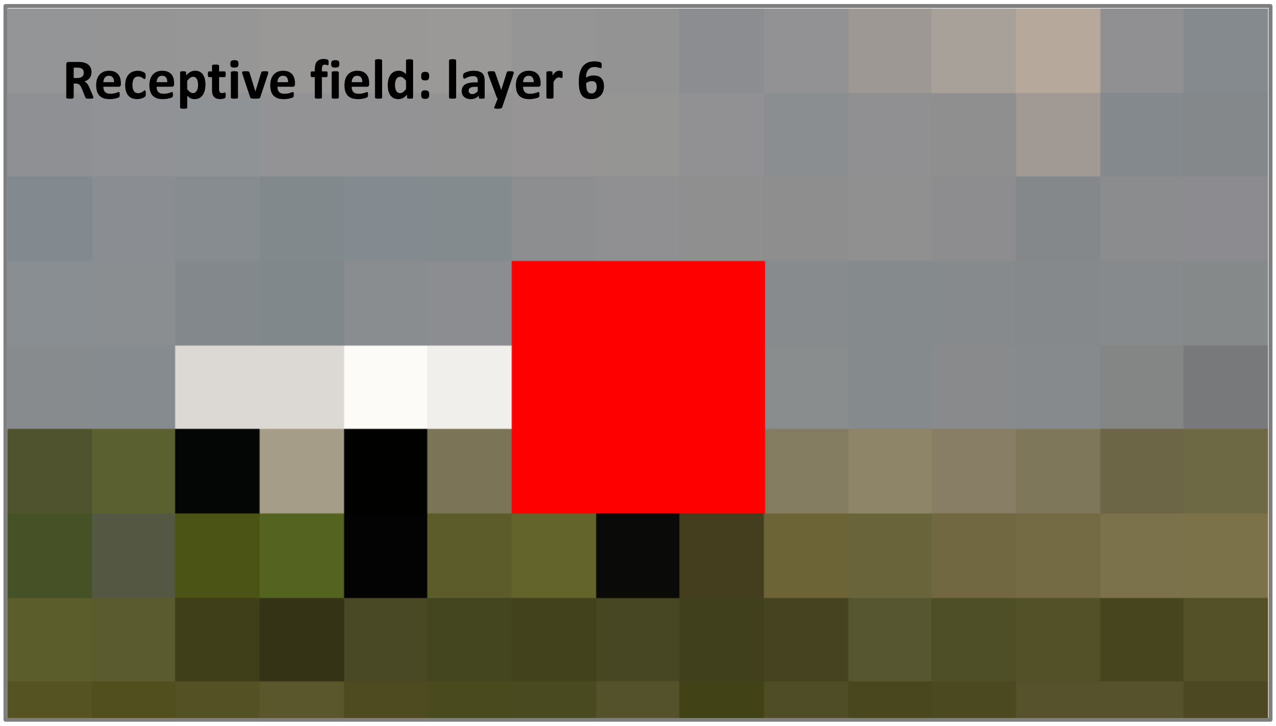

Receptive field: layer 1
PIA

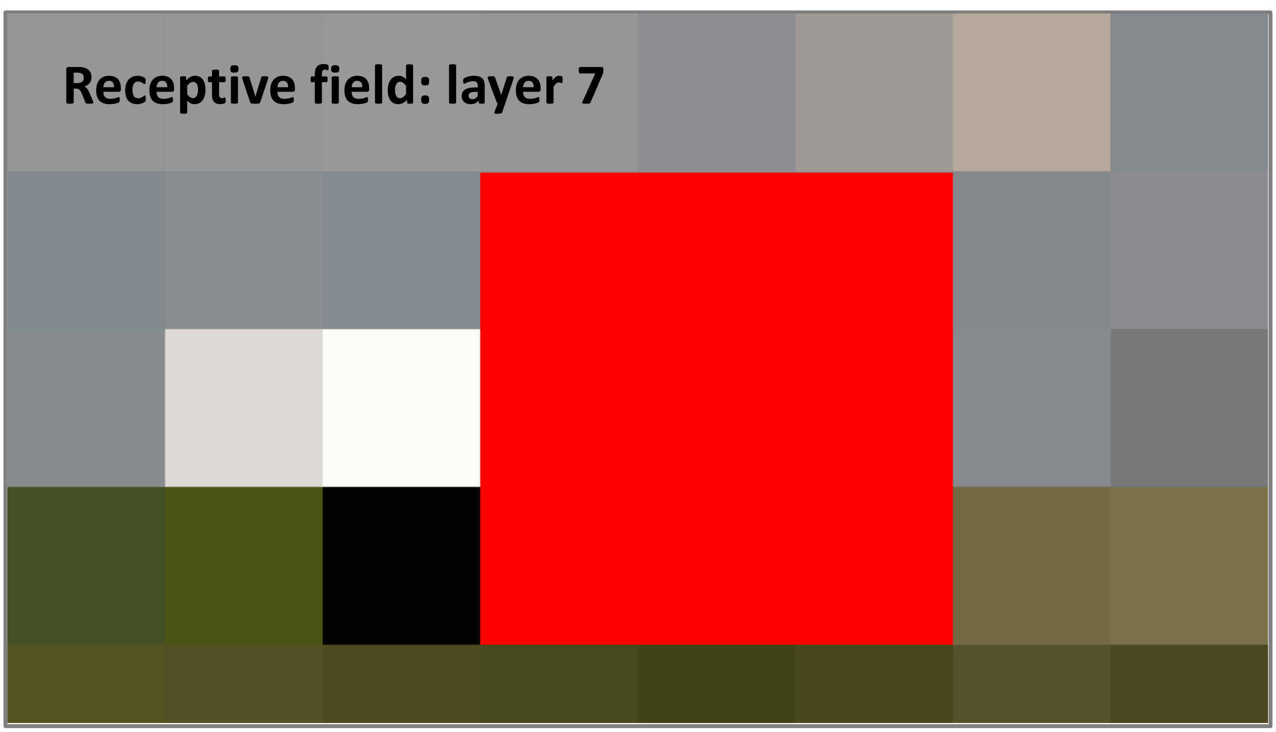

Receptive field: layer 2
PIA


Receptive field: layer 3


Receptive field: layer 4


Receptive field: layer 5

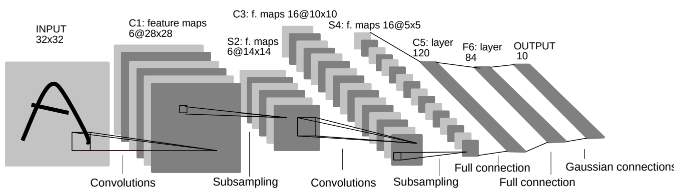

Receptive field: layer 6


Receptive field: layer 7

Receptive field: layer 8

Modern Architectures

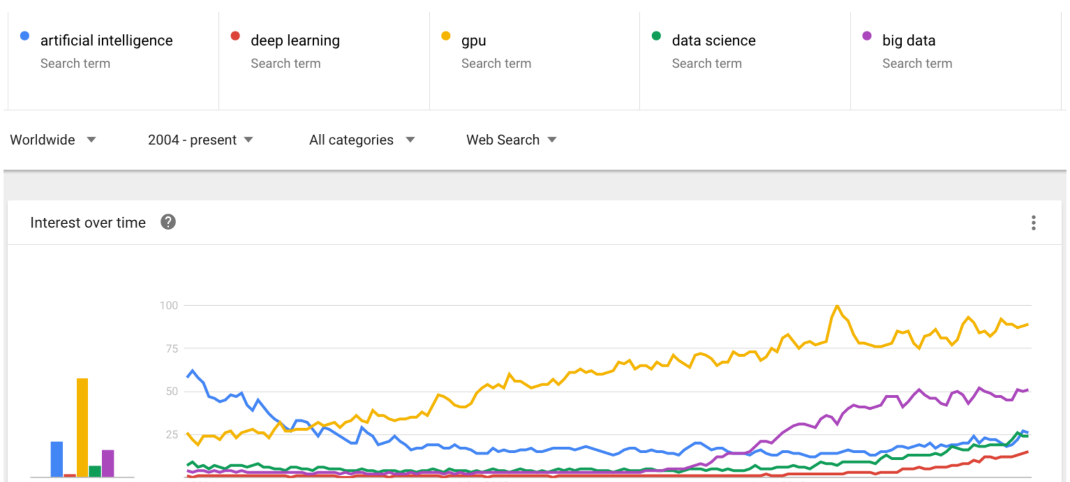
CNNs, late 1980’s: LeNet



https://www.youtube.com/watch?v=FwFduRA_L6Q

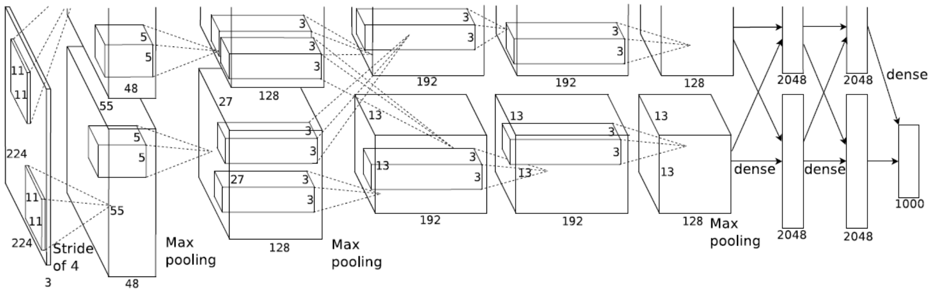
Gradient-based learning applied to document recognition.
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998

What happened in between?



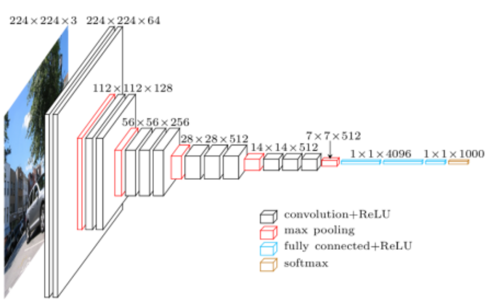
deep learning = neural networks (+ big data + GPUs) + a few more recent tricks!

CNNs, 2012



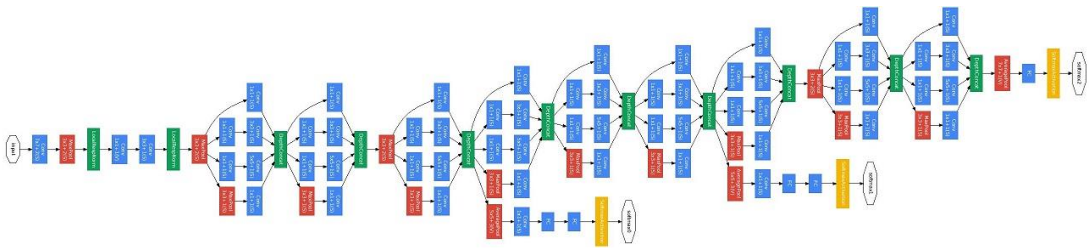
AlexNet
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet classification with deep convolutional neural networks. Commun. ACM 60(6): 84-90 (2017)

CNNs, 2014: VGG



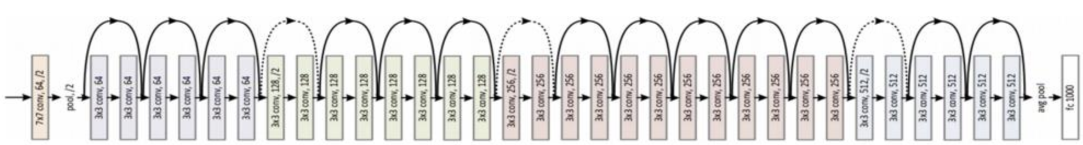
Karen Simonyan, Andrew Zisserman (=Visual Geometry Group)
Very Deep Convolutional Networks for Large-Scale Image Recognition, arxiv, 2014.

CNNs, 2014: GoogLeNet



Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich
Going Deeper with Convolutions, CVPR 2015

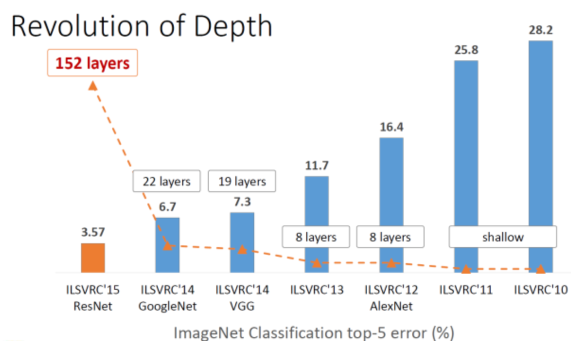
CNNs, 2015: ResNet



ResNet
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,
Deep Residual Learning for Image Recognition
CVPR 2016

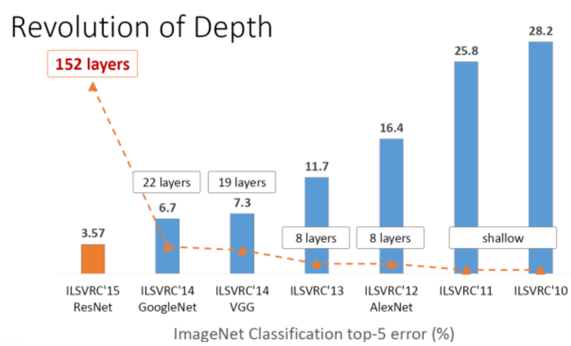
The Deeper, the Better

- Deeper networks can cover more complex problems
  - Increasingly large receptive field size & rich patterns



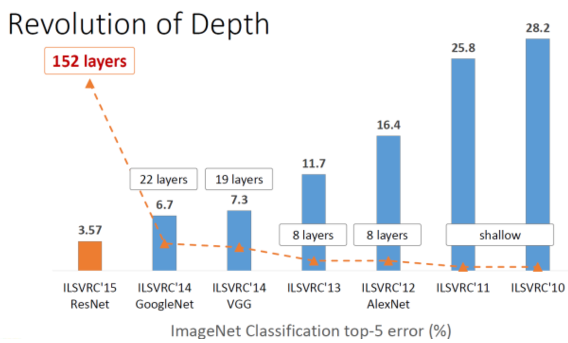
Going Deeper

- From 2 to 10: 2010-2012
  - ReLUs
  - Dropout
  - …



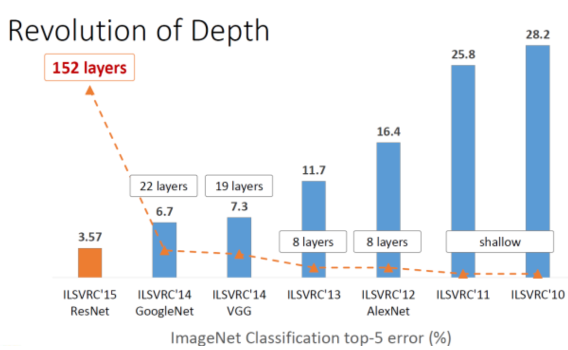
Going Deeper

- From 10 to 20: 2015
  - Batch Normalization



Going Deeper

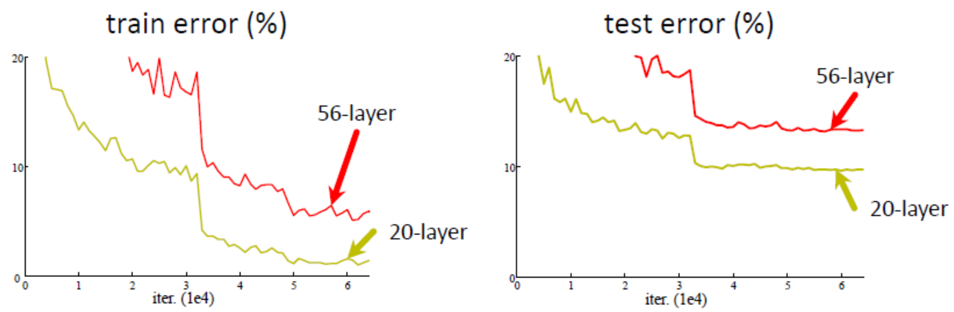
- From 20 to 100/1000
  - Residual networks



Plain network: deeper is not necessarily better

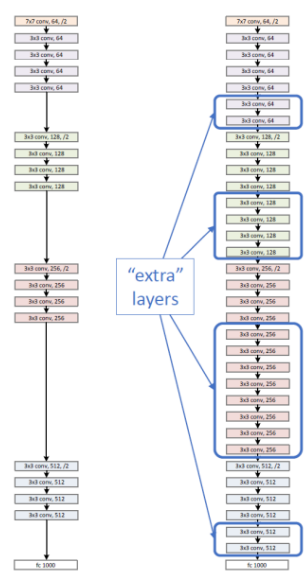
- Plain nets: stacking 3x3 conv layers
- 56-layer net has higher training error and test error than 20-layer net

CIFAR-10



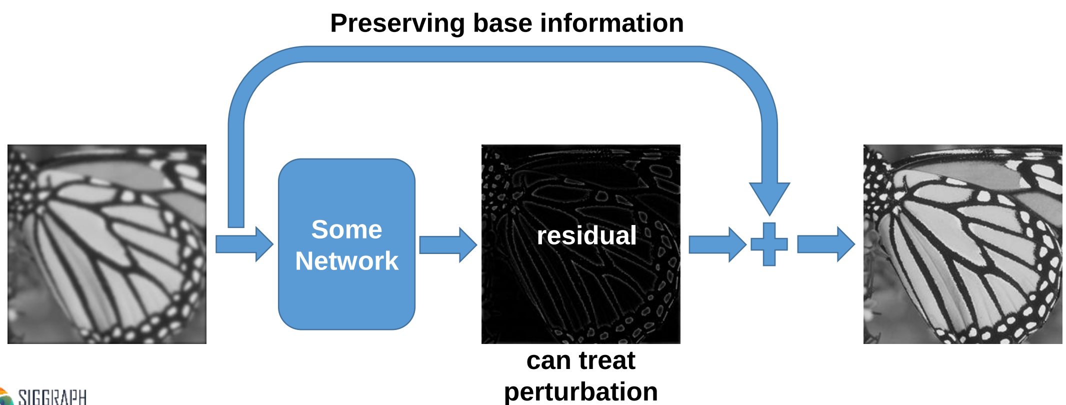
Residual Network

- Naïve solution
  - If extra layers are an identity mapping, then training errors can not increase



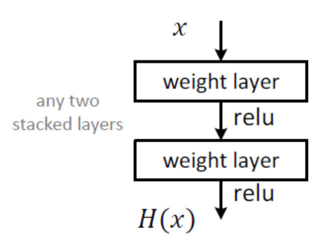
Residual Modelling: Basic Idea in Image Processing

- Goal: estimate update between an original image and a changed image



Residual Network

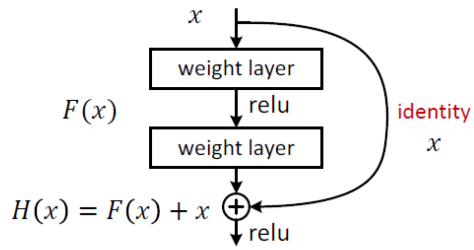
- Plain block
  - Difficult to make identity mapping because of multiple non-linear layers



Residual Network

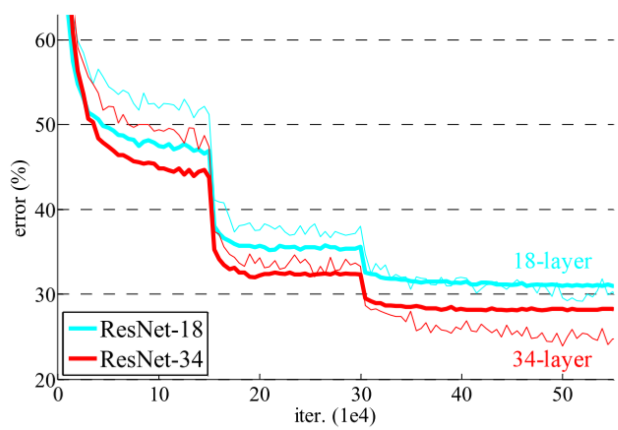
- Residual block
  - If identity were optimal, easy to set weights as 0
  - If optimal mapping is closer to identity, easier to find small fluctuations

Appropriate for treating perturbation as keeping a base information

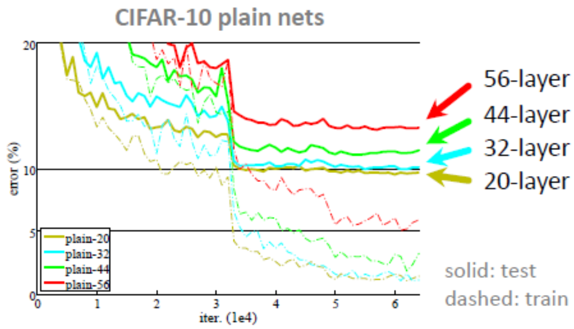


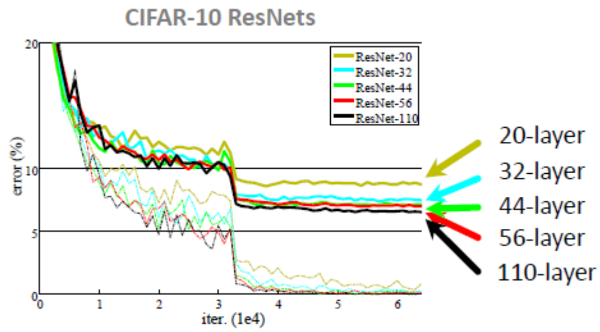
Residual Network: deeper is better

- Deeper ResNets have lower training error



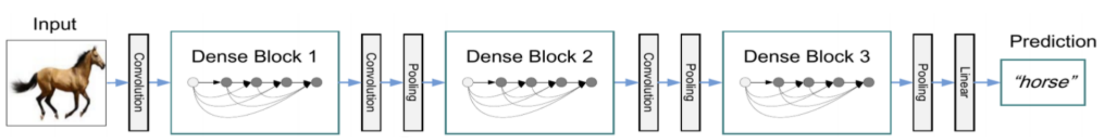
Residual Network: deeper is better



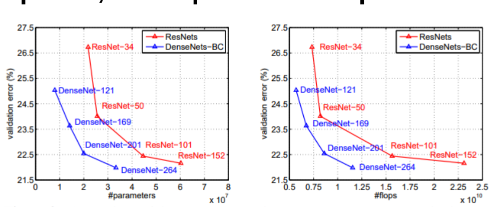


CNNs, 2017: DenseNet

Densely Connected Convolutional Networks, CVPR 2017
Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger



Recently proposed, better performance/parameter ratio



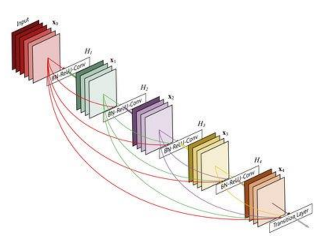
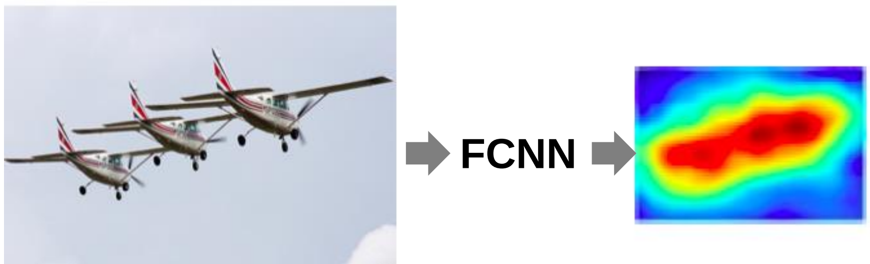


Image-to-Image

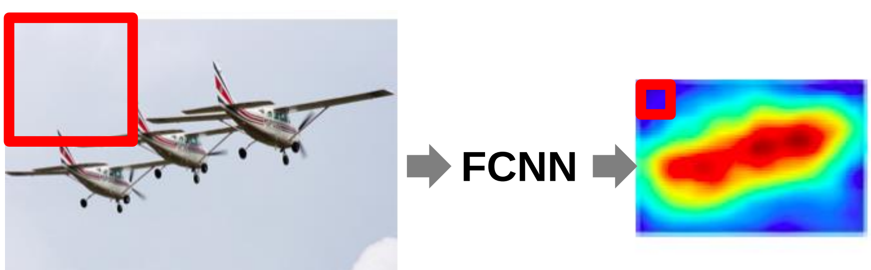
Image-to-image

- So far we mapped an image image to a number or label
- In graphics, output often is “richer”:
  - An image
  - A volume
  - A 3D mesh
  - …
- Architectures
  - Encoder-Decoder
  - Skip connections

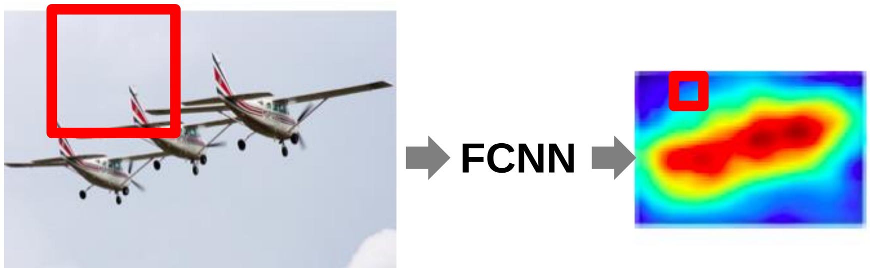
Fully-convolutional Neural Networks



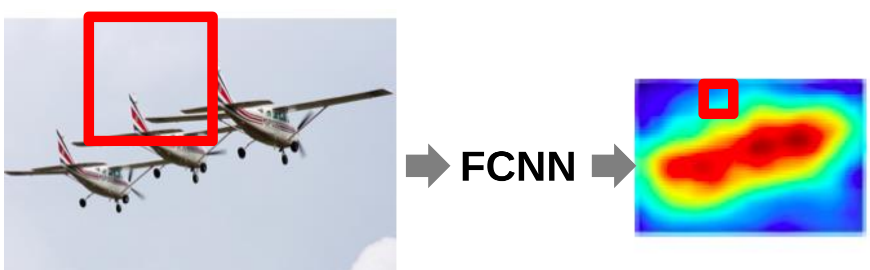
Fully-convolutional Neural Networks



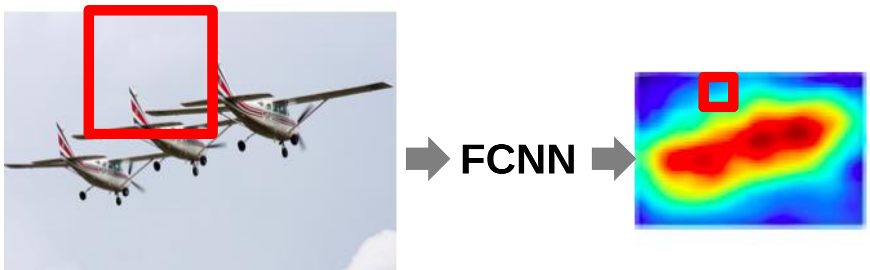
Fully-convolutional Neural Networks



Fully-convolutional Neural Networks

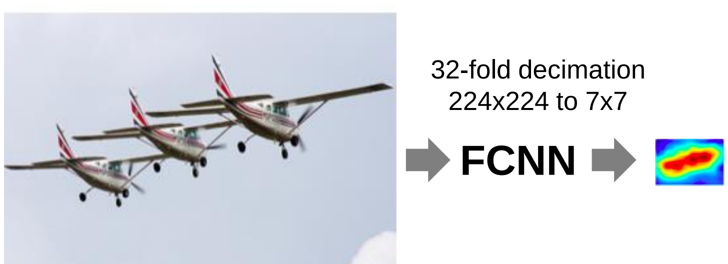


Fully-convolutional Neural Networks



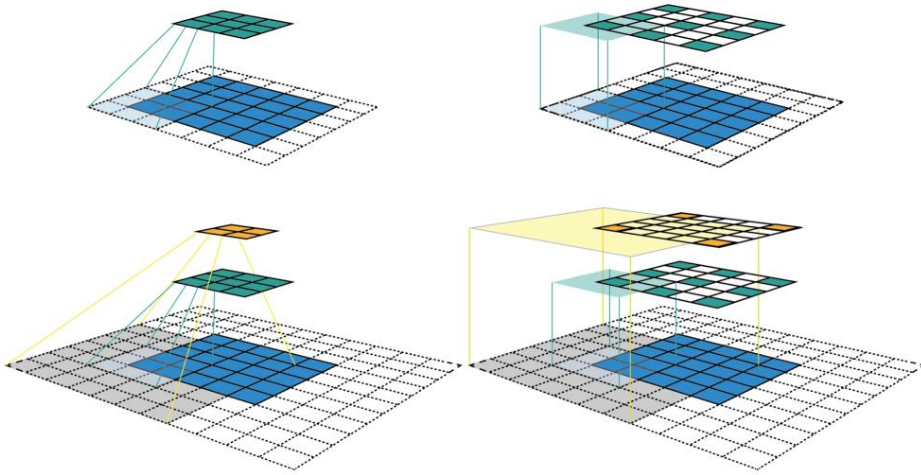
Fast (shared convolutions)
Simple (dense)

Fully Convolutional Neural Networks in Practice



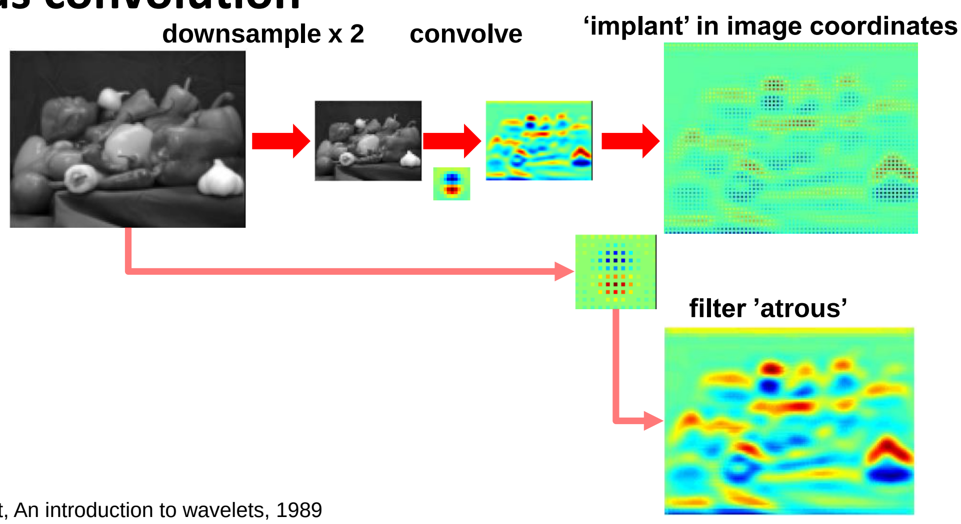
Fast (shared convolutions)
Simple (dense)
Low resolution

Receptive field arithmetic



https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

Atrous convolution



S. Mallat, An introduction to wavelets, 1989
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs
Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille

Atrous convolution = Dilated Convolution

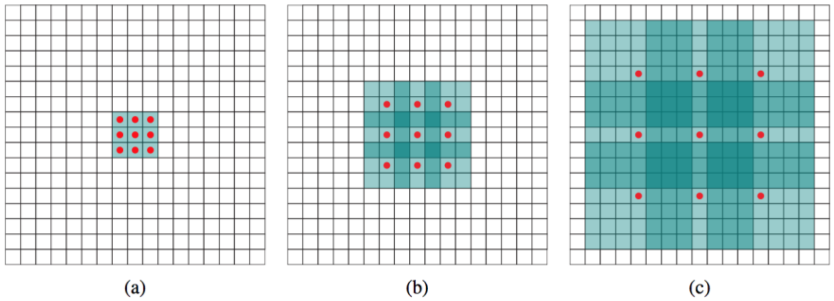
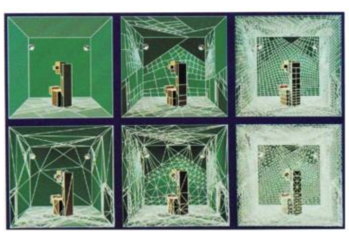


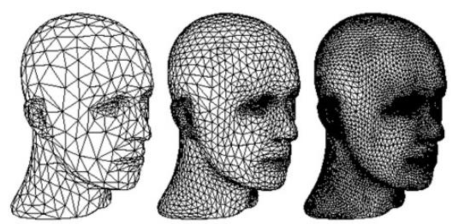
Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

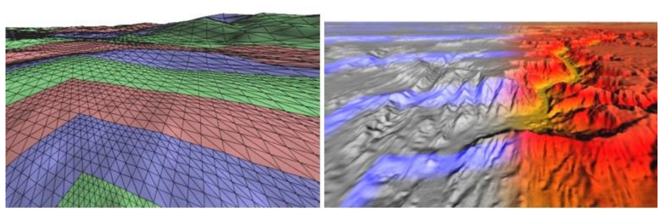
F. Yu, V. Koltun, Multi-Scale Context Aggregation by Dilated Convolutions, ICLR 2016

SIGGRAPH ASIA 2018 TOKYO

Graphics: Multiresolution

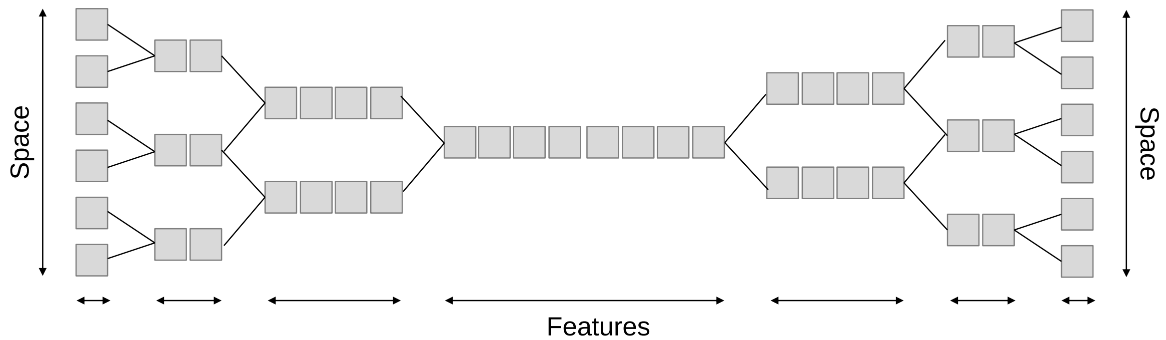






SIGGRAPH ASIA 2018 TOKYO

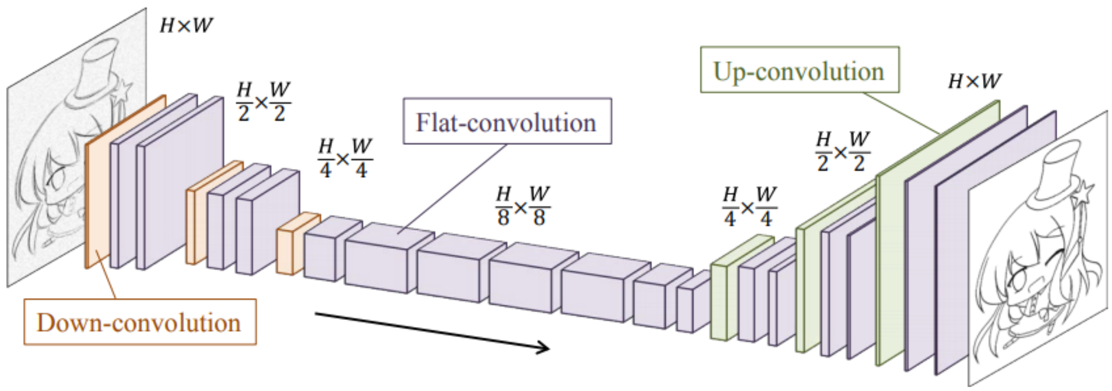
Encoder-decoder



Interpretation

- Turns image into vector
- This vector is a very compact and abstract “code”
- Turns code back into image

Encoder-decoder



Learning to simplify. Simo-Serra et al. 2016

SIGGRAPH ASIA 2018 TOKYO

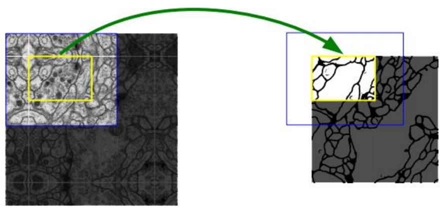
Up-sampling

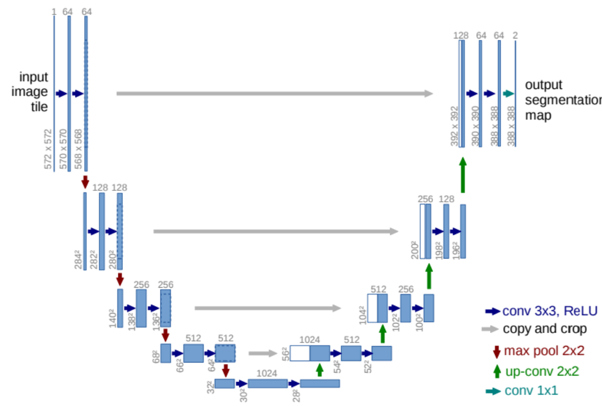
- We saw
  - … how to keep resolution
  - … how to reduce it with pooling
- But how to increase it again?
- Options
  - Interpolation
  - Padding (insert zeros)
  - Transpose convolutions

SIGGRAPH ASIA 2018 TOKYO

Encoder-decoder + Skip connections

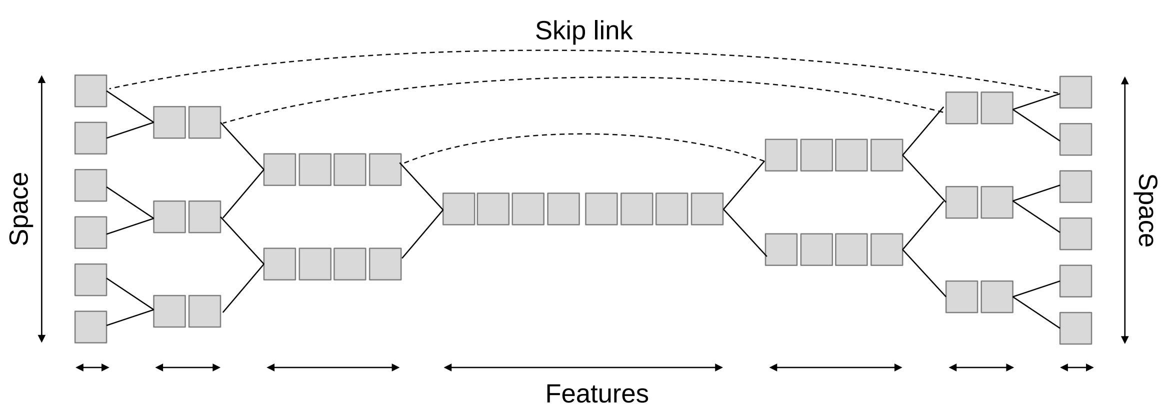
- 1st: Reduce resolutions as before
- 2nd: Increase resolution
- Transposed convolutions





U-Net: Convolutional Networks for Biomedical Image Segmentatio. Ronneberger et al. 2015

Encoder-decoder with skip connections



Interpretation

- Turns image into vector
- Turns vector back into image
- At every step of increasing the resolution, check back with the input to preserve details
- Familiar trick to graphics people
  - (Haar) wavelet
  - Residual coding
  - Pyramidal schemes (Laplacian pyramid, etc.)

Deep Learning Frameworks

Main frameworks

TensorFlow™
(Python, C++, Java)

Keras
(Python, backends support other languages)

PYTORCH
(Python)

Caffe
(C++, Python, Matlab)

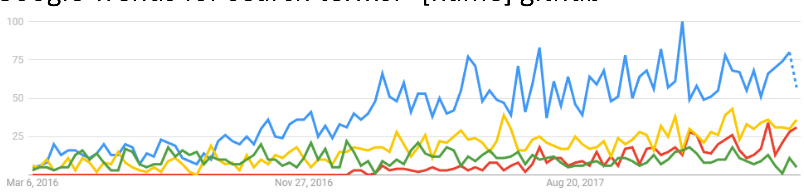
Currently less frequently used

Chainer (Python)
theano (Python)
Caffe2 (Python, C++)
Microsoft CNTK (Python, C++, C#)
MATLAB (Matlab)
DL4J (Python, Java, Scala)
mxnet (Python, C++, and others)
…

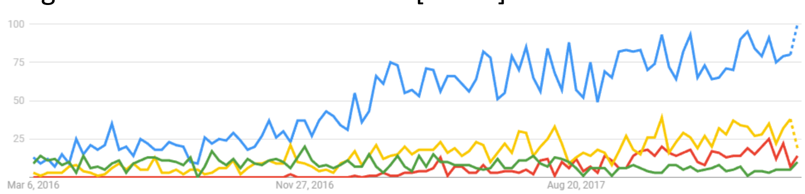
SIGGRAPH ASIA 2018 TOKYO

Popularity

Google Trends for search terms: "[name] github"



Google Trends for search terms: "[name] tutorial"



TensorFlow
Keras
PYTORCH
Caffe

SIGGRAPH ASIA 2018 TOKYO

Typical Training Steps

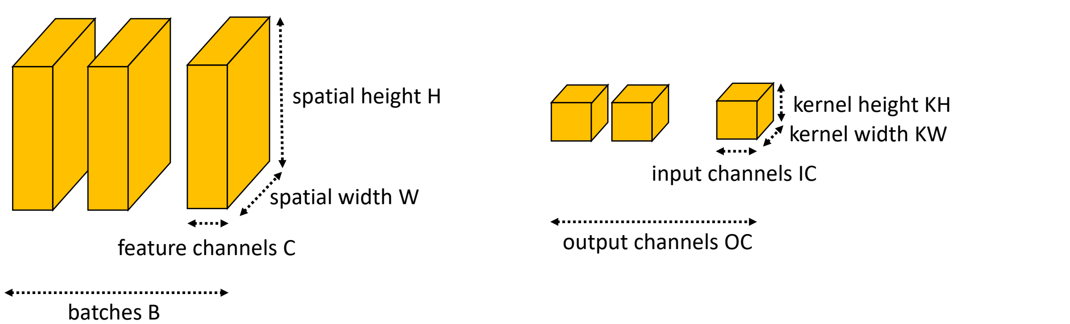
```
for i = 1 .. max_iterations

    input, ground_truth = load_minibatch(data, i)

    output = network_evaluate(input, parameters)

    loss = compute_loss(output, ground_truth)

    # gradients of loss with respect to parameters
    gradients = network_backpropagate(loss, parameters)

    parameters = optimizer_step(parameters, gradients)
```

Tensors

- Frameworks typically represent data as tensors
- Examples:

4D input data: B x C x H x W

4D convolution kernel: OC x IC x KH x KW



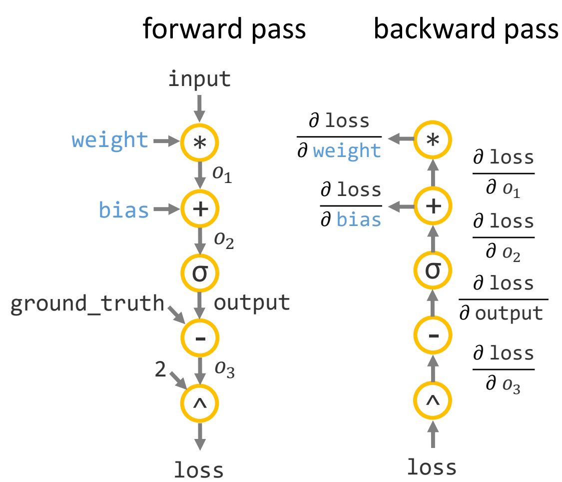
What Does a Deep Learning Framework Do?

- Tensor math
- Common network operations/layers
- Gradients of common operations
- Backpropagation
- Optimizers
- GPU implementations of the above
- usually: data loading, network parameter saving/loading
- sometimes: distributed computing

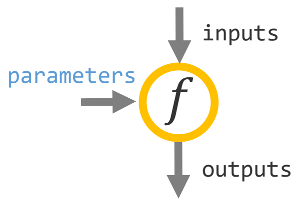
Automatic Differentiation & the Computation Graph

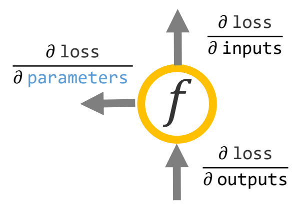
```
parameters = (weight, bias)

output = σ(weight * input + bias)

loss = (output - ground_truth)^2

# gradients of loss with respect to parameters
gradients = backpropagate(loss, parameters)
```

Since loss is a scalar, the gradients are the same size as the parameters



Automatic Differentiation & the Computation Graph





`outputs = forward(inputs, parameters)`

$\frac{\partial \text{loss}}{\partial \text{inputs}}, \frac{\partial \text{loss}}{\partial \text{parameters}} = \text{backward}\left(\frac{\partial \text{loss}}{\partial \text{outputs}}\right)$

Static vs Dynamic Computation Graphs

- Static analysis allows optimizations and distributing workload
- Dynamic graphs make data-driven control flow easier
- In static graphs, the graph is usually defined in a separate 'language'
- Static graphs have less support for debugging

define once,
evaluate during training

Static

```
x = Variable()
loss = if_node(x < parameter[0],
    x + parameter[0],
    x - parameter[1])

for i = 1 .. max_iterations
    x = data()
    run(loss)
    backpropagate(loss, parameters)
```

define implicitly by running operations,
a new graph is created in each evaluation

Dynamic

```
for i = 1 .. max_iterations
    x = data()
    if x < parameter[0]
        loss = x + parameter[0]
    else
        loss = x - parameter[1]
    backpropagate(loss, parameters)
```

Tensorflow

TensorFlow™

- Currently the largest community
- Static graphs (dynamic graphs are in development: Eager Execution)
- Good support for deployment
- Good support for distributed computing
- Typically slower than the other three main frameworks on a single GPU

SIGGRAPH ASIA 2018 TOKYO

PyTorch

PYTORCH

- Fast growing community
- Dynamic graphs
- Distributed computing is in development (some support is already available)
- Intuitive code, easy to debug and good for experimenting with less traditional architectures due to dynamic graphs
- Very Fast

SIGGRAPH ASIA 2018 TOKYO

Keras

Keras

- A high-level interface for various backends (Tensorflow, CNTK, Theano)
- Intuitive high-level code
- Focus on optimizing time from idea to code
- Static graphs

SIGGRAPH ASIA 2018 TOKYO

Caffe

Caffe

- Created earlier than Tensorflow, PyTorch or Keras
- Less flexible and less general than the other three frameworks
- Static graphs
- Legacy - to be replaced by Caffe2: focus is on performance and deployment
  - Facebook's platform for Detectron (Mask-RCNN, DensePose, …)

SIGGRAPH ASIA 2018 TOKYO

Converting Between Frameworks

- Example: develop in one framework, deploy in another
- Currently: a large range of converters, but no clear standard
- Standardized model formats are in development

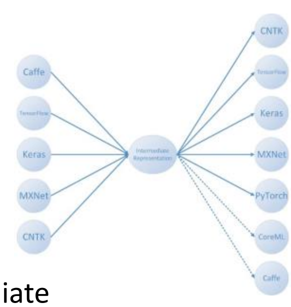
from https://github.com/ysh329/deep-learning-model-convertor

convertor	tensorflow	pytorch	keras	caffe	caffe2	CNTK	chainer	mxnet
tensorflow	-	pytorch-tf/ MMdnn	model-converters/ nn_toolsconvert-to-tensorflow/MMdnn	MMdnn/ nn_tools	None	crosstalk/MMdnn	None	MMdnn
pytorch	pytorch2keras (over Keras)	-	Pytorch2keras/ nn-transfer	Pytorch2caffe/pytorch-caffe-darknet-convert	onnx-caffe2	ONNX	None	None
keras	nn_tools /convert-to-tensorflow/keras_to_tensorflow/keras_to_tensorflow/MMdnn	MMdnn/ nn-transfer	-	MMdnnnn_tools	None	MMdnn	None	MMdnn
caffe	MMdnn/nn_tools/caffe-tensorflow	MMdnn/ pytorch-caffe-darknet-convert/ pytorch-resnet	caffe_weight_converter/ caffe2keras/nn_tools/ kerascaffe2keras/ Deep_Learning_Model_Converter/MMdnn	-	CaffeToCaffe2	crosstalkcaffe/CaffeConverterMMdnn	None	mxnet/tools/caffe_converter/ResNet_caffe2mxnet/ MMdnn
caffe2	None	ONNX	None	None	-	ONNX	None	None
CNTK	MMdnn	ONNX MMdnn	MMdnn	MMdnn	ONNX	-	None	MMdnn
chainer	None	chainer2pytorch	None	None	None	None	-	None
mxnet	MMdnn	MMdnn	MMdnn	MMdnn/MXNet2Caffe/ Mxnet2Caffe	None	MMdnn	None	-

ONNX

- Standard format for models
- Native support in development for Pytorch, Caffe2, Chainer, CNTK, and MxNet
- Converter in development for Tensorflow

MMdnn



- Converters available for several frameworks
- Common intermediate representation, but no clear standard

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

Alternatives to Direct Supervision

Niloy Mitra	Iasonas Kokkinos	Paul Guerrero	Nils Thuerey	Tobias Ritschel
UCL	UCL/Facebook	UCL	TU Munich	UCL

UCL facebook Artificial Intelligence Research Technische Universität München

Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

Unsupervised Learning

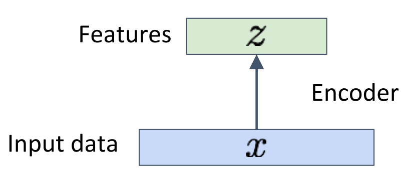
- There is no direct ground truth for the quantity of interest

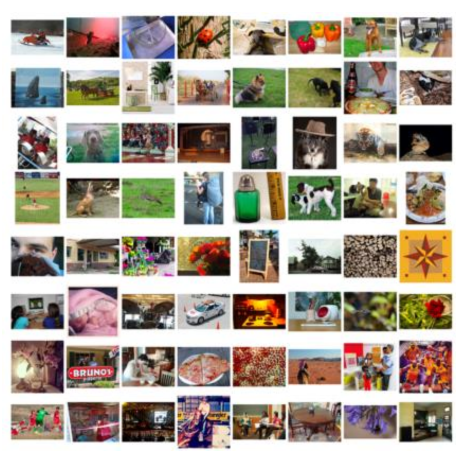
- Autoencoders
- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)

Autoencoders

Goal: Meaningful features that capture the main factors of variation in the dataset

- These are good for classification, clustering, exploration, generation, …
- We have no ground truth for them



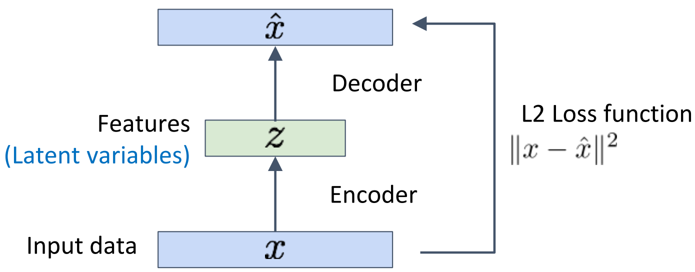


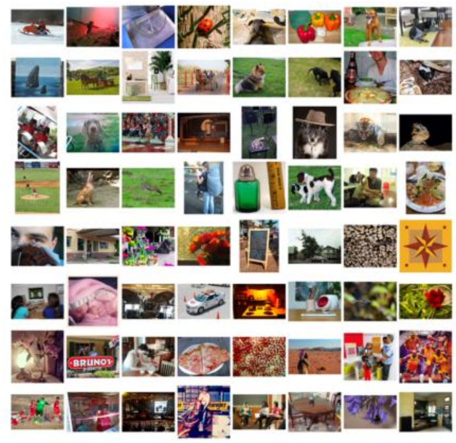
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

Goal: Meaningful features that capture the main factors of variation

Features that can be used to reconstruct the image



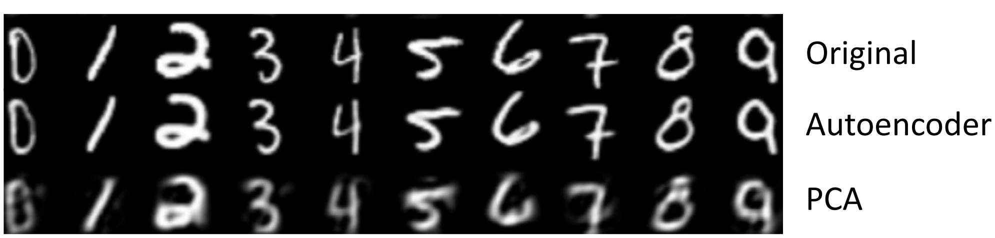


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

Linear Transformation for Encoder and Decoder give result close to PCA

Deeper networks give better reconstructions, since basis can be non-linear



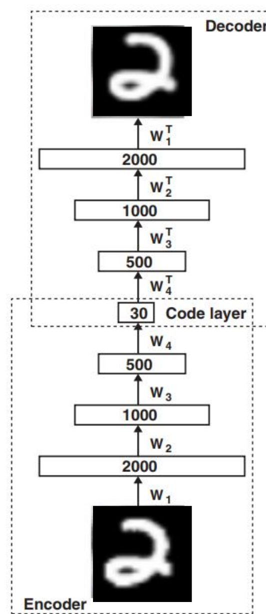
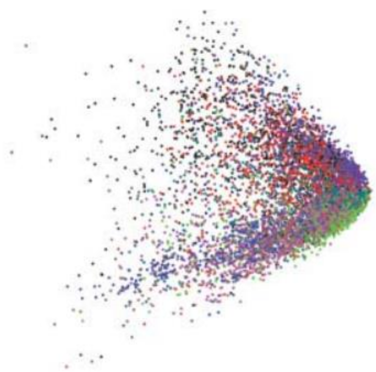


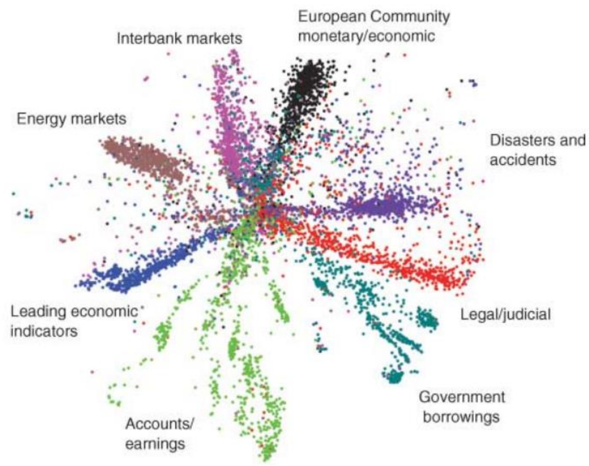
Image Credit: Reducing the Dimensionality of Data with Neural Networks, . Hinton and Salakhutdinov

Example: Document Word Prob. → 2D Code

LSA (based on PCA)



Autoencoder



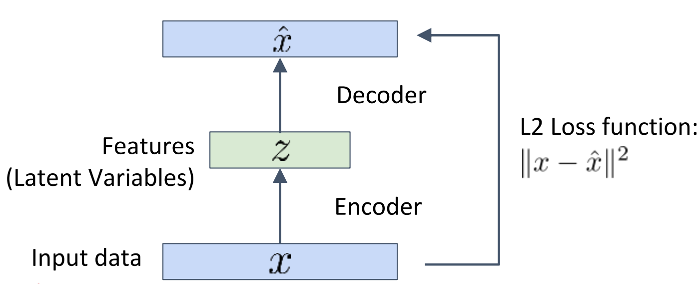
SIGGRAPH ASIA 2018 TOKYO

Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

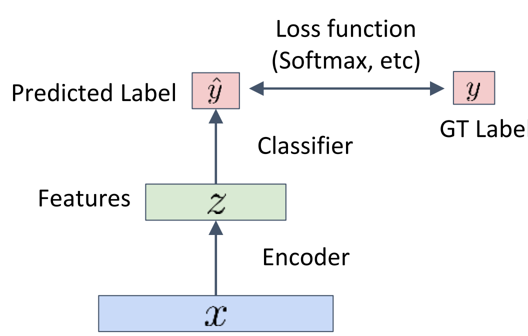
Example: Semi-Supervised Classification

- Many images, but few ground truth labels

start unsupervised
train autoencoder on many images



supervised fine-tuning
train classification network on labeled images



SIGGRAPH ASIA 2018 TOKYO

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Code example

Autoencoder
(autoencoder.ipynb)

9

Generative Models

- Assumption: the dataset are samples from an unknown distribution $p_{\text{data}}(x)$
- Goal: create a new sample from $p_{\text{data}}(x)$ that is not in the dataset

Dataset Generated

SIGGRAPH ASIA 2018 TOKYO

Image credit: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* Karras et al.

Generative Models

- Assumption: the dataset are samples from an unknown distribution $p_{\text{data}}(x)$
- Goal: create a new sample from $p_{\text{data}}(x)$ that is not in the dataset

Dataset

Generated

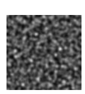
SIGGRAPH ASIA 2018 TOKYO

Image credit: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* Karras et al.

Generative Models

$p_\theta(x) \approx p_{\text{data}}(x)$

Generator with parameters θ

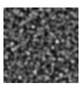


$p(z)$

known and easy to sample from

SIGGRAPH ASIA 2018 TOKYO

Generative Models



$p_\theta(x) \approx p_{\text{data}}(x)$

Generator with parameters θ

$p(z)$

known and easy to sample from

How to measure similarity of $p_\theta(x)$ and $p_{\text{data}}(x)$?

1) Likelihood of data in $p_\theta(x)$

Variational Autoencoders (VAEs)

2) Adversarial game:

Discriminator distinguishes $p_\theta(x)$ and $p_{\text{data}}(x)$ vs *Generator* makes it hard to distinguish

Generative Adversarial Networks (GANs)

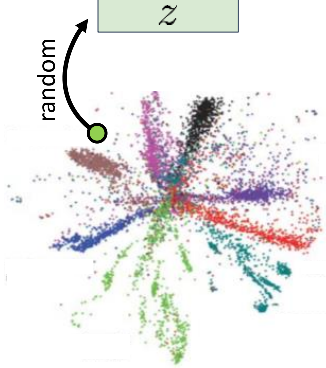
SIGGRAPH ASIA 2018 TOKYO

Autoencoders as Generative Models?

$\hat{x}$

Decoder = Generator?

z



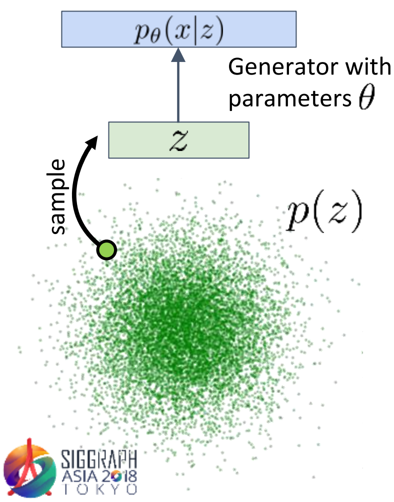
Feature space / latent space

- A trained decoder transforms some features z to approximate samples from $p_{\text{data}}(x)$
- What happens if we pick a random z?
- We do not know the distribution $p(z)$ of features that decode to likely samples

SIGGRAPH ASIA 2018 TOKYO

Image Credit: *Reducing the Dimensionality of Data with Neural Networks*, Hinton and Salakhutdinov

Variational Autoencoders (VAEs)



- Pick a parametric distribution $p(z)$ for features
- The generator maps $p(z)$ to an image distribution $p_\theta(x)$ (where θ are parameters)

$$p_\theta(x) = \int p_\theta(x|z)\ p(z)\ dz$$

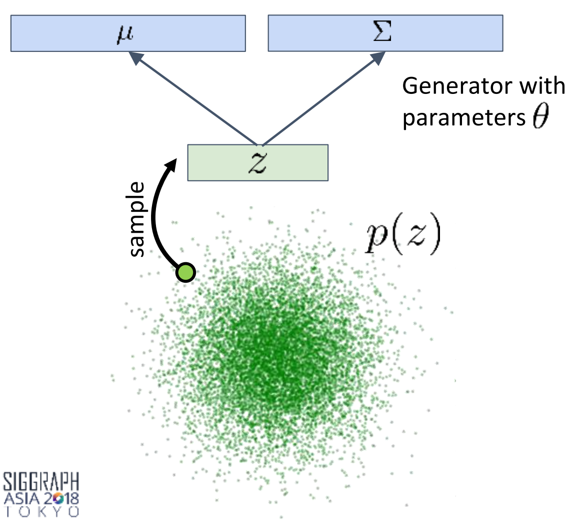
- Train the generator to maximize the likelihood of the data in $p_\theta(x)$:

$$\max_\theta \sum_{x \in \text{data}} \log p_\theta(x)$$

Outputting a Distribution

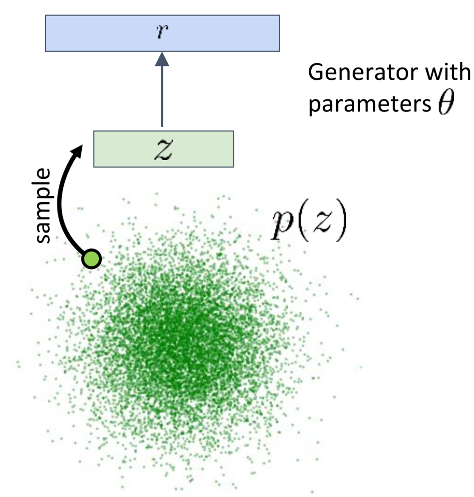
Normal distribution

$$p_\theta(x|z) = N(x; \mu(z), \Sigma(z))$$



Bernoulli distribution

$$p_\theta(x|z) = Bern(x; r(z))$$



Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)

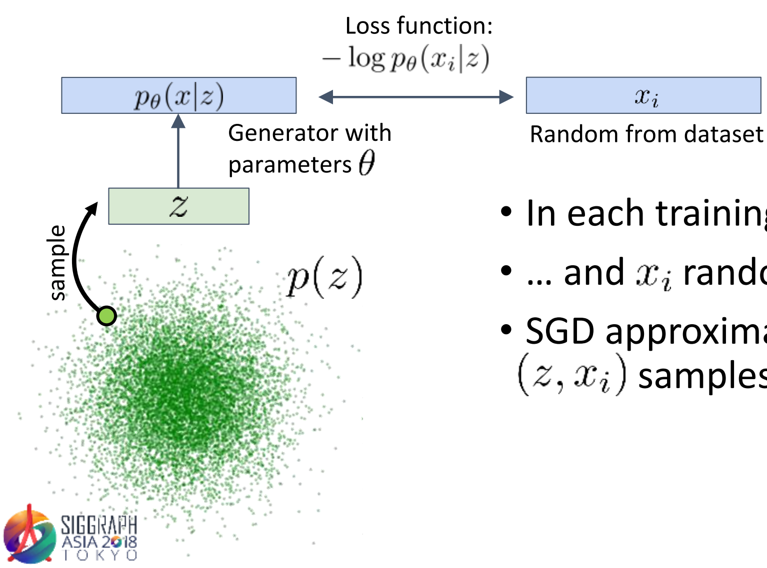
$$\theta^* = \arg\max_\theta \sum_{x \in \text{data}} \log \int p_\theta(x|z)\ p(z)\ dz$$

$$\theta^* \approx \arg\max_\theta \mathbb{E}_{x_i \sim p_{\text{data}}(x)} \mathbb{E}_{z \sim p(z)} \log p_\theta(x_i|z)$$

- SGD approximates the expected values over (z, x_i) samples
- In each training iteration, sample z from $p(z)$ …
- … and x_i randomly from the dataset, and maximize:

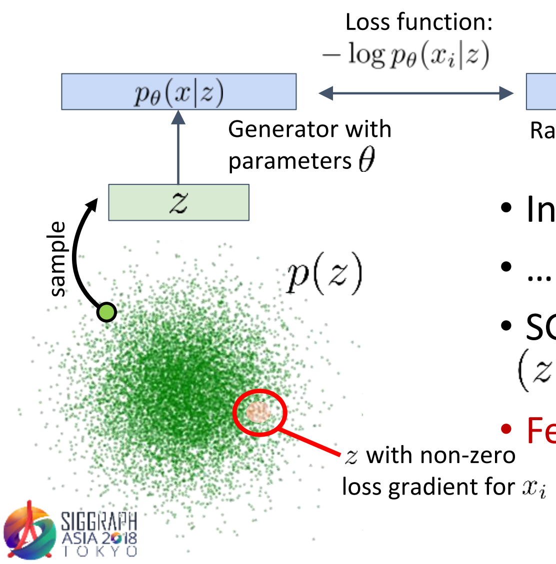
$$\max_\theta \log p_\theta(x_i|z)$$

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)



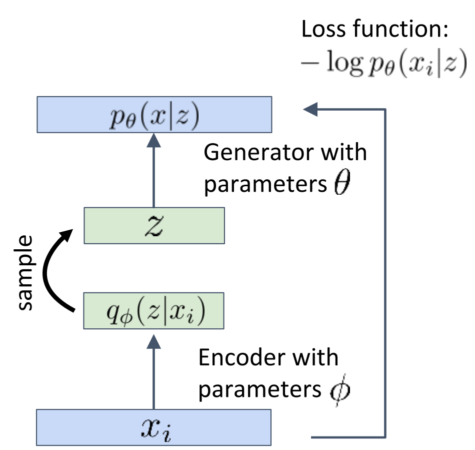
- In each training iteration, sample z from $p(z)$ …
- … and x_i randomly from the dataset
- SGD approximates the expected values over (z, x_i) samples

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)



- In each training iteration, sample z from $p(z)$ …
- … and x_i randomly from the dataset
- SGD approximates the expected values over (z, x_i) samples
- Few (z, x_i) pairs have non-zero gradients

Variational Autoencoders (VAEs): The Encoder

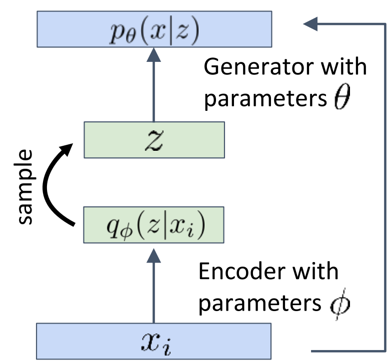


$$p_\theta(x) = \int p_\theta(x|z)\ p(z)\ dz$$

- During training, another network can guess a good z for a given x_i
- $q_\phi(z|x_i)$ should be much smaller than $p(z)$
- This also gives us the data point x_i

Variational Autoencoders (VAEs): The Encoder

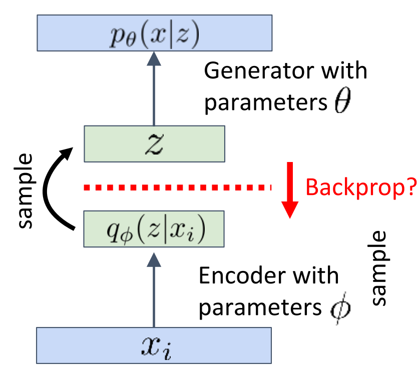
Loss function:
$-\log p_\theta(x_i|z) + KL(\ q_\phi(z|x_i) \ \|\ p(z)\ )$



- Can we still easily sample a new z?
- Need to make sure $q_\phi(z|x_i)$ approximates $p(z)$
- Regularize with KL-divergence
- Negative loss can be shown to be a lower bound for the likelihood, and equivalent if $q_\phi(z|x) = p_\theta(z|x)$

SIGGRAPH ASIA 2018 TOKYO

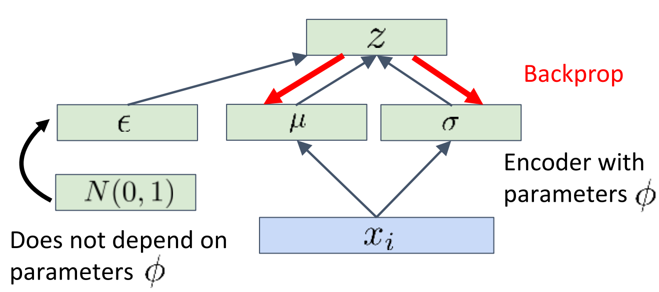
Reparameterization Trick



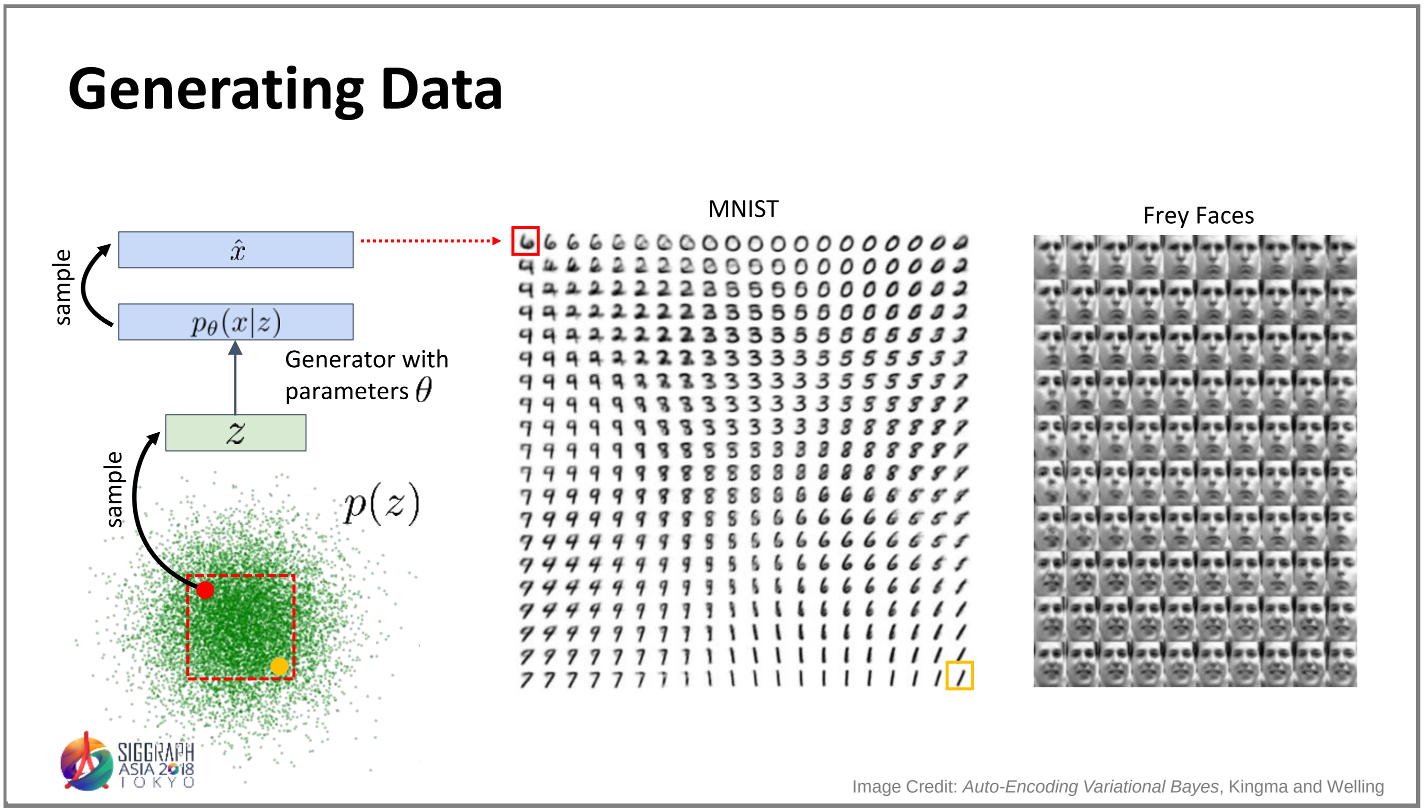
Example when $q_\phi(z|x_i) = N(z; \mu(x_i), \sigma(x_i))$:

$z = \sigma + \mu \cdot \epsilon$, where $\epsilon \sim N(0,1)$

$$\frac{\partial z}{\partial \phi} = \frac{\partial \mu}{\partial \phi} + \frac{\partial \sigma}{\partial \phi} \cdot \epsilon$$



SIGGRAPH ASIA 2018 TOKYO



Demos

VAE on MNIST
http://dpkingma.com/sgvb_mnist_demo/demo.html

VAE on Faces
http://vdumoulin.github.io/morphing_faces/online_demo.html

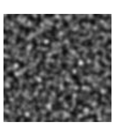
24

Code example

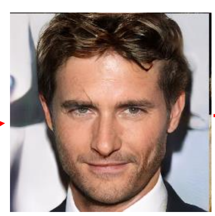
Variational Autoencoder
(variational_autoencoder.ipynb)

25

Generative Adversarial Networks



Player 1: generator
Scores if discriminator can't distinguish output from real image

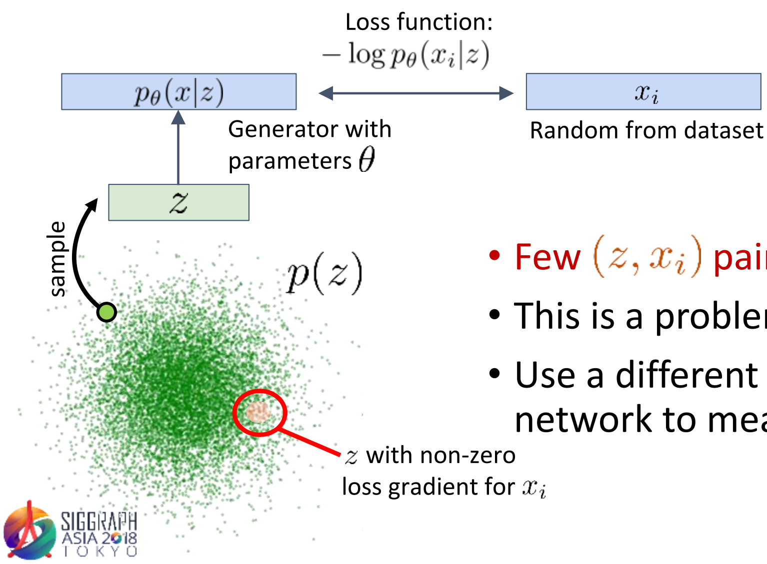


from dataset

Player 2: discriminator → real/fake
Scores if it can distinguish between real and fake

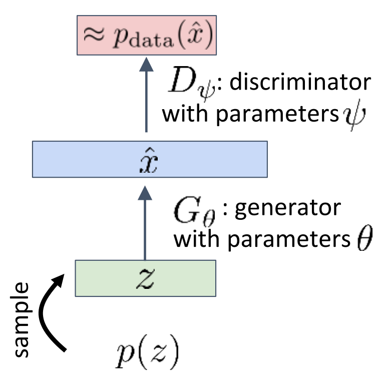
SIGGRAPH ASIA 2018 TOKYO

Naïve Sampling Revisited



- Few (z, x_i) pairs have non-zero gradients
- This is a problem of the maximum likelihood
- Use a different loss: Train a discriminator network to measure similarity $p_\theta(x) \approx p_{\text{data}}(x)$

Why Adversarial?



- If discriminator approximates $p_{\text{data}}(x)$:
- x^* at maximum of $p_{\text{data}}(x)$ has lowest loss
- Optimal $p_\theta(x)$ has single mode at x^*, small variance

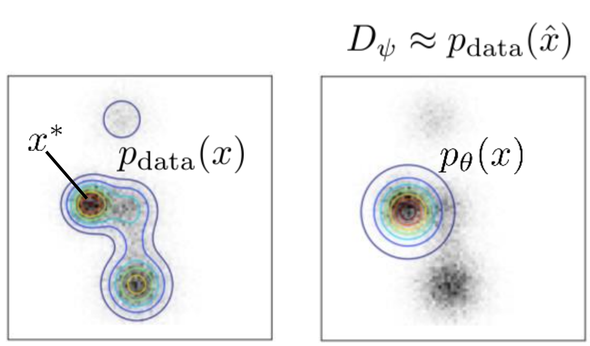
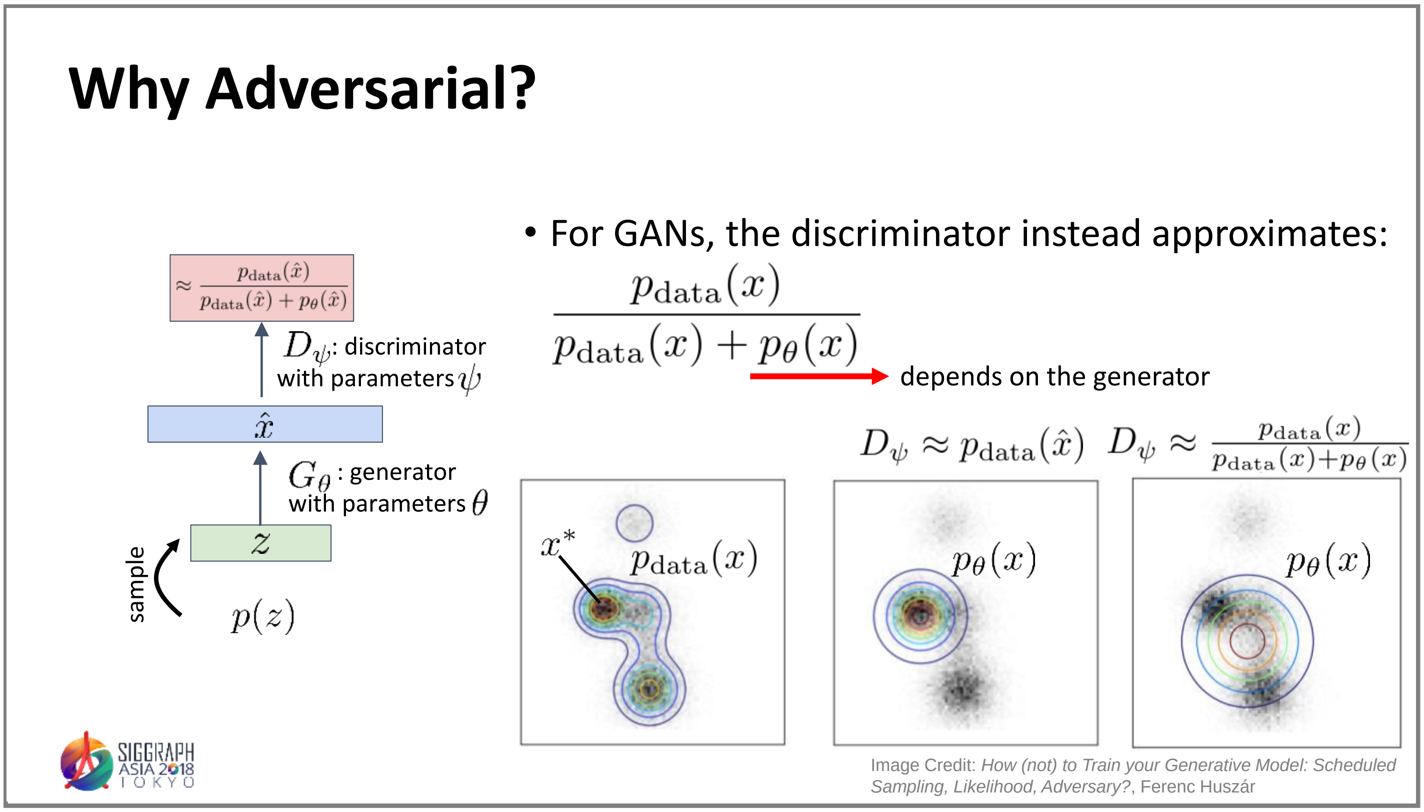
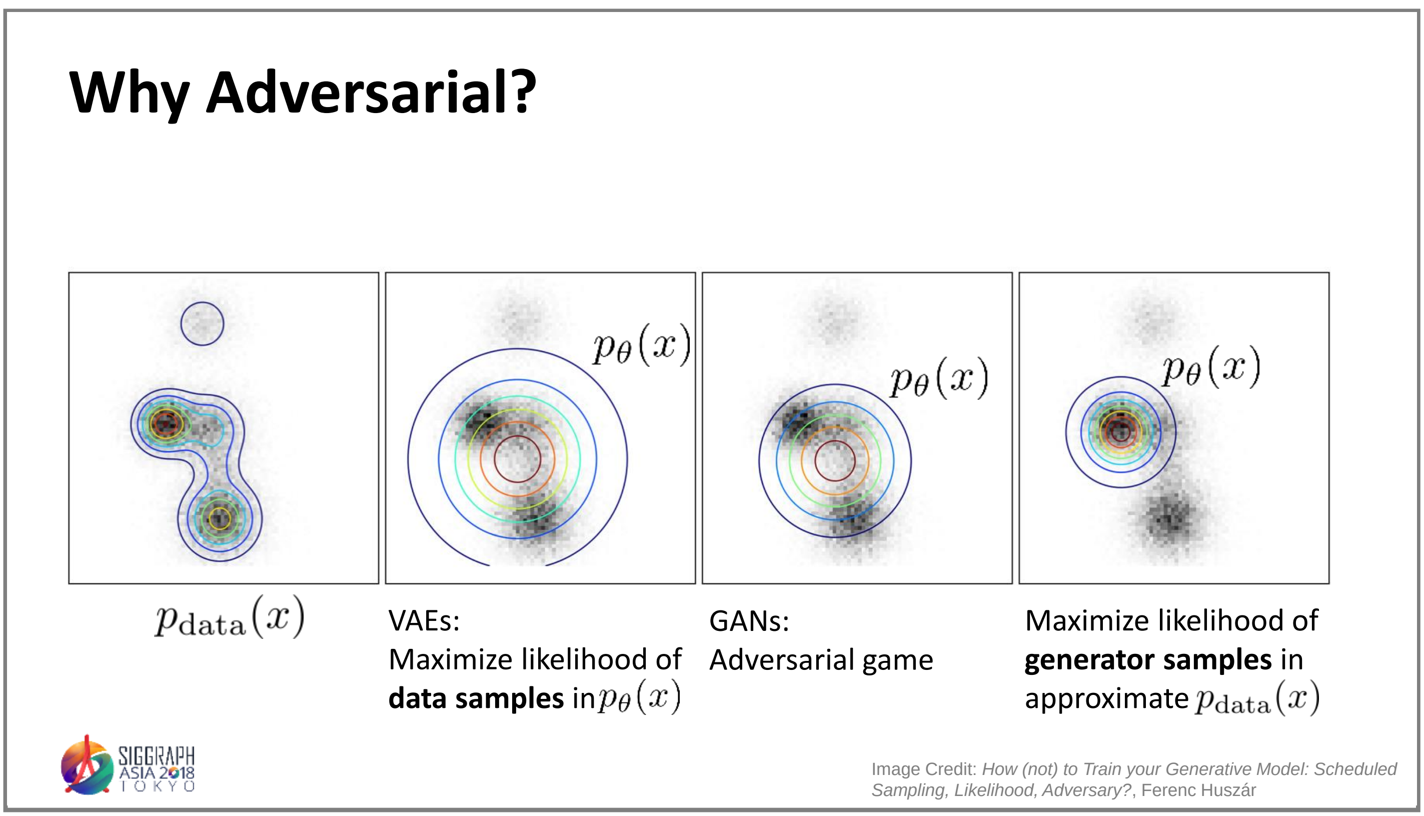
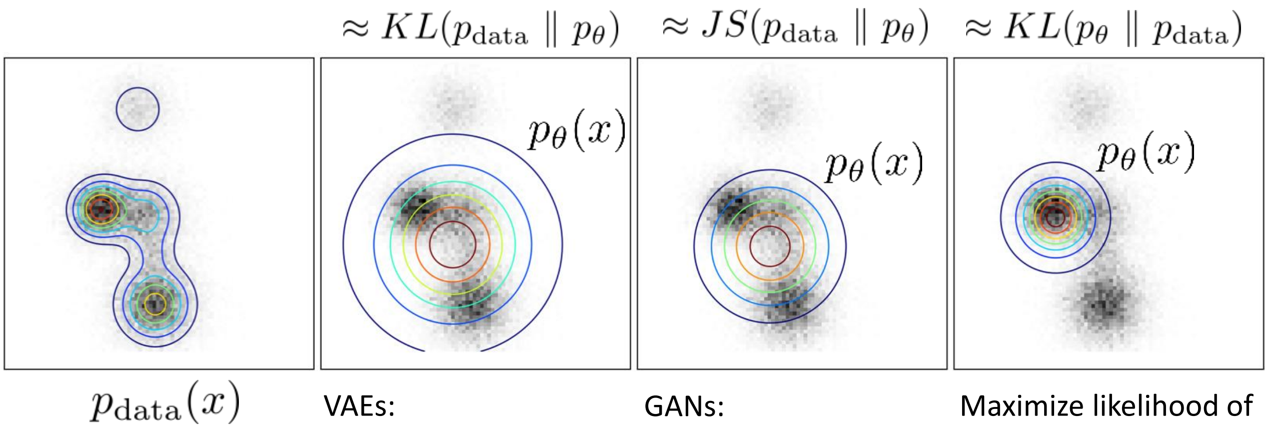


Image Credit: *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, Ferenc Huszár


Why Adversarial?
• For GANs, the discriminator instead approximates:
$\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\theta(x)}$
depends on the generator
$\approx \frac{p_{\text{data}}(\hat{x})}{p_{\text{data}}(\hat{x}) + p_\theta(\hat{x})}$
D_ψ: discriminator with parameters ψ
$\hat{x}$
G_θ : generator with parameters θ
z
sample
$p(z)$
$D_\psi \approx p_{\text{data}}(\hat{x})$
$D_\psi \approx \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_\theta(x)}$
x^*
$p_{\text{data}}(x)$
$p_\theta(x)$
$p_\theta(x)$
SIGGRAPH ASIA 2018 TOKYO
Image Credit: How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?, Ferenc Huszár


Why Adversarial?
$p_\theta(x)$
$p_\theta(x)$
$p_\theta(x)$
$p_{\text{data}}(x)$
VAEs:
Maximize likelihood of **data samples** in $p_\theta(x)$
GANs:
Adversarial game
Maximize likelihood of **generator samples** in approximate $p_{\text{data}}(x)$
SIGGRAPH ASIA 2018 TOKYO
Image Credit: How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?, Ferenc Huszár

Why Adversarial?



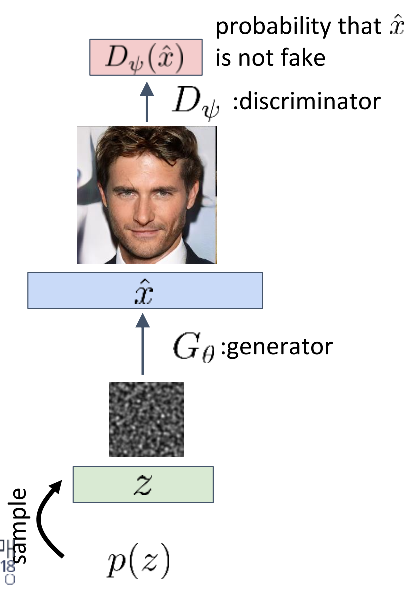
VAEs:
Maximize likelihood of **data samples** in $p_\theta(x)$

GANs:
Adversarial game

Maximize likelihood of **generator samples** in approximate $p_{\text{data}}(x)$

Image Credit: *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, Ferenc Huszár

GAN Objective



fake/real classification loss (BCE):

$$L(\theta, \psi) = -0.5\, \mathbb{E}_{x \sim p_{\text{data}}} \log D_\psi(x) - 0.5\, \mathbb{E}_{x \sim p_\theta} \log(1 - D_\psi(x))$$

Discriminator objective:

$$\min_\psi L(\theta, \psi)$$

Generator objective:

$$\max_\theta L(\theta, \psi)$$

Non-saturating Heuristic

$$L(\theta, \psi) = -0.5\ \mathbb{E}_{x\sim p_{\text{data}}}\ \log D_\psi(x) - 0.5\ \mathbb{E}_{x\sim p_\theta}\ \log(1 - D_\psi(x))$$

Generator loss is negative binary cross-entropy:

$L_G(\theta, \psi) = 0.5\ \mathbb{E}_{x\sim p_\theta}\ \log(1 - D_\psi(x))$ poor convergence

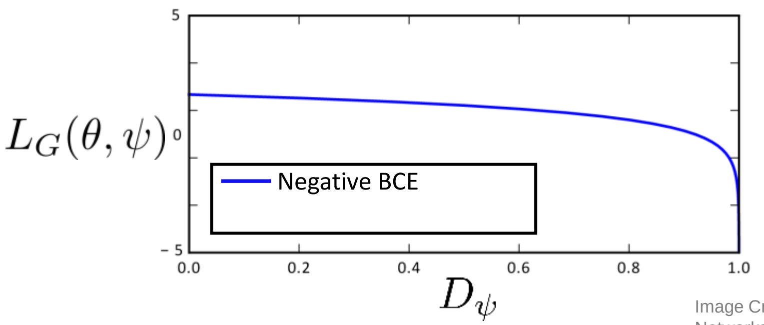


Image Credit: NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow

Non-saturating Heuristic

Generator loss is negative binary cross-entropy:

$L_G(\theta, \psi) = 0.5\ \mathbb{E}_{x\sim p_\theta}\ \log(1 - D_\psi(x))$ poor convergence

Flip target class instead of flipping the sign for generator loss:

$L_G(\theta, \psi) = -0.5\ \mathbb{E}_{x\sim p_\theta}\ \log D_\psi(x)$ good convergence – like BCE

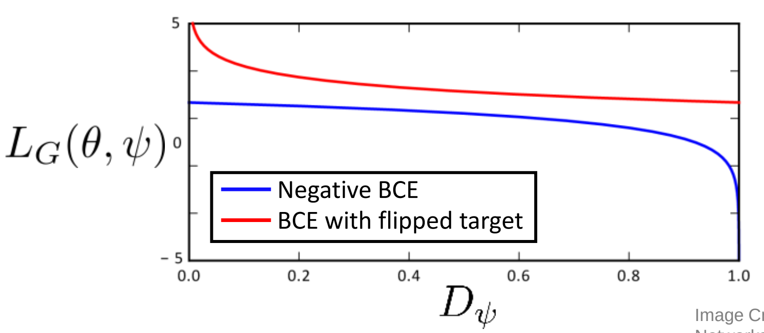
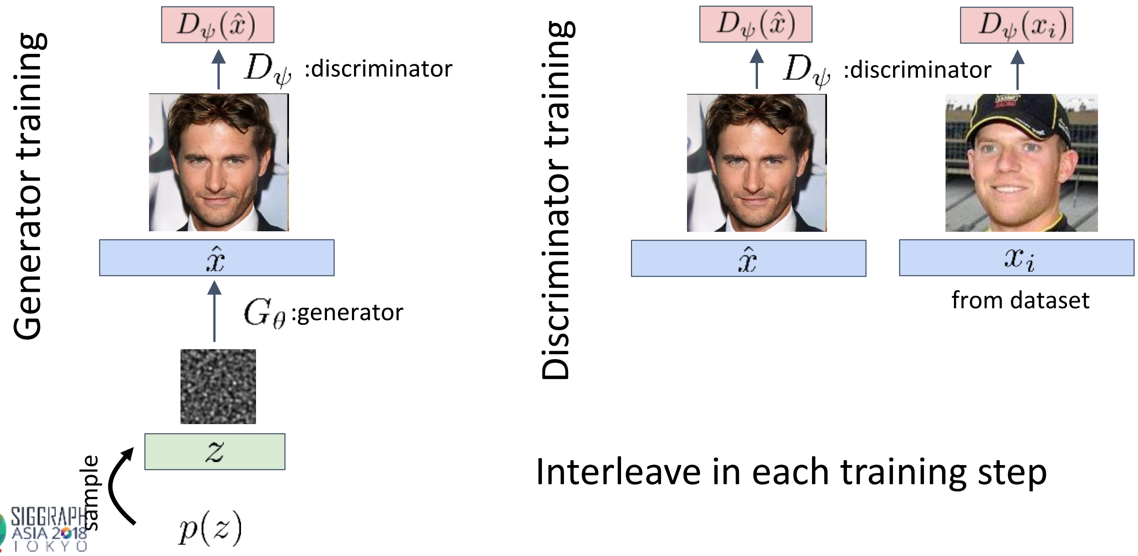


Image Credit: NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow

GAN Training

Loss:
$L_G(\theta, \psi) = -\log D_\psi(\hat{x})$

Loss:
$L_D(\theta, \psi) = -0.5 \log(1 - D_\psi(\hat{x})) - 0.5 \log D_\psi(x_i)$



Interleave in each training step

DCGAN

- First paper to successfully use CNNs with GANs
- Due to using novel components (at that time) like batch norm., ReLUs, etc.

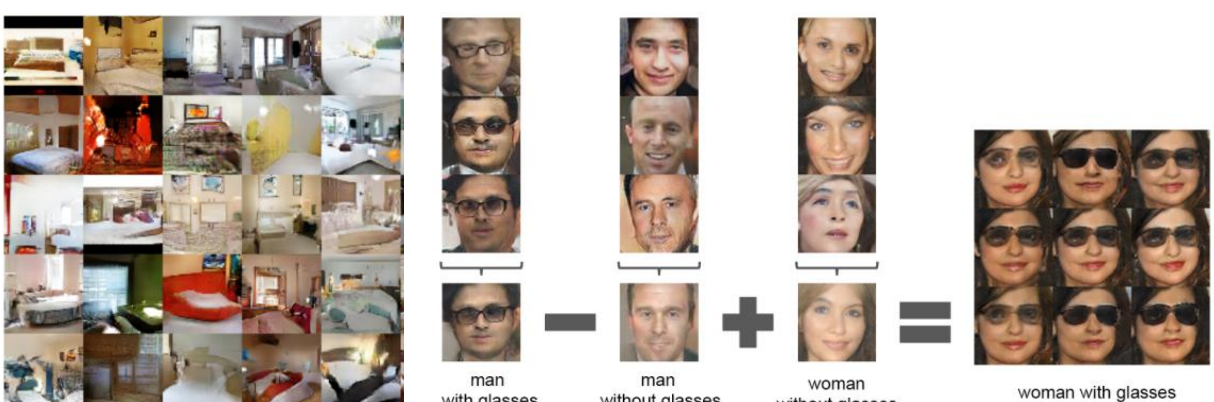
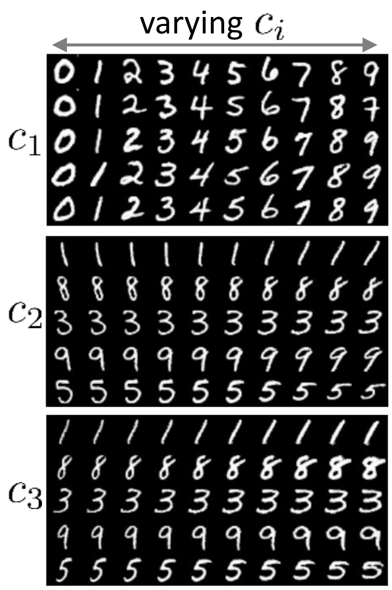


Image Credit: *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, Radford et al.

InfoGAN



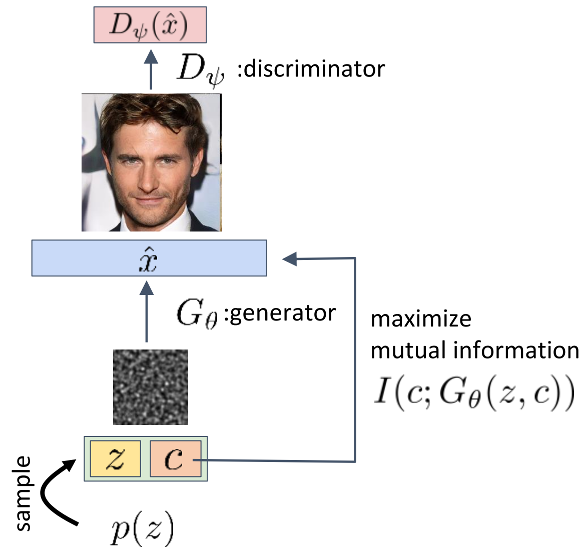


Image Credit: *InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets*, Chen et al.

Code example

Generative Adversarial Network (gan.ipynb)

38

Conditional GANs (CGANs)

- ≈ learn a mapping between images from example pairs
- Approximate sampling from a conditional distribution $p_{\text{data}}(x \mid c)$

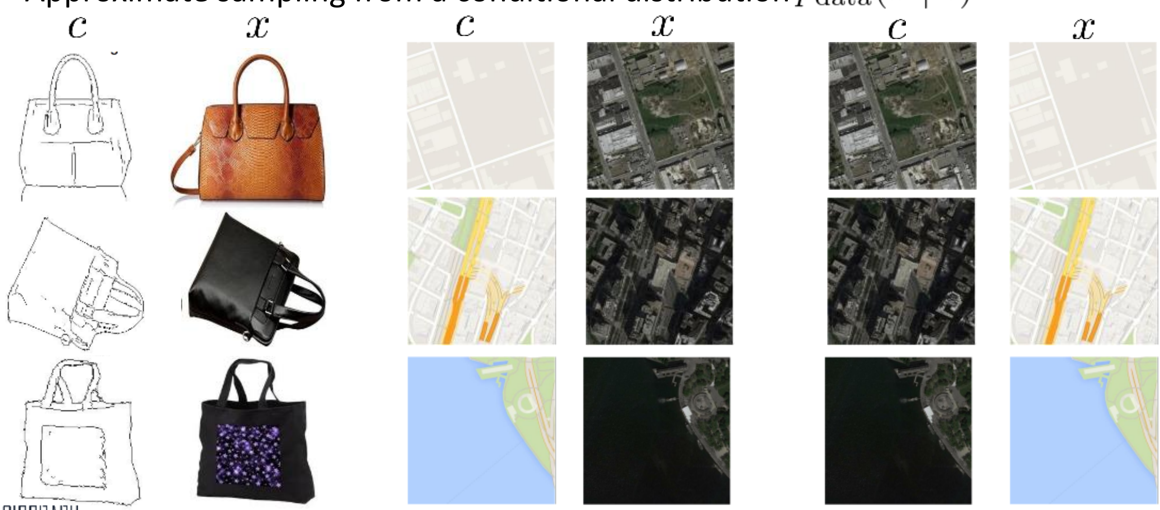


Image Credit: *Image-to-Image Translation with Conditional Adversarial Nets*, Isola et al.

Conditional GANs

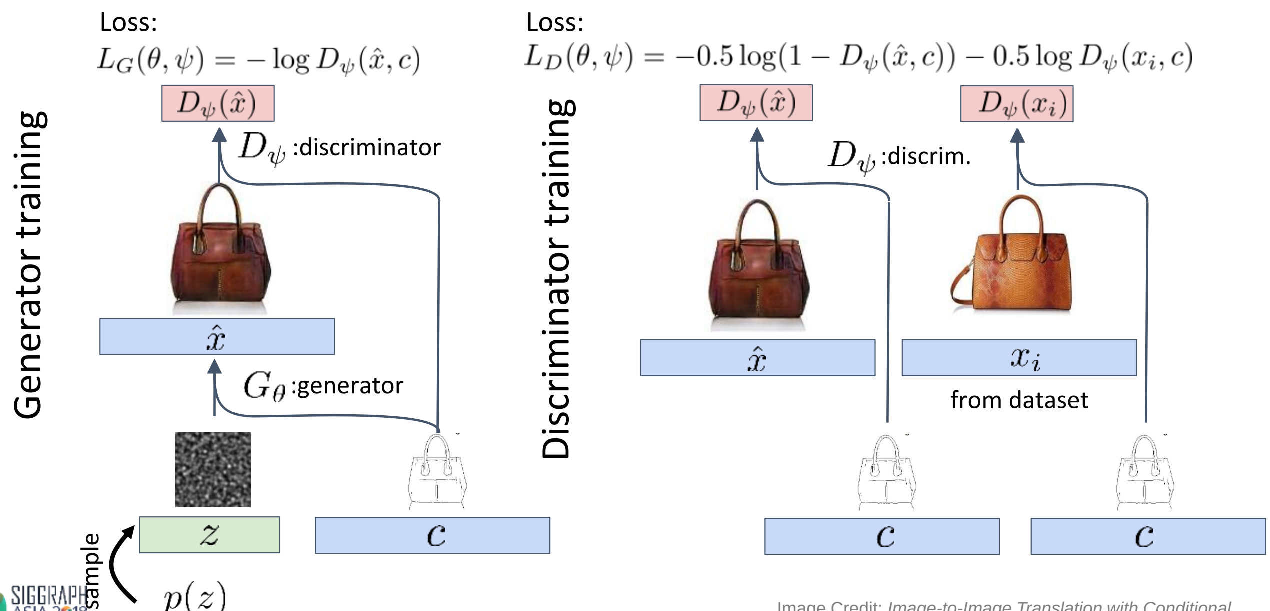
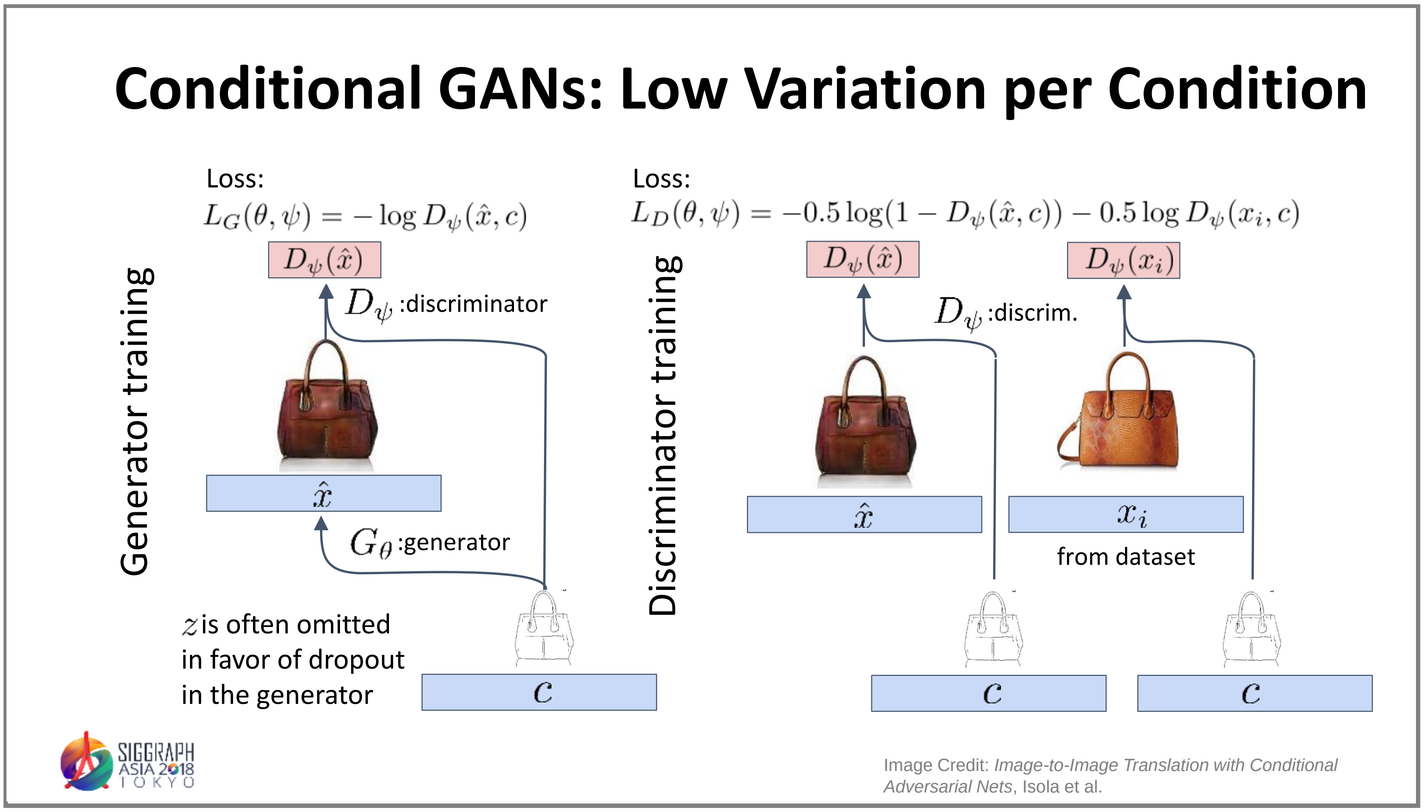


Image Credit: *Image-to-Image Translation with Conditional Adversarial Nets*, Isola et al.



Demos

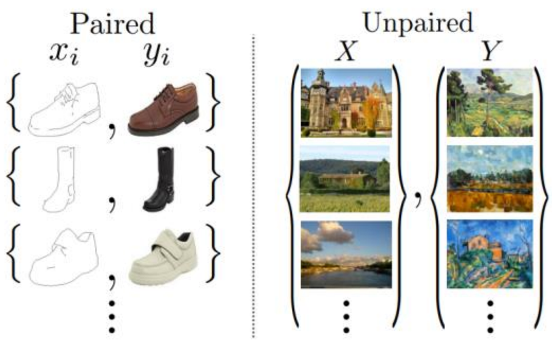
CGAN

https://affinelayer.com/pixsrv/index.html

42

CycleGANs

- Less supervision than CGANs: mapping between unpaired datasets
- Two GANs + cycle consistency



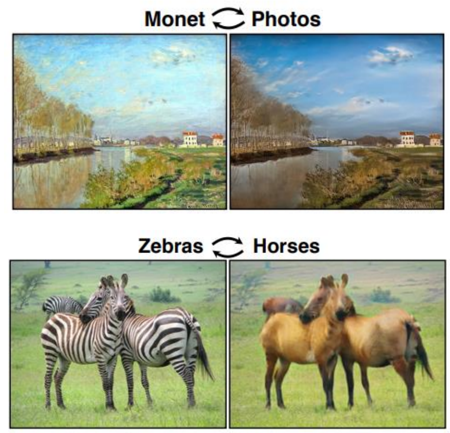
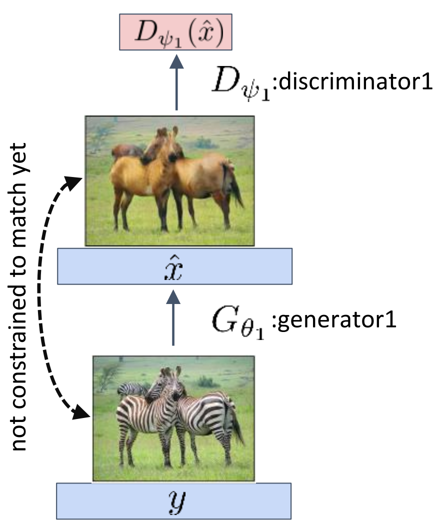


Image Credit: *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, Zhu et al.

CycleGAN: Two GANs …

- Not conditional, so this alone does not constrain generator input and output to match



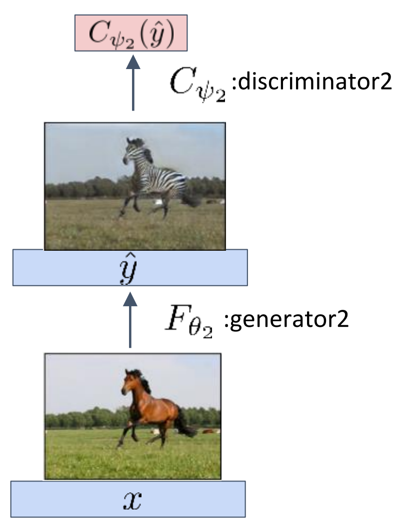


Image Credit: *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, Zhu et al.

CycleGAN: … and Cycle Consistency

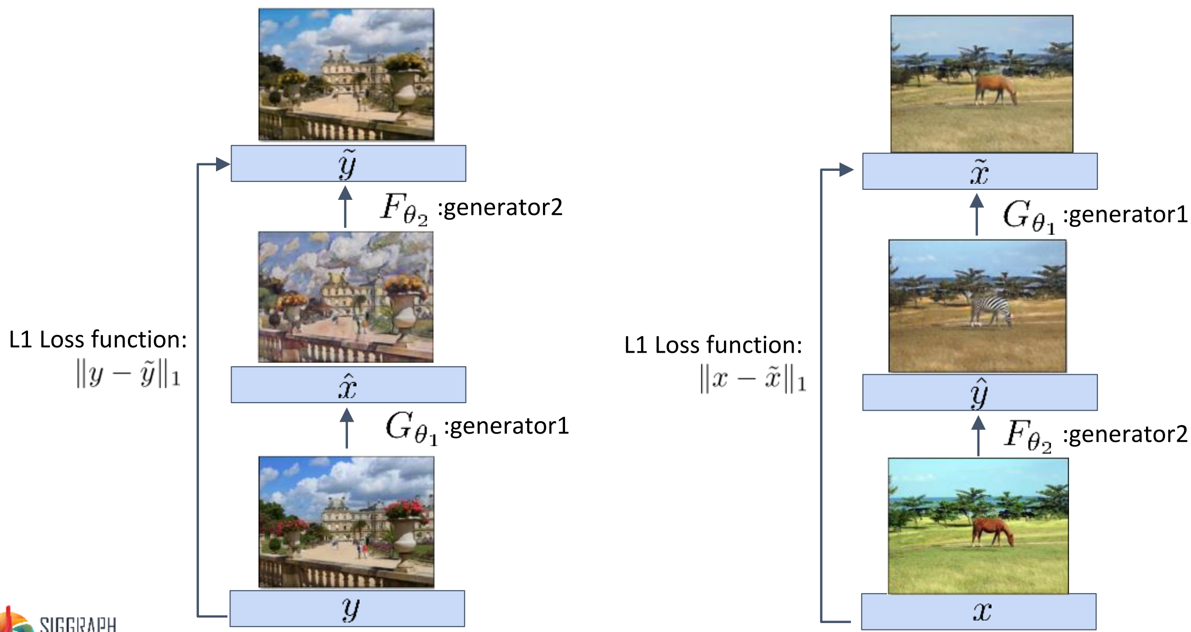


Image Credit: *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, Zhu et al.

Unstable Training

GAN training can be unstable

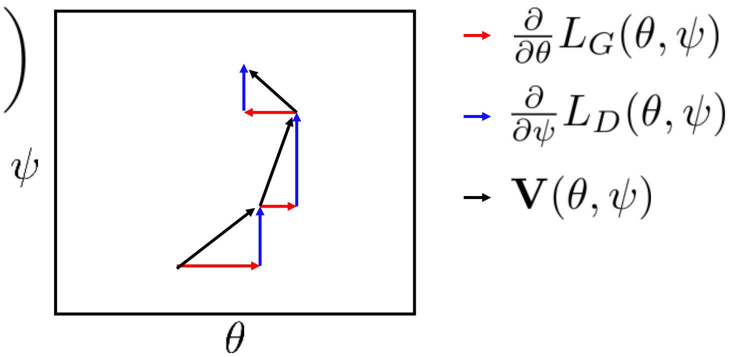
Three current research problems (may be related):

- Reaching a Nash equilibrium (the gradient for both L_G and L_D is 0)
- p_θ and p_{data} initially don't overlap
- Mode Collapse

GAN Training

- Vector-valued loss: $\mathbf{L}(\theta, \psi) = \begin{pmatrix} L_G(\theta, \psi) \\ L_D(\theta, \psi) \end{pmatrix}$
- In each iteration, gradient descent approximately follows this vector over the parameter space (θ, ψ):

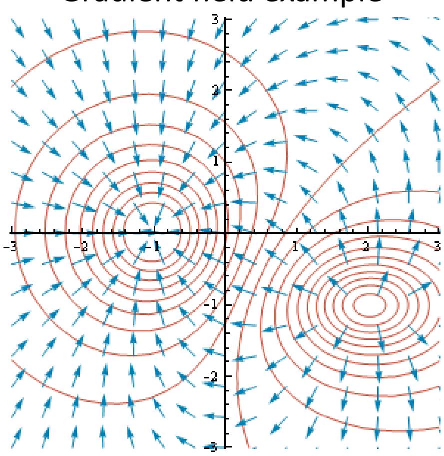
$$\mathbf{V}(\theta, \psi) = \begin{pmatrix} \frac{\partial}{\partial \theta} L_G(\theta, \psi) \\ \frac{\partial}{\partial \psi} L_D(\theta, \psi) \end{pmatrix}$$



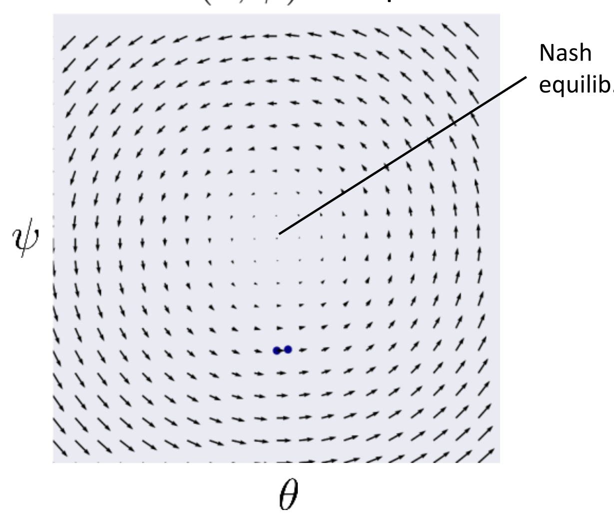
SIGGRAPH ASIA 2018 TOKYO

Reaching Nash Equilibrium

Gradient field example



$\mathbf{V}(\theta, \psi)$ Example



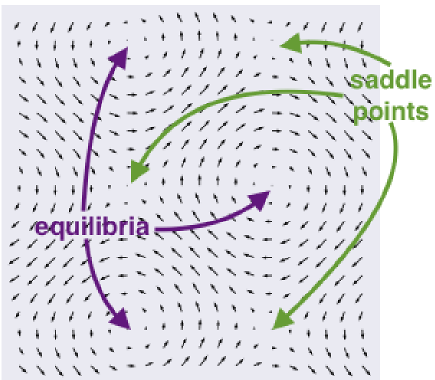
SIGGRAPH ASIA 2018 TOKYO

Image Credit: *GANs are Broken in More than One Way: The Numerics of GANs*, Ferenc Huszár

Reaching Nash Equilibrium

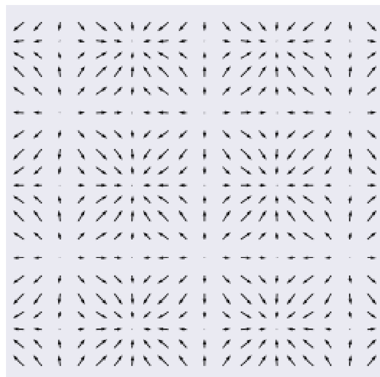
Solution attempt: relaxation with term: $-\nabla L = \frac{\partial}{\partial \theta} \|\mathbf{V}(\theta, \psi)\|_2^2$

Non-conservative field v



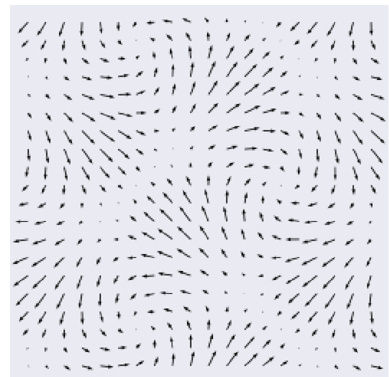
no relaxation has cycles

Conservative field $-\nabla L$



full relaxation introduces
bad Nash equilibria

Combined field $v - 0.6\nabla L$

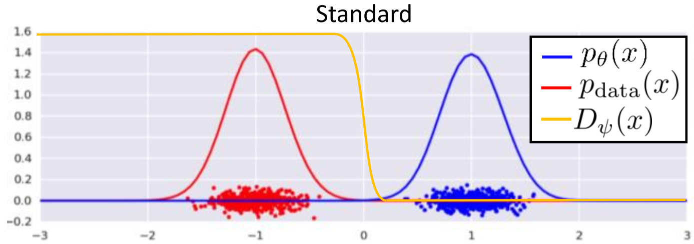


mixture works sometimes

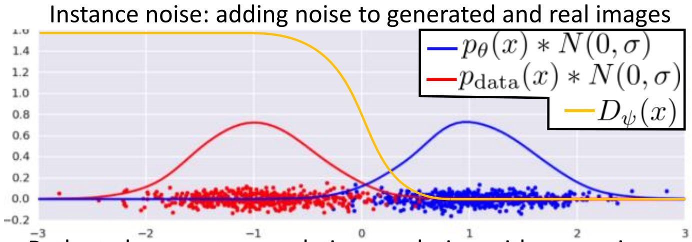
Image Credit: *GANs are Broken in More than One Way: The Numerics of GANs*, Ferenc Huszár

Generator and Data Distribution Don't Overlap

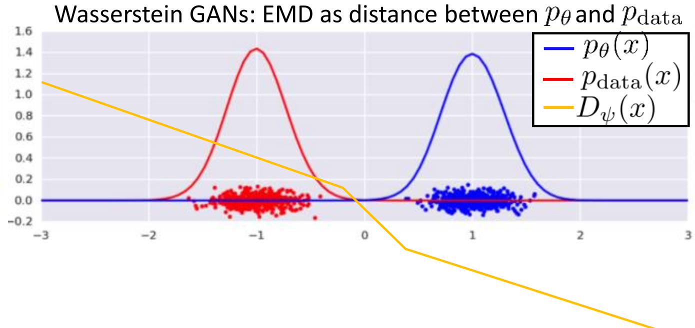
Standard



Instance noise: adding noise to generated and real images



Wasserstein GANs: EMD as distance between p_θ and p_{data}



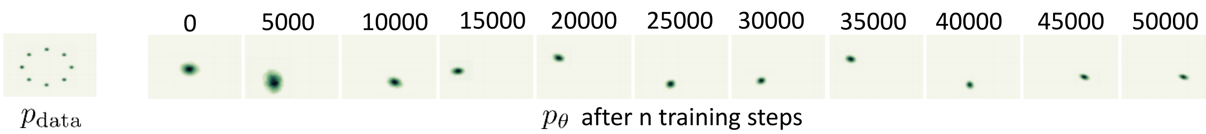
Roth et al. suggest an analytic convolution with a gaussian:
Stabilizing Training of Generative Adversarial Networks through Regularization, Roth et al. 2017

Image Credit: *Amortised MAP Inference for Image Super-resolution*, Sønderby et al.

Mode Collapse

Optimal $D_\psi(x)$: $\dfrac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\theta(x)}$

p_θ only covers one or a few modes of p_{data}



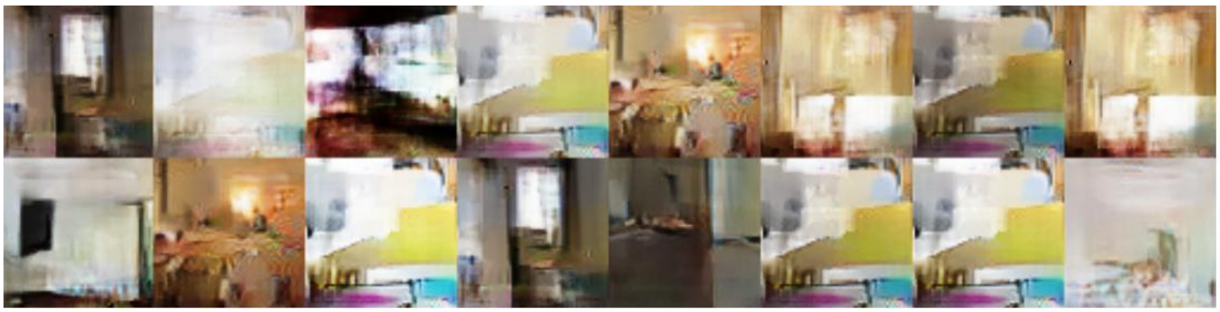


Image Credit: *Wasserstein GAN,* Arjovsky et al.
Unrolled Generative Adversarial Networks, Metz et al.

Mode Collapse

Solution attempts:

- Minibatch comparisons: Discriminator can compare instances in a minibatch (*Improved Techniques for Training GANs*, Salimans et al.)
- Unrolled GANs: Take k steps with the discriminator in each iteration, and backpropagate through all of them to update the generator

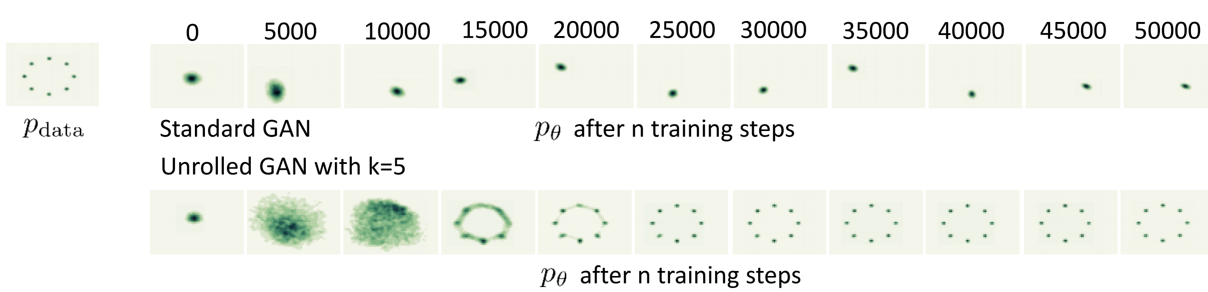


Image Credit: *Wasserstein GAN,* Arjovsky et al.
Unrolled Generative Adversarial Networks, Metz et al.

Progressive GANs

- Resolution is increased progressively during training
- Also other tricks like using minibatch statistics and normalizing feature vectors

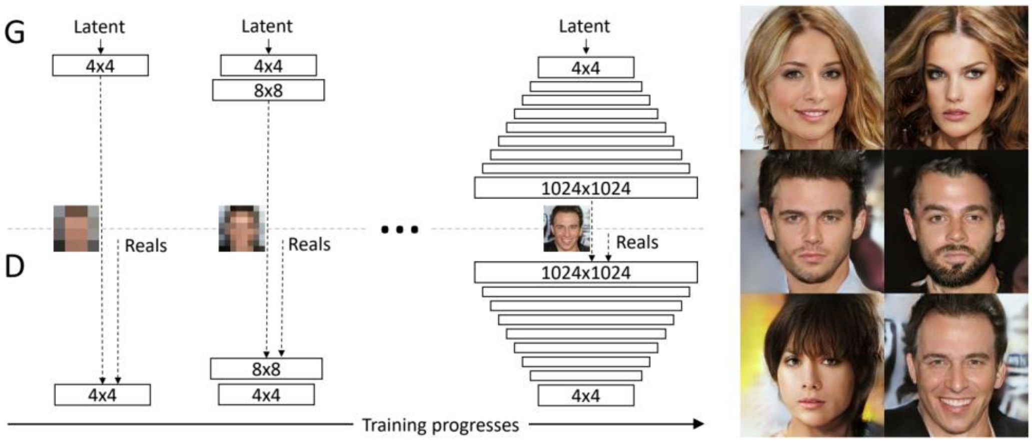


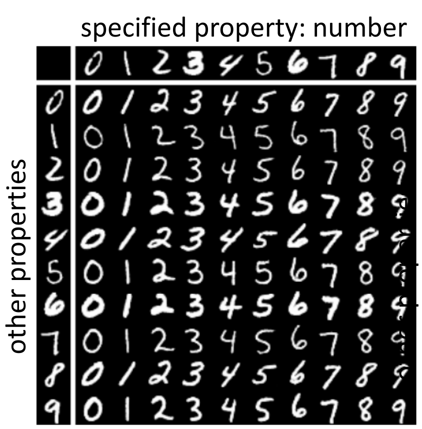
Image Credit: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* Karras et al.

53

Disentanglement

z Entangled: different properties may be mixed up over all dimensions

z_a z_b $\cdots$ Disentangled: different properties are in different dimensions



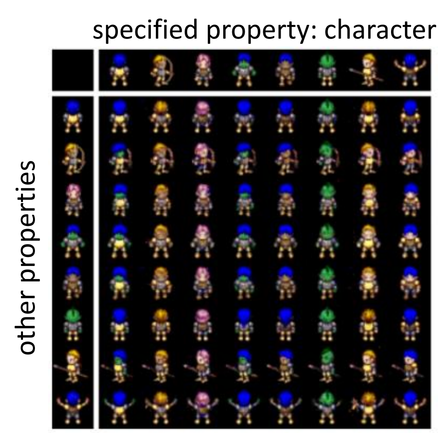


Image Credit: *Disentangling factors of variation in deep representations using adversarial training*, Mathieu et al.

Summary

- Autoencoders
  - Can infer useful latent representation for a dataset
  - Bad generators
- VAEs
  - Can infer a useful latent representation for a dataset
  - Better generators due to latent space regularization
  - Lower quality reconstructions and generated samples (usually blurry)
- GANs
  - Can not find a latent representation for a given sample (no encoder)
  - Usually better generators than VAEs
  - Currently unstable training (active research)

SIGGRAPH ASIA 2018 TOKYO

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

Feature Visualization

Niloy Mitra UCL

Iasonas Kokkinos UCL/Facebook

Paul Guerrero UCL

Nils Thuerey TU Munich

Tobias Ritschel UCL

UCL | facebook Artificial Intelligence Research | TUM Technische Universität München

Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	**Feature Visualization**					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

SIGGRAPH ASIA 2018 TOKYO

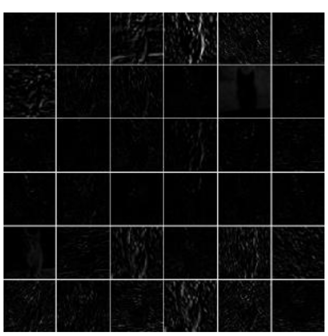
SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics** 2

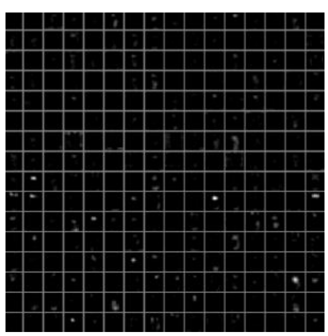
What to Visualize

- Features (activations)
- Weights (filter kernels in a CNN)
- Inputs that maximally activate some class probabilities or features
- Inputs that maximize the error (adversarial examples)

Feature Samples

- In good training, features are usually sparse
- Can find “dead” features that never activate



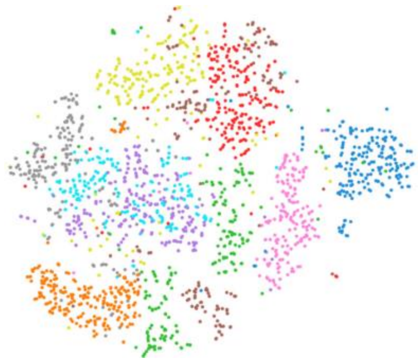


Images from: http://cs231n.github.io/understanding-cnn/

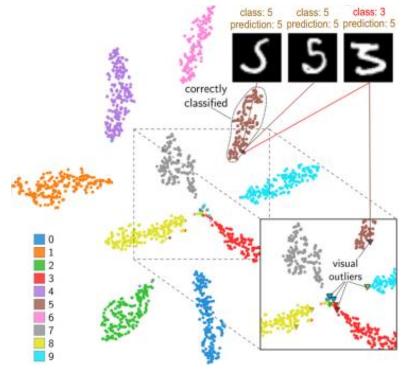
Feature Distribution using t-SNE

- Low-dimensional embedding of the features for visualization

t-SNE embedding of image features in a CNN layer



before training



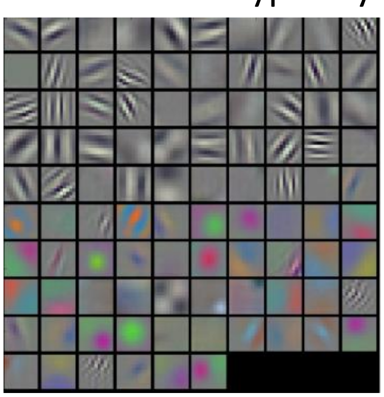
after training

t-SNE embedding of MNIST (images of digits) features in a CNN layer, colored by class

Images from: https://cs.stanford.edu/people/karpathy/cnnembed/ and
Rauber et al. *Visualizing the Hidden Activity of Artificial Neural Networks*. TVCG 2017

Weights

- Useful for CNN kernels, not useful for fully connected layers
- Kernels are typically smooth and diverse after a successful training



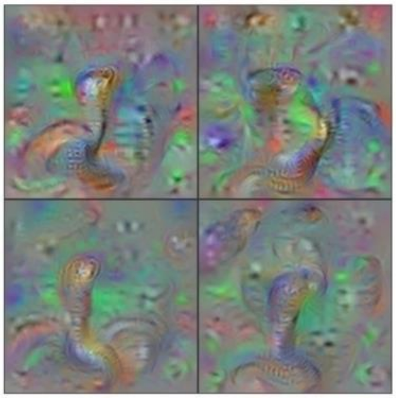
first layer filters of AlexNet

Images from: http://cs231n.github.io/understanding-cnn/

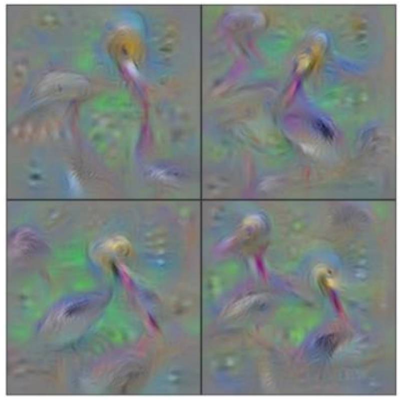
Inputs that Maximize Feature Response

Local maxima of the response for class:

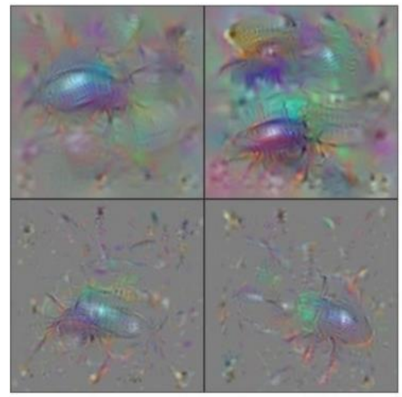
Indian Cobra



Pelican



Ground Beetle

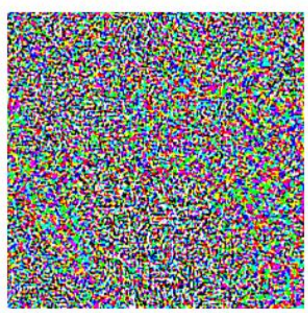


Images from: Yosinski et al. *Understanding Neural Networks Through Deep Visualization*. ICML 2015

Inputs that Maximize the Error

$$\max_{\delta \in \Delta} \mathcal{L}(x + \delta, y; \theta) \qquad \Delta = \{\delta \in \mathbb{R}^d \mid \|\delta\|_p \leq \varepsilon\}$$

$+ .007 \times$



$=$

x δ $x + \delta$

"Panda" 55.7% conf. "Gibbon" 99.3% conf.

Images from: Goodfellow et al. *Explaining and Harnessing Adversarial Examples*. ICLR 2015

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics**

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

Image Domains

Niloy Mitra UCL

Iasonas Kokkinos UCL/Facebook

Paul Guerrero UCL

Nils Thuerey TU Munich

Tobias Ritschel UCL

UCL

facebook Artificial Intelligence Research

TUM Technische Universität München

Timetable

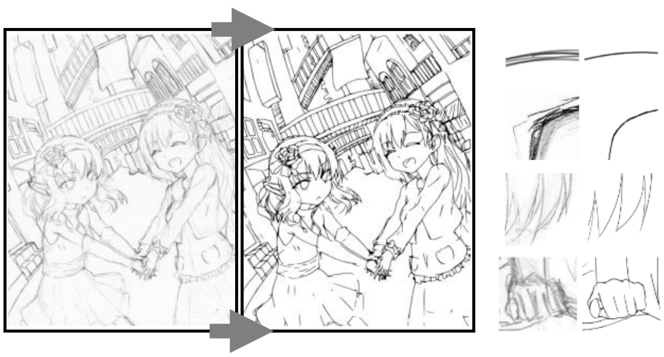
		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

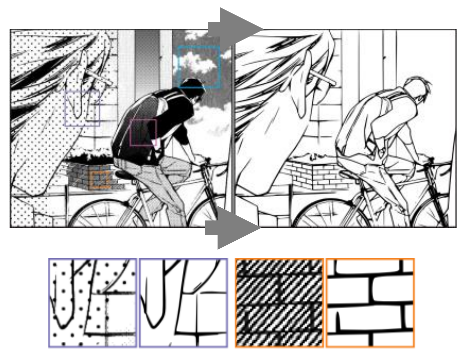
SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics** 2

Sketch Simplification

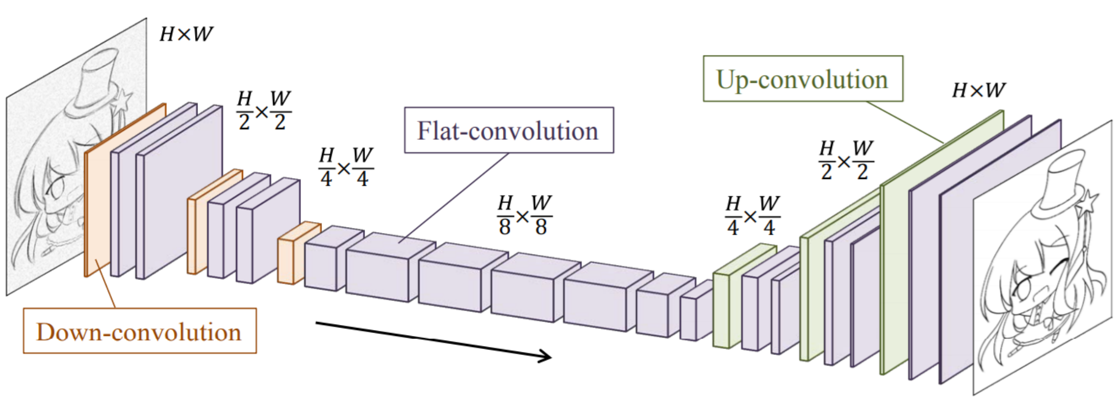
- *Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup*, Simon-Serra et al., 2016
- *Deep Extraction of Manga Structural Lines*, Li et al., 2017





3

Sketch Simplification: *Learning to Simplify*

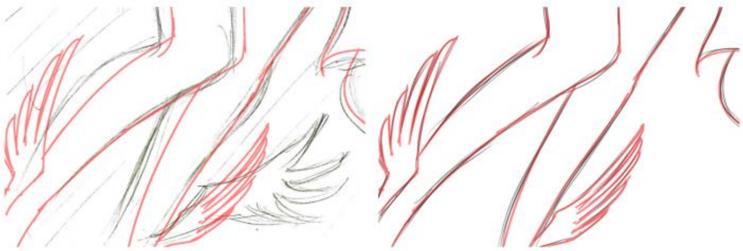


Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup, Simo-Serra et al.

4

Sketch Simplification: *Learning to Simplify*

- Loss for thin edges saturates easily
- Authors take extra steps to align input and ground truth edges



Pencil: input
Red: ground truth

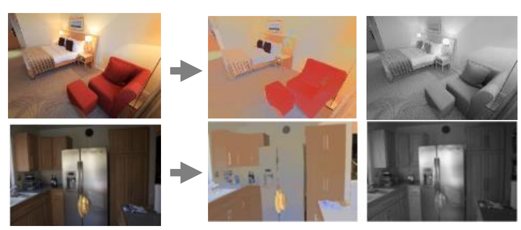
Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup, **Simo-Serra et al.**

SIGGRAPH ASIA 2018 TOKYO

5

Image Decomposition

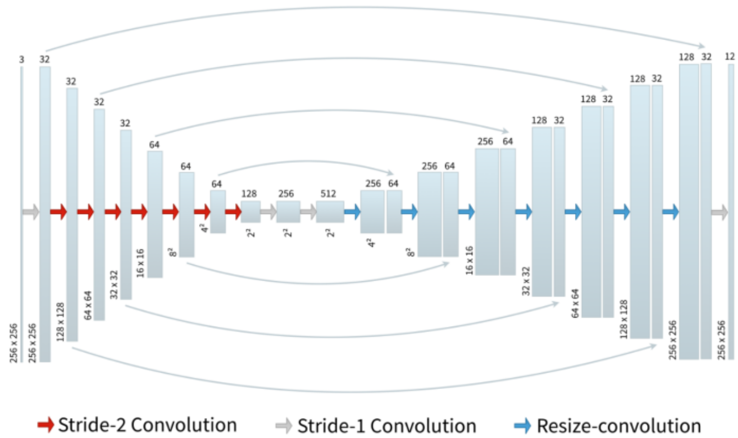
- A selection of methods:
- *Direct Instrinsics*, Narihira et al., 2015
- *Learning Data-driven Reflectance Priors for Intrinsic Image Decomposition,* Zhou et al., 2015
- *Decomposing Single Images for Layered Photo Retouching*, Innamorati et al. 2017



SIGGRAPH ASIA 2018 TOKYO

6

Image Decomposition: Decomposing *Single Images for Layered Photo Retouching*



SIGGRAPH ASIA 2018 TOKYO

7

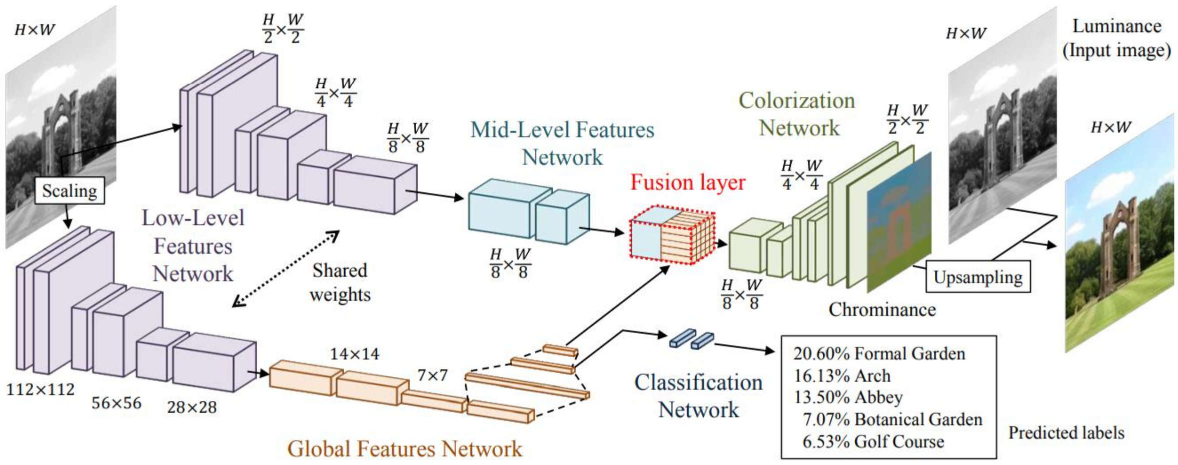
Colorization

- Concurrent methods:
  - *Let there be Color!*, Iizuka et al., 2016
  - *Colorful Image Colorization*, Zhang et al. 2016
  - *Learning Representations for Automatic Colorization,* Larsson et al., 2016
  - *Real-Time User-Guided Image Colorization with Learned Deep Priors*, Zhang et al. 2017

SIGGRAPH ASIA 2018 TOKYO

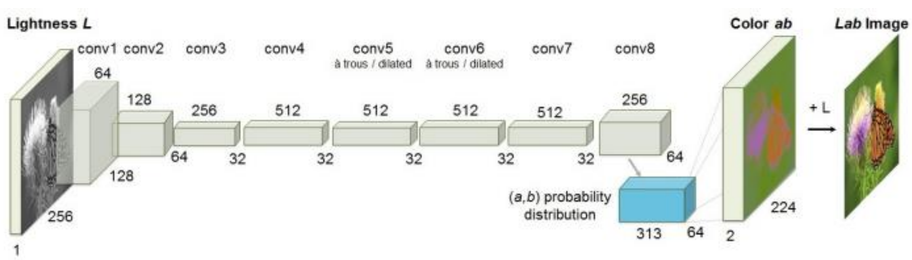
8

Colorization: *Let There Be Color!*



Let there be Color!: Iizuka et al.

Colorization: *Colorful Image Colorization*

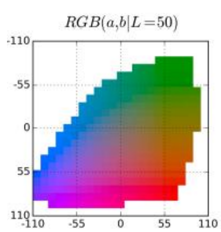


input

output

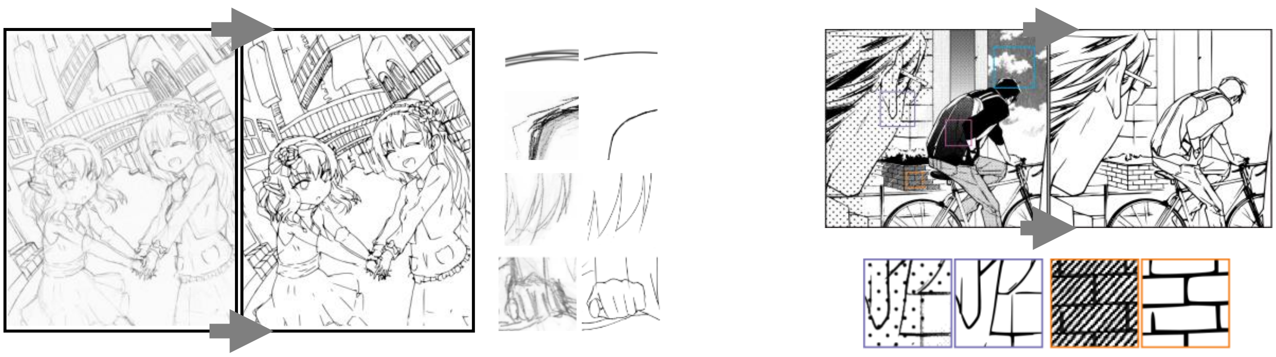
direct regression probability distr.

Image Credit: *Colorful Image Colorization*, Zhang et al.



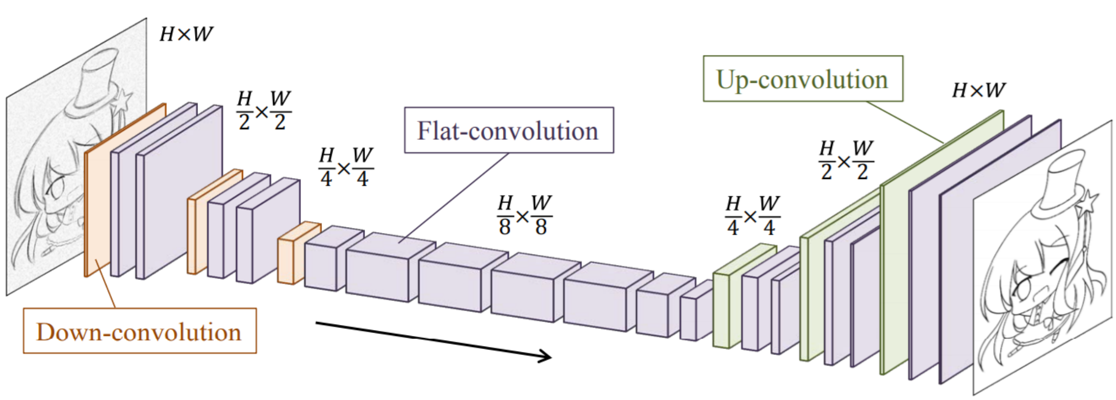
Sketch Simplification

- *Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup*, Simon-Serra et al., 2016
- *Deep Extraction of Manga Structural Lines*, Li et al., 2017



SIGGRAPH ASIA 2018 TOKYO 11

Sketch Simplification: *Learning to Simplify*

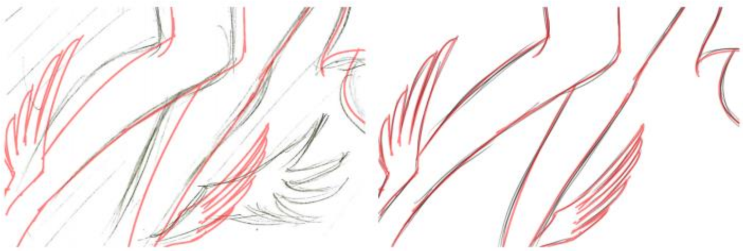


Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup, Simo-Serra et al.

SIGGRAPH ASIA 2018 TOKYO 12

Sketch Simplification: *Learning to Simplify*

- Loss for thin edges saturates easily
- Authors take extra steps to align input and ground truth edges



Pencil: input
Red: ground truth

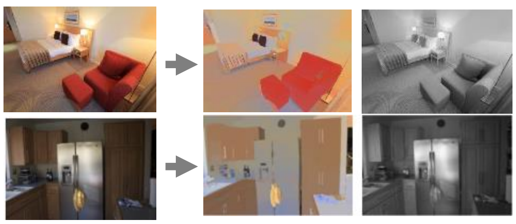
Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup, **Simo-Serra et al.**

SIGGRAPH ASIA 2018 TOKYO

13

Image Decomposition

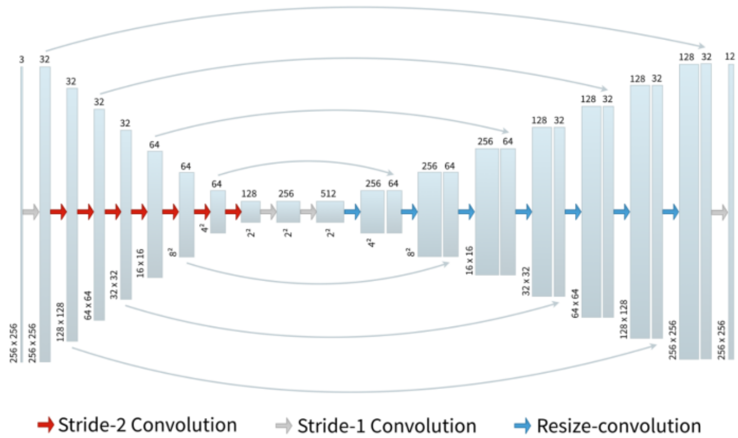
- A selection of methods:
- *Direct Instrinsics*, Narihira et al., 2015
- *Learning Data-driven Reflectance Priors for Intrinsic Image Decomposition,* Zhou et al., 2015
- *Decomposing Single Images for Layered Photo Retouching*, Innamorati et al. 2017



SIGGRAPH ASIA 2018 TOKYO

14

Image Decomposition: Decomposing *Single Images for Layered Photo Retouching*



SIGGRAPH ASIA 2018 TOKYO

15

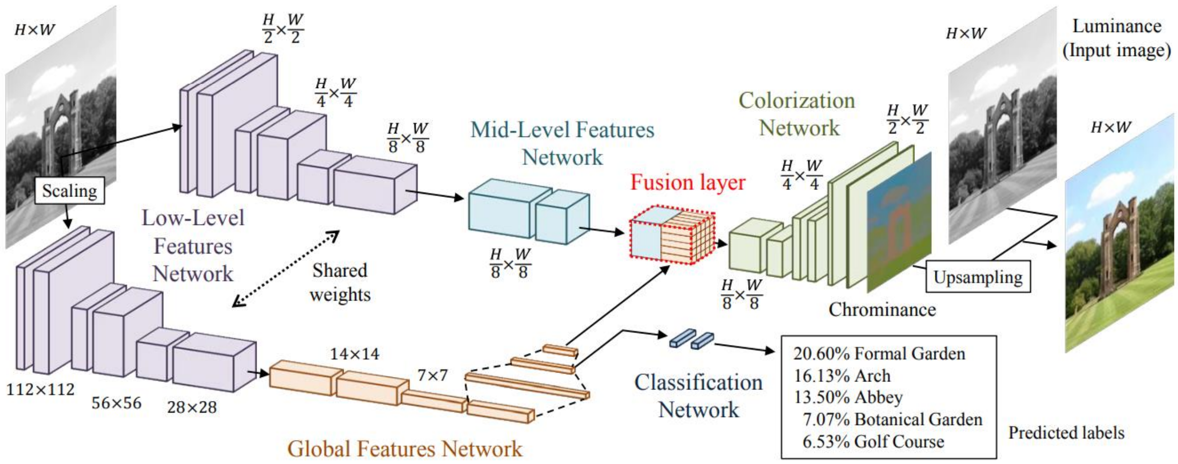
Colorization

- Concurrent methods:
  - *Let there be Color!*, Iizuka et al., 2016
  - *Colorful Image Colorization*, Zhang et al. 2016
  - *Learning Representations for Automatic Colorization,* Larsson et al., 2016
  - *Real-Time User-Guided Image Colorization with Learned Deep Priors*, Zhang et al. 2017

SIGGRAPH ASIA 2018 TOKYO

16

Colorization: *Let There Be Color!*

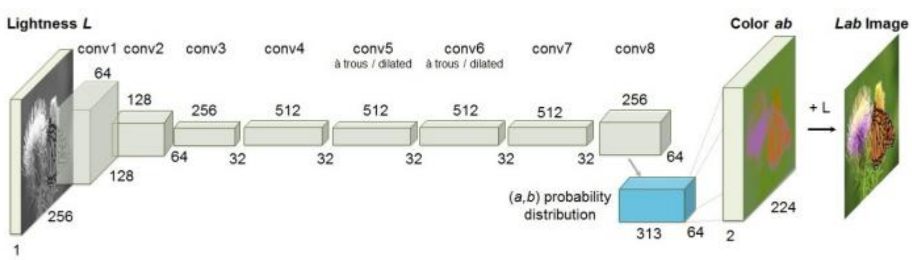


Let there be Color!: Iizuka et al.

SIGGRAPH ASIA 2018 TOKYO

17

Colorization: *Colorful Image Colorization*

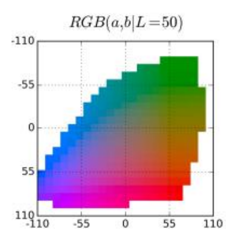


input

output

direct regression probability distr.

Image Credit: *Colorful Image Colorization*, Zhang et al.



SIGGRAPH ASIA 2018 TOKYO

18

LDR to HDR Image Reconstruction:

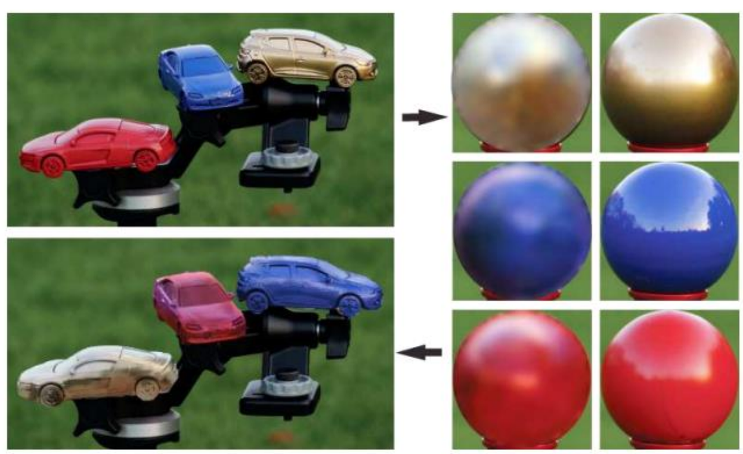
- Concurrently:
- *Deep Reverse Tone Mapping*, Endo et al. 2017
- *HDR image reconstruction from a single exposure using deep CNNs*, Eilertsen et al. 2017

SIGGRAPH ASIA 2018 TOKYO

19

Reflectance Maps

- Paint a sphere as if it is made of a material under a certain illumination of another object in a photo



Deep Reflectance Maps. Rematas et al. **CVPR 2015**

SIGGRAPH ASIA 2018 TOKYO

20

DeLight

- Factor BRDF and (HDR) Illumination

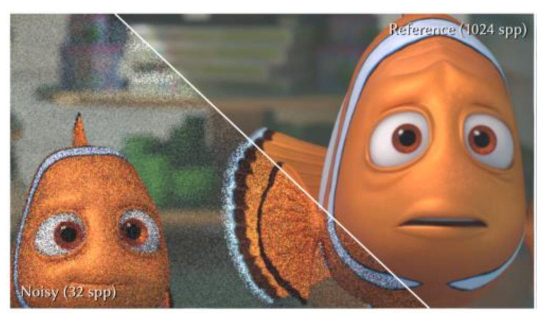
Reflectance and Natural Illumination from Single-Material Specular Objects Using Deep Learning. Georgoulis et al. **PAMI 2017**

SIGGRAPH ASIA 2018 TOKYO

21

Denoising Renderings

- Concurrent:
- *Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings*, Bako et al. 2017
- *Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder,* Chaitanya et al. 2017 (more on Autoencoders later)



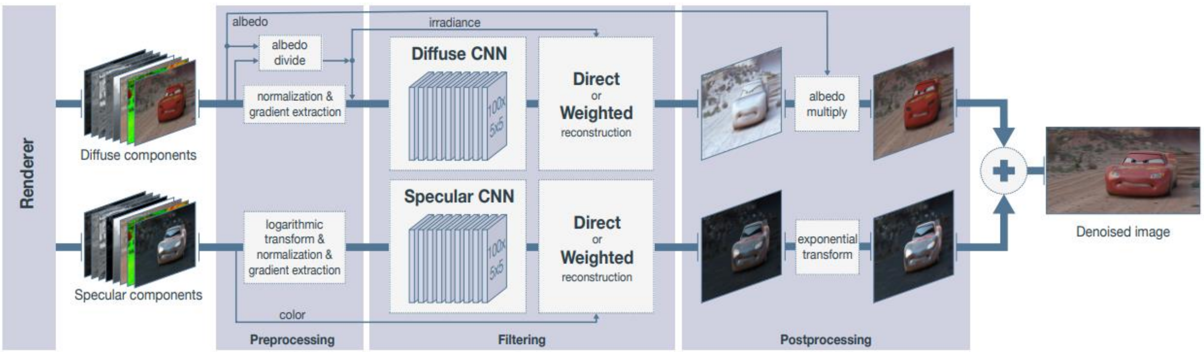
TRAINING

Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, **Bako et al.**

SIGGRAPH ASIA 2018 TOKYO

22

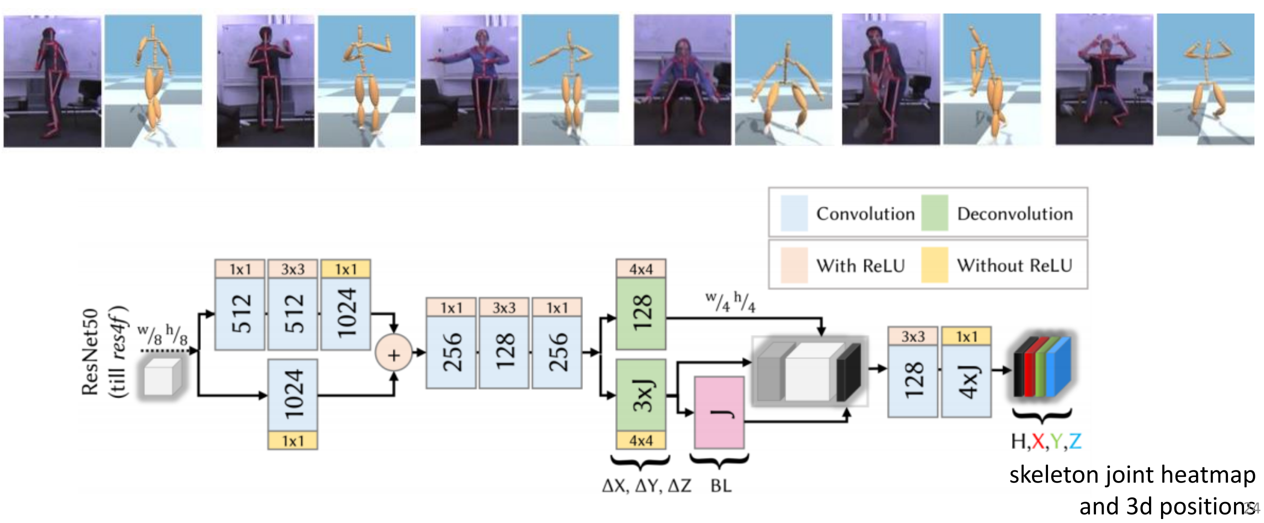
Denoising Renderings:



Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, Bako et al. **SIGGRAPH 2017**

23

3D Pose Estimation: *VNECT*

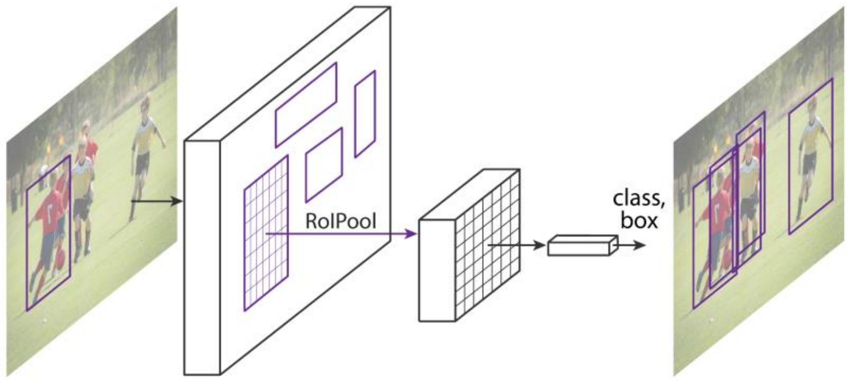


VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera, Mehta et al., **SIGGRAPH 2017**

24

Object Detection: Fast(er)-RCNN

- Fast/Faster R-CNN
  - ✓Good speed
  - ✓Good accuracy
  - ✓Intuitive
  - ✓Easy to use

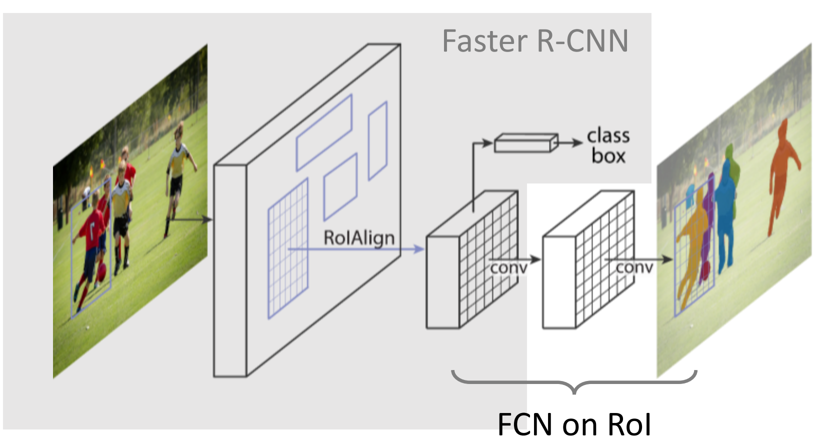


Ross Girshick. "Fast R-CNN". ICCV 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

SIGGRAPH ASIA 2018 TOKYO

Mask R-CNN

- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



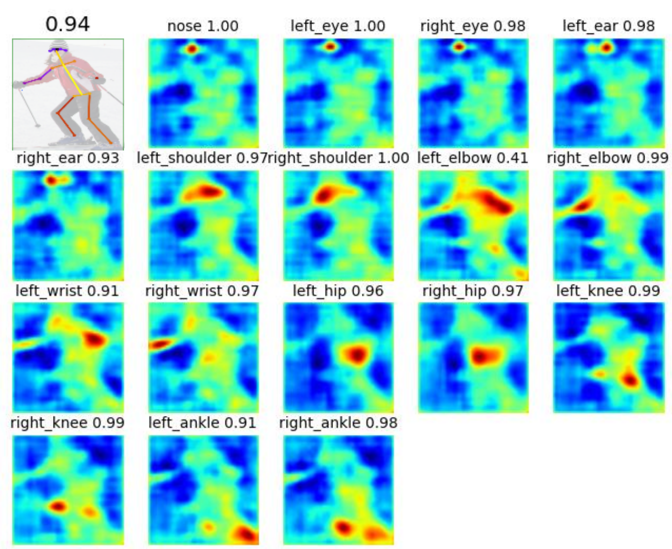
SIGGRAPH ASIA 2018 TOKYO

Mask R-CNN results on COCO

SIGGRAPH ASIA 2018 TOKYO

Mask R-CNN for Human Keypoint Detection

- 1 keypoint = 1-hot "mask"
- Human pose = 17 masks
- Softmax over spatial locations
  - e.g. 56^2-way softmax on 56x56

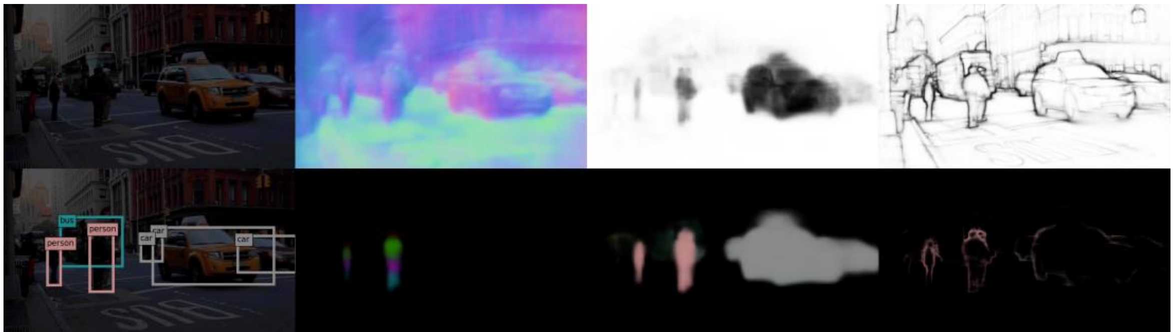


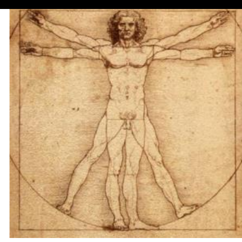
SIGGRAPH ASIA 2018 TOKYO


Mask R-CNN frame-by-frame


Mask R-CNN frame-by-frame

UberNet: a “universal” network for all tasks





https://github.com/jkokkin/UberNet

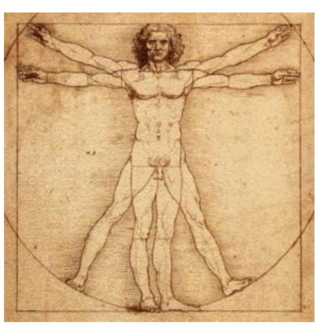
I. Kokkinos, UberNet: Training a Universal CNN for *Low- Mid- and High-Level* Vision, CVPR 2017

SIGGRAPH ASIA 2018 TOKYO

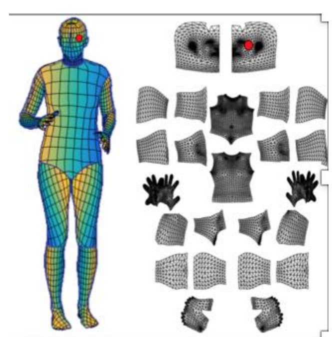
What is the ultimate vision task?

“Inverse graphics”: understand how an image was generated from a scene

If we focus on a single object category: surface-based models



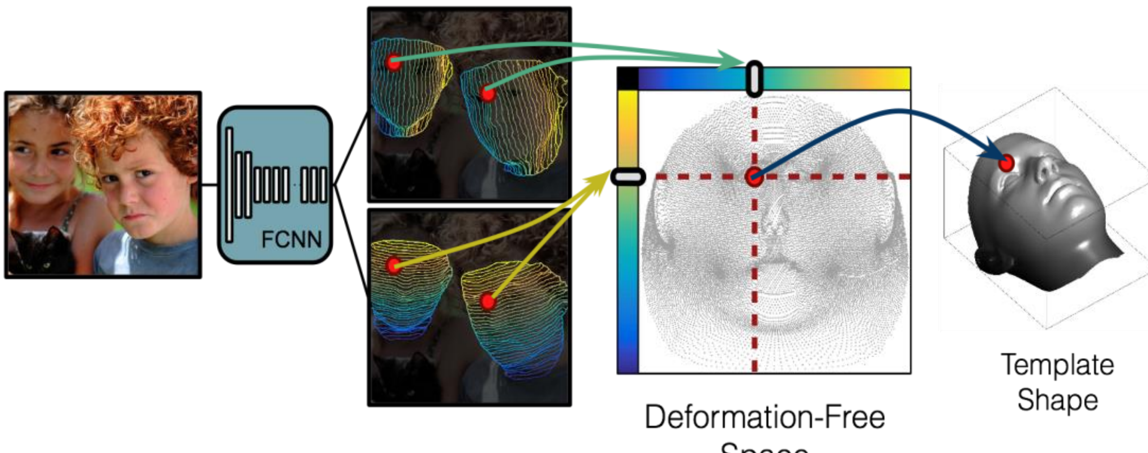
UberNet:
Universal Network



DensePose:
Unified model

SIGGRAPH ASIA 2018 TOKYO

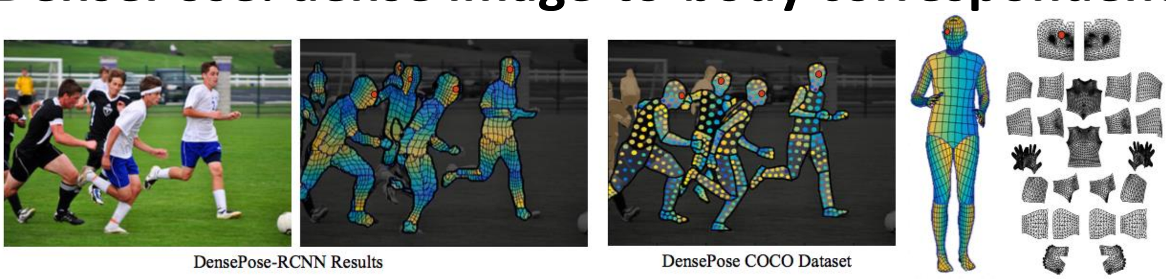
DenseReg: dense image-to-face regression

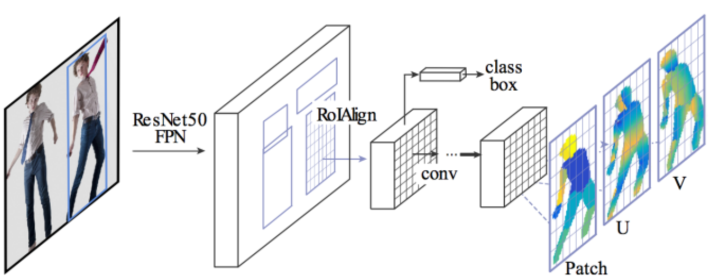


R. A. Guler, G. Trigeorgis, E. Antonakos, P. Snape, S. Zafeiriou, I. Kokkinos,
DenseReg: Fully Convolutional Dense Shape Regression In-the-Wild, CVPR 2017

SIGGRAPH ASIA 2018 TOKYO

DensePose: dense image-to-body correspondence

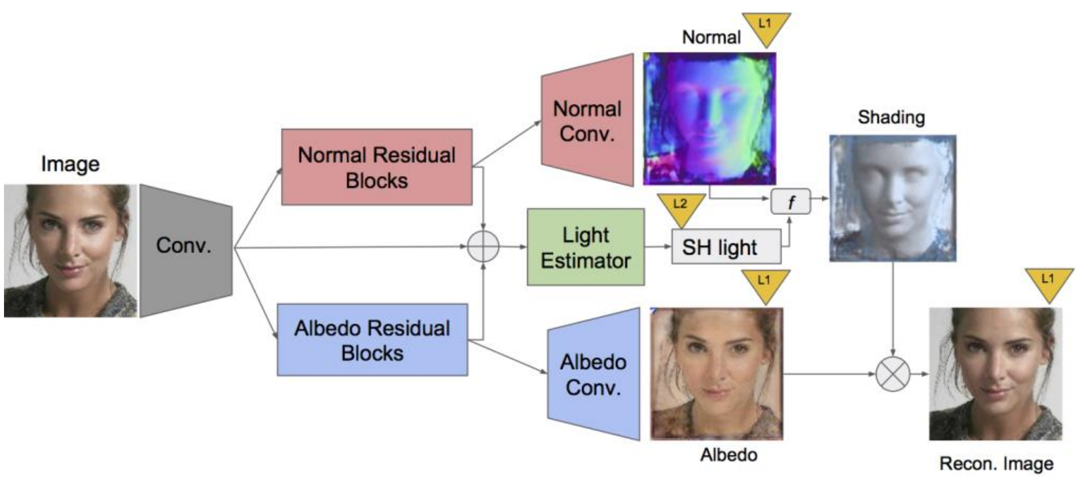




DensePose-RCNN: ~25 FPS

R. A. Guler, N. Neverova, I. Kokkinos “DensePose: Dense Human Pose Estimation In The Wild”, CVPR’18

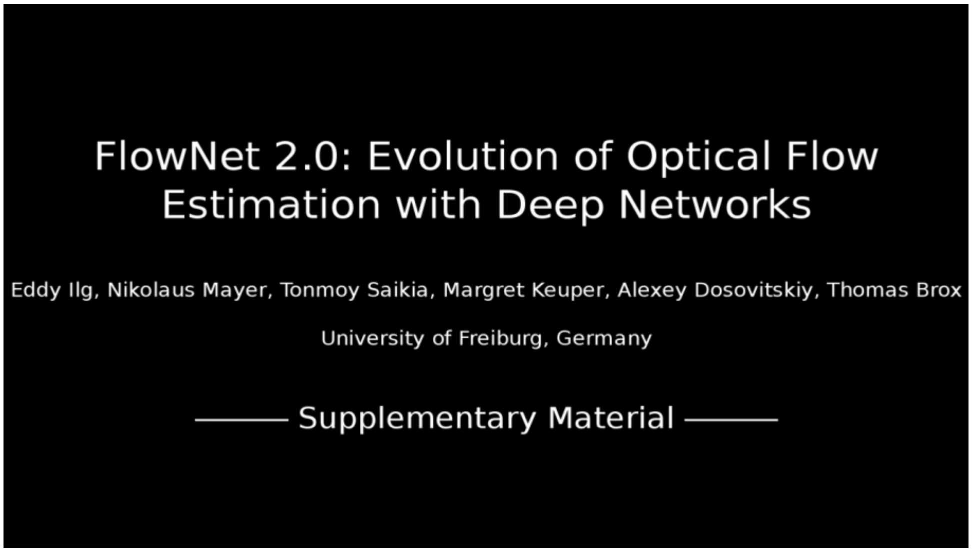
SFSNet: incorporating image formation in model



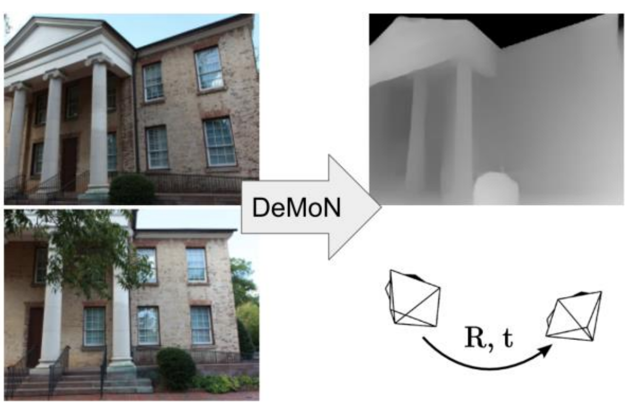
SfSNet: Learning Shape, Reflectance and Illuminance of Faces 'in the wild' Soumyadip Sengupta Angjoo Kanazawa Carlos D. Castillo David W. Jacobs, CVPR 2018

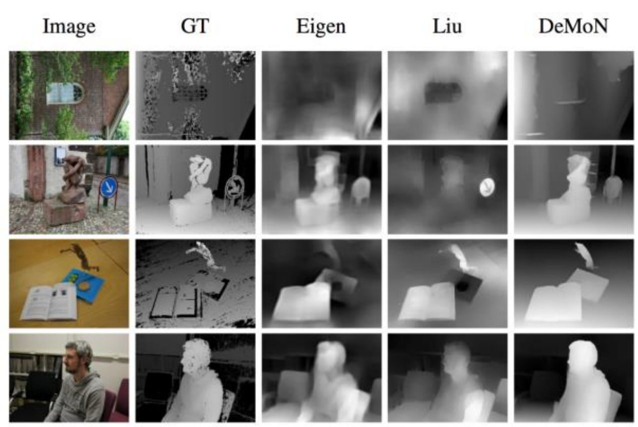
SIGGRAPH ASIA 2018 TOKYO

Beyond single frames: end-to-end optical flow



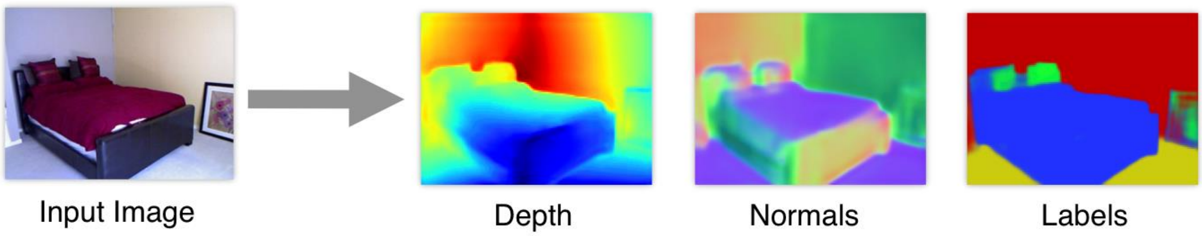
End-to-end Structure From Motion





- DeMoN: Depth and Motion Network for Learning Monocular Stereo, B. Ummenhofer, et al, CVPR 2017
- Unsupervised learning of depth and ego-motion from video, T Zhou, M Brown, N Snavely, DG Lowe, CVPR 2017

Monocular depth & normal estimation



- D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics**

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

3D Domains

Niloy Mitra UCL

Iasonas Kokkinos UCL/Facebook

Paul Guerrero UCL

Nils Thuerey TU Munich

Tobias Ritschel UCL

UCL

facebook Artificial Intelligence Research

TUM Technische Universität München

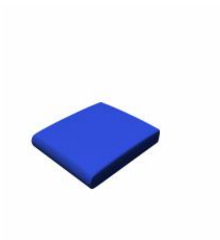
Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics** 2

Motivating Applications



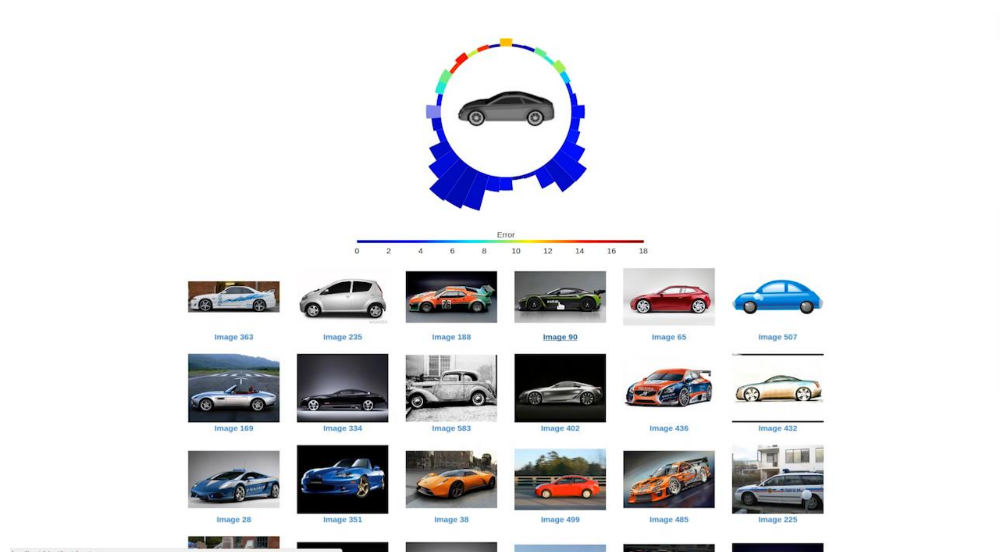
Deep neural network predicts the **next best part** to add and its **position** to enable non-expert users to create novel shapes.

[Sung et al. 2017]

SIGGRAPH ASIA 2018 TOKYO

3

CrossLink: Linking Images and 3D Models



SIGGRAPH ASIA 2018 TOKYO

[Heuting et al. 2015]

4

Motivating Applications

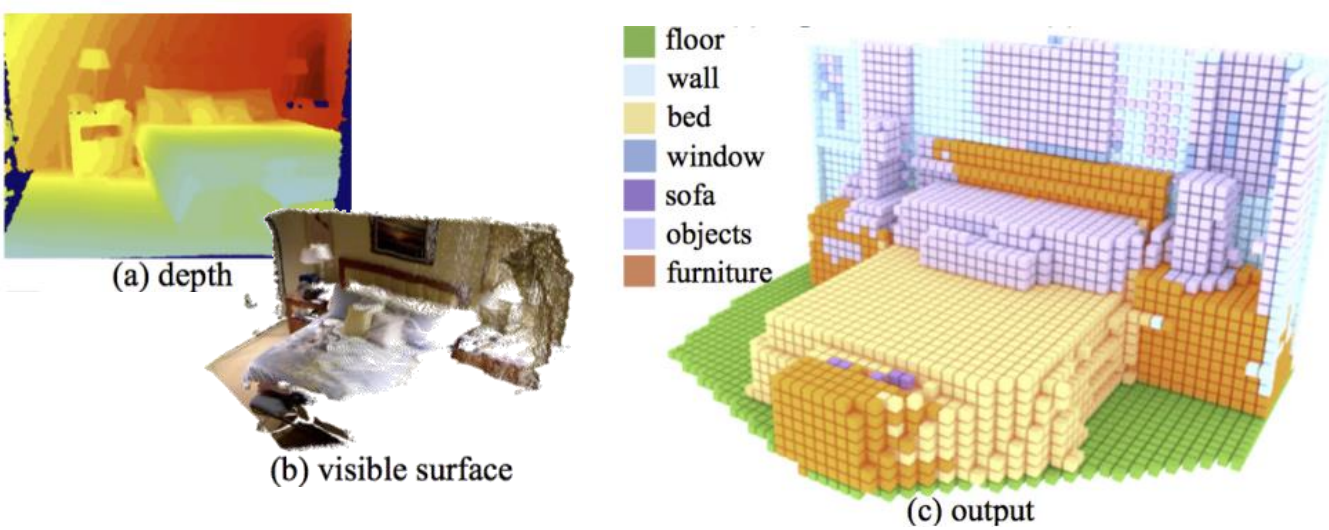
understanding 3D shapes can benefit image understanding

Physically based Rendering

[Zhang et al. 2017]

5

Motivating Applications: Semantic Scene Understanding

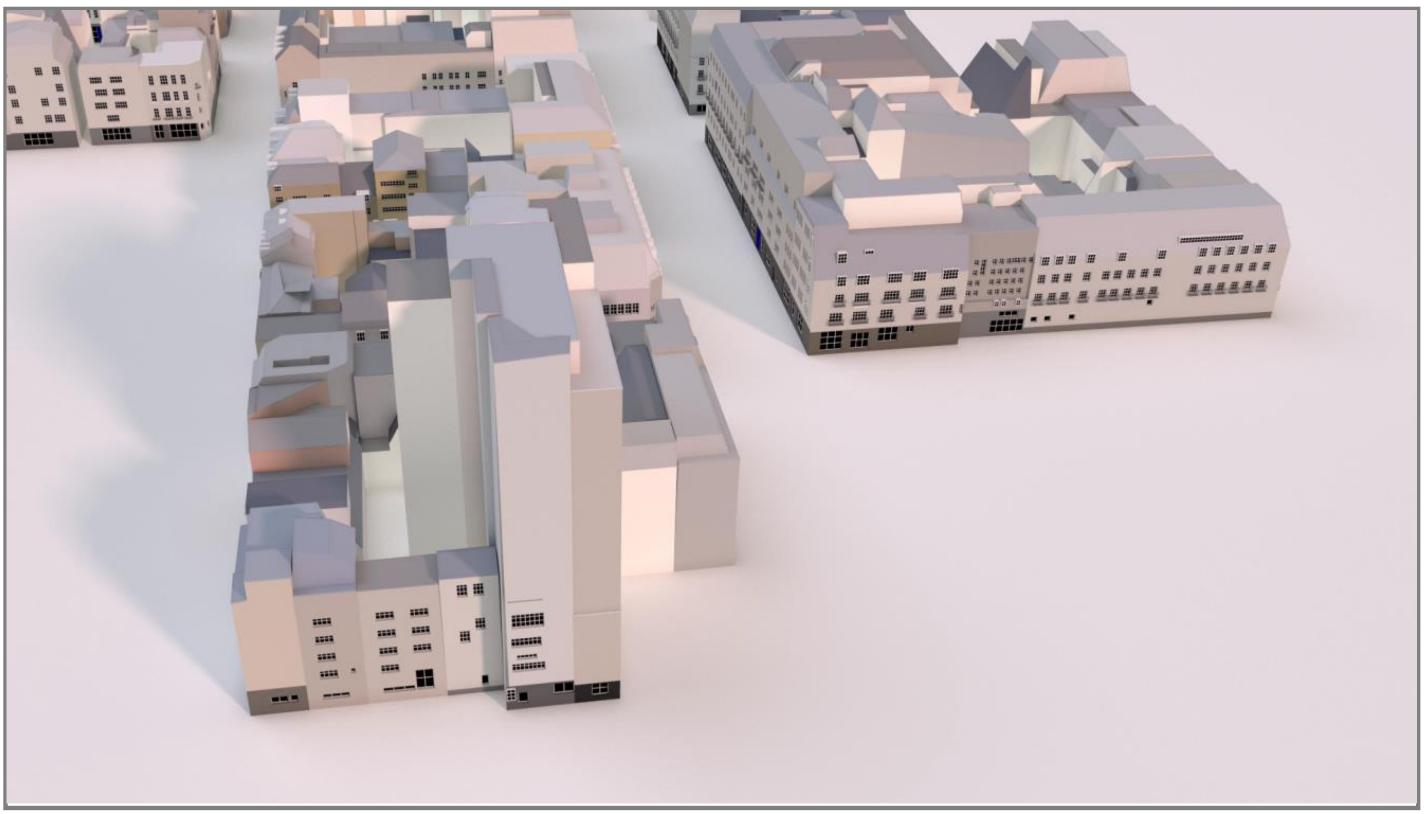


[Song et al. 2017]

6

Motivating Applications: Semantic Scene Understanding

[Kelly et al. 2017]



Representation for 3D

- Image-based
- Volumetric
- Point-based
- Surface-based
- Parametric

SIGGRAPH ASIA 2018 TOKYO

9

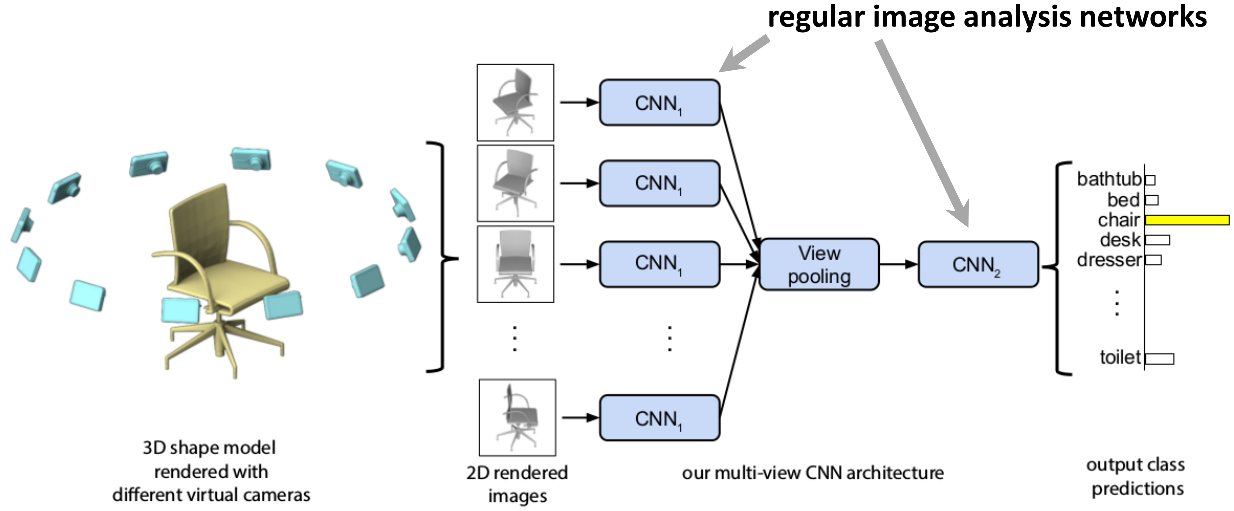
Representation for 3D

- **Image-based**
- Volumetric
- Point-based
- Surface-based
- Parametric

SIGGRAPH ASIA 2018 TOKYO

10

Representation for 3D: Multi-view CNN

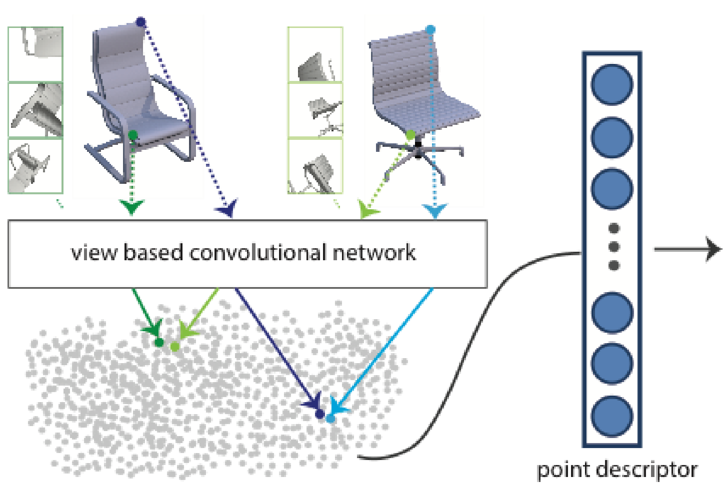


[Kalogerakis et al. 2015]

SIGGRAPH ASIA 2018 TOKYO

11

Representation for 3D: Local Multi-view CNN



Segmentation
Correspondence
Feature matching
Predicting semantic functions

localized renderings for point-wise features

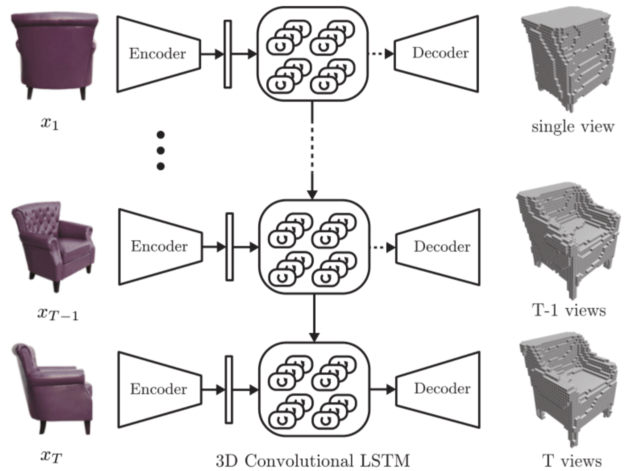
[Huang et al. 2018]

SIGGRAPH ASIA 2018 TOKYO

12

3D-R^{2}N^2 (3D Recurrent Reconstruction Neural Network)

- Multiple views are treated as image sequence
- An LSTM controls what part of the latent representation is updated by each view



Choy et al. *3d-r2n2: A unified approach for single and multi-view 3d object reconstruction*. ECCV 2016

Representation for 3D

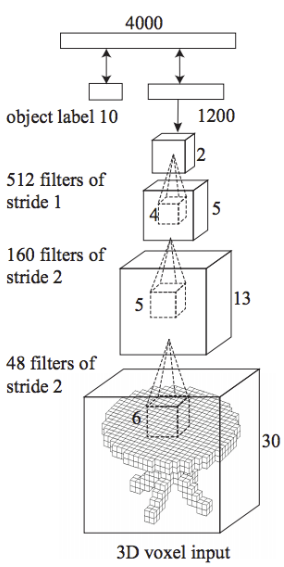
- **Image-based**
  - PROS: directly use image networks, good performance
  - **CONS**: rendering is slow and memory-heavy, not very geometric
- Volumetric
- Point-based
- Surface-based
- Parametric

Representation for 3D

- Image-based
- **Volumetric**
- Point-based
- Surface-based
- Parametric

1
5

Representation for 3D: Volumetric

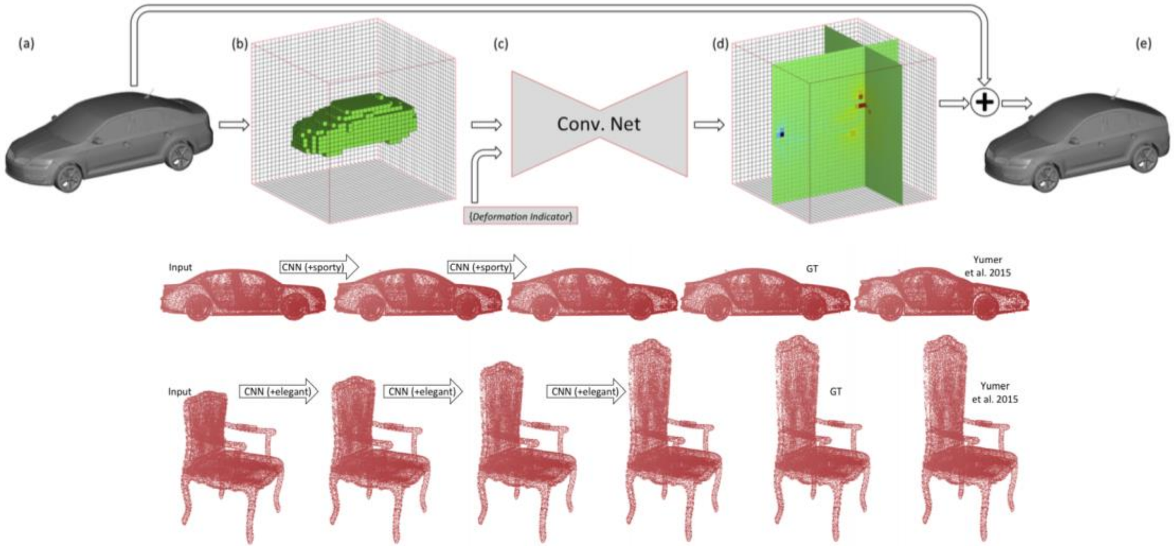


- Add one dimension to kernels and intermediate outputs: batches x channels x w x h batches x channels x d x w x h
- Does not scale well to high resolutions

[Xiao et al. 2014]

16

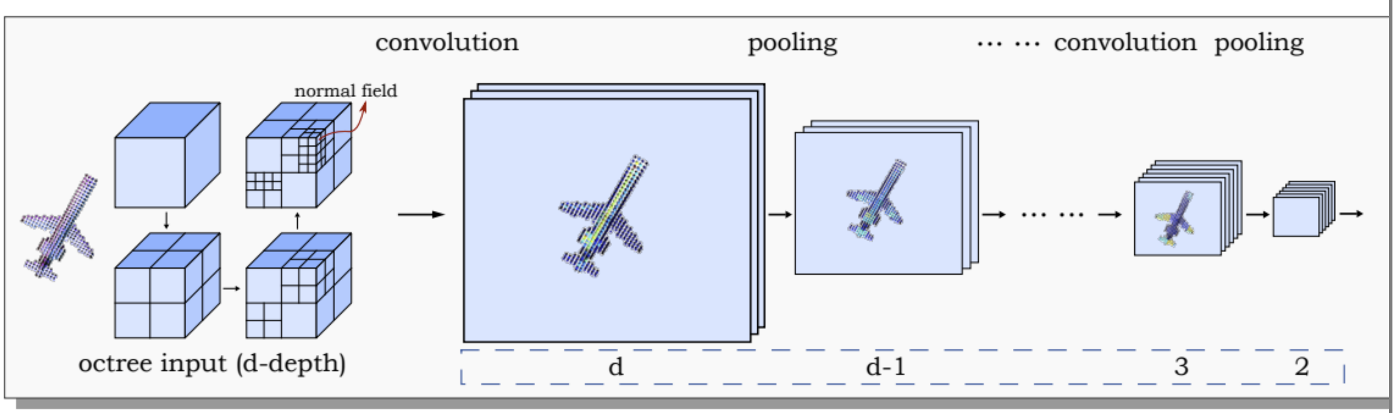
Representation for 3D: Volumetric Deformation



[Yumer et al. 2014]

17

Efficient Volumetric Datastructures



[Wang et al. 2017]

18

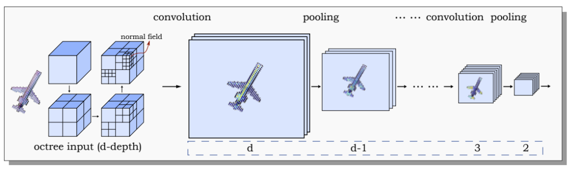
Efficient Volumetric Datastructures

Generator / Decoder

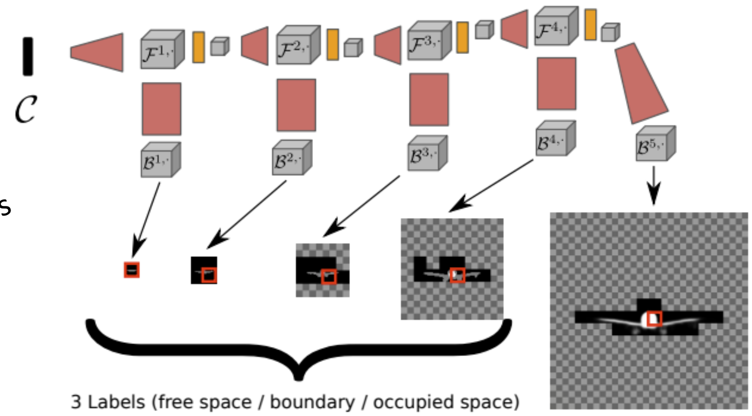
Volumetric (Up-) Convolutions

Cropping

Encoder



Wang et al. 2017

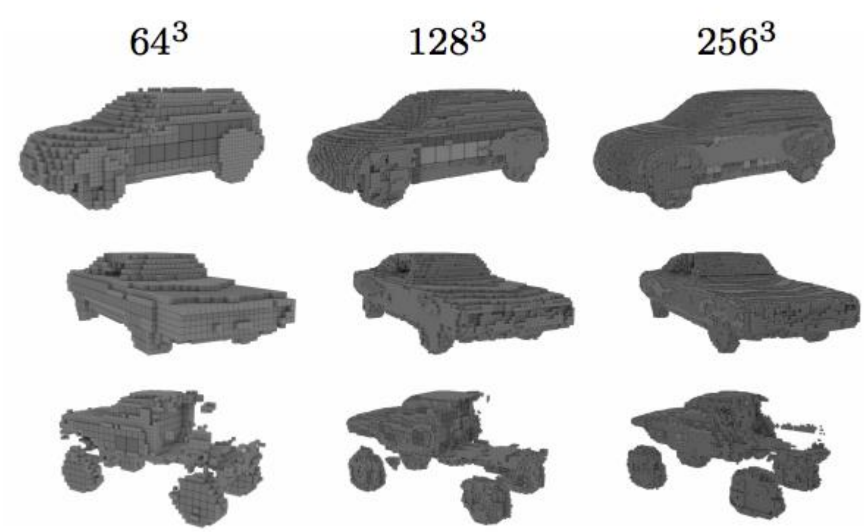


only generate non-empty voxels

[Hane et al. 2018]

19

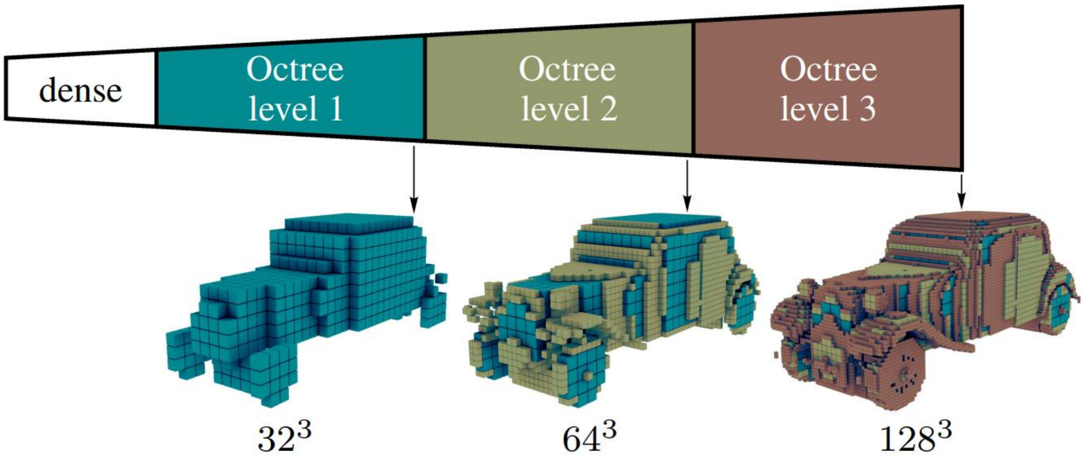
Efficient Volumetric Datastructures



[Hane et al. 2018]

20

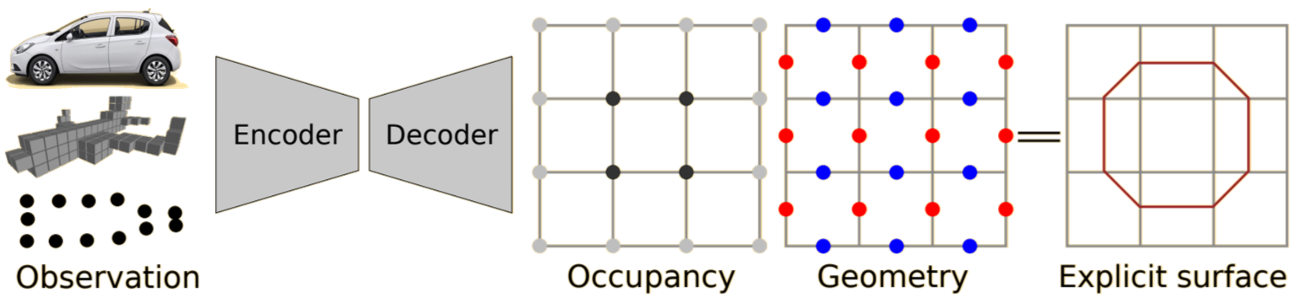
Octree Generating Networks



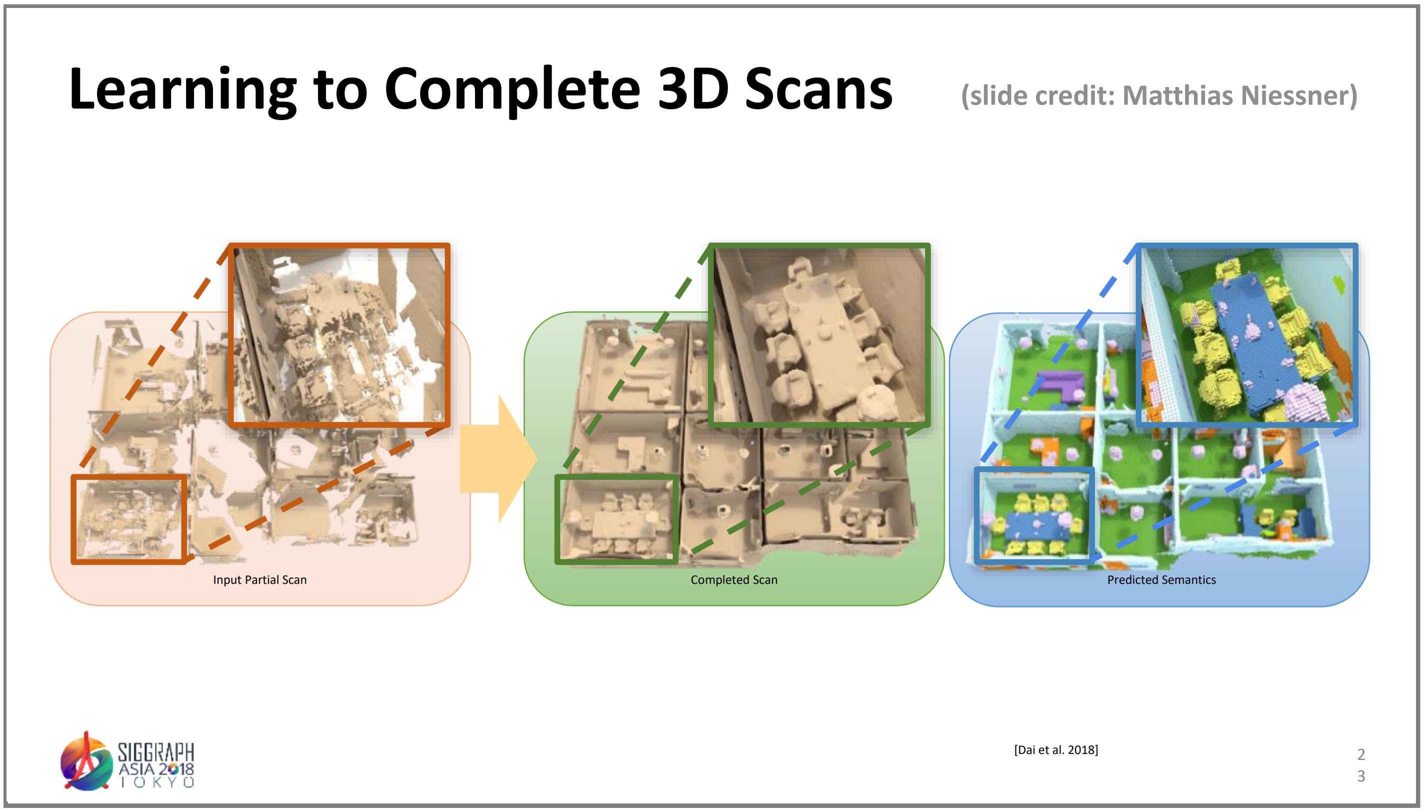
Tatarchenko et al. *Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs*. ICCV 2017

Deep Marching Cubes

- Input Domain: images, volumetric grids, point clouds
- Output Domain: Meshes

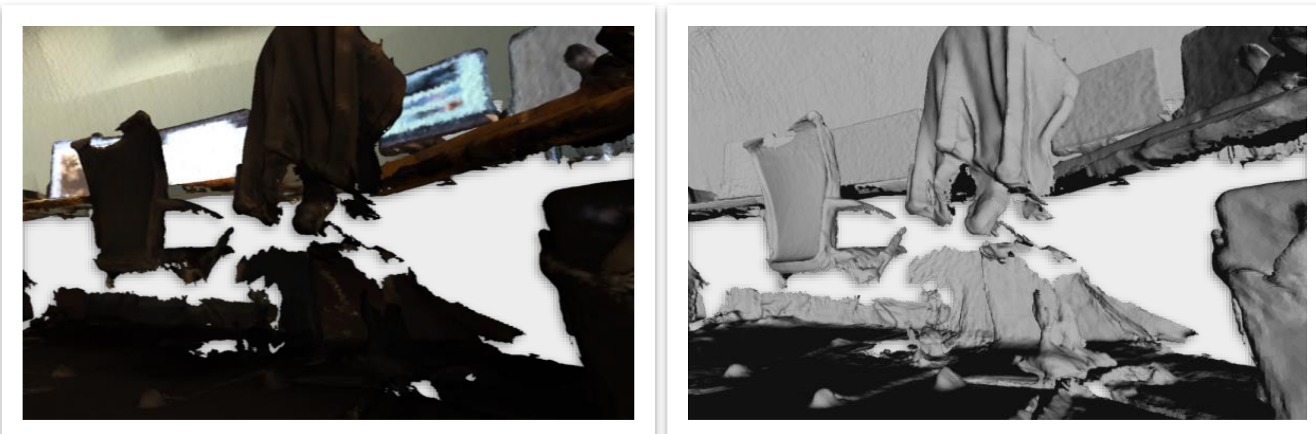


Yiyi Liao et al. *Deep Marching Cubes.* CVPR 2018


Learning to Complete 3D Scans
(slide credit: Matthias Niessner)
Input Partial Scan
Completed Scan
Predicted Semantics
SIGGRAPH ASIA 2018 TOKYO
[Dai et al. 2018]
23


(slide credit: Matthias Niessner)
State-of-the-art 3D Reconstructions
SIGGRAPH ASIA 2018 TOKYO
24
TOG'17 [Dai et al.]: BundleFusion

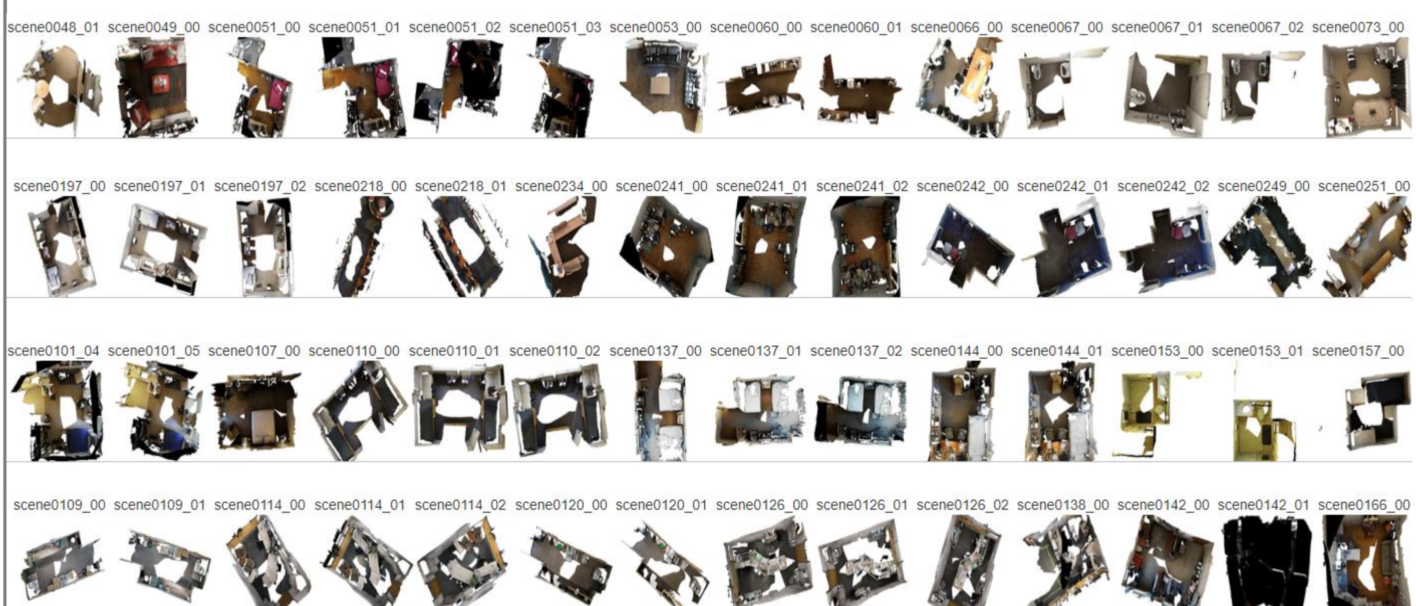
Problem: Incomplete Scan Geometry



SIGGRAPH ASIA 2018 TOKYO

25

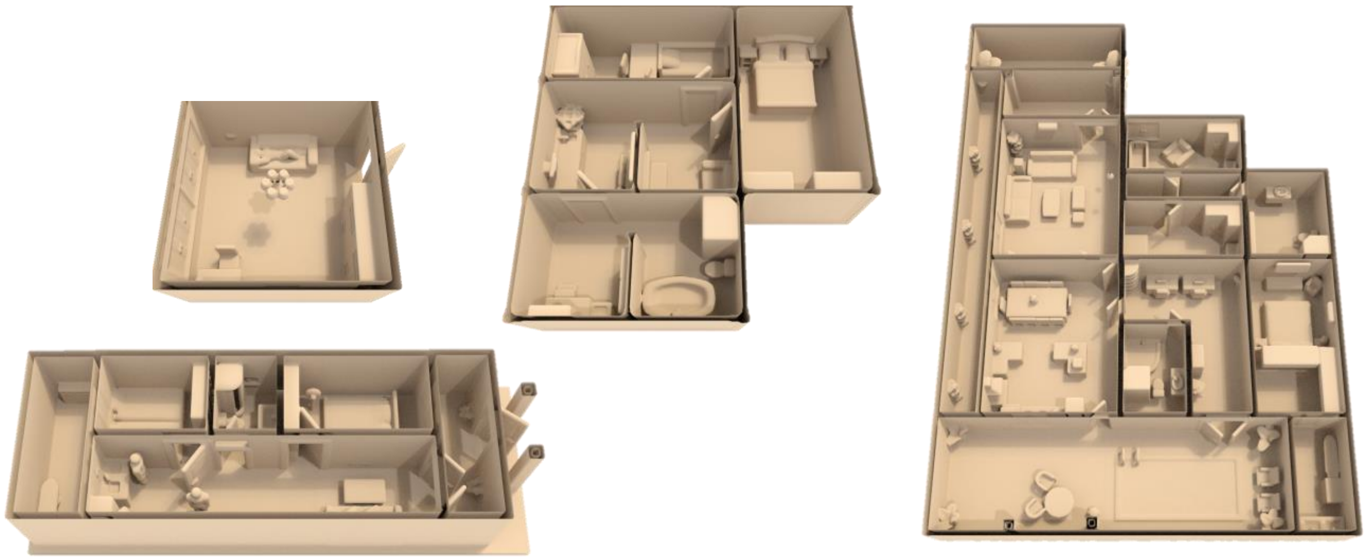
Problem: Incomplete Scan Geometry



ASIA 2018 TOKYO

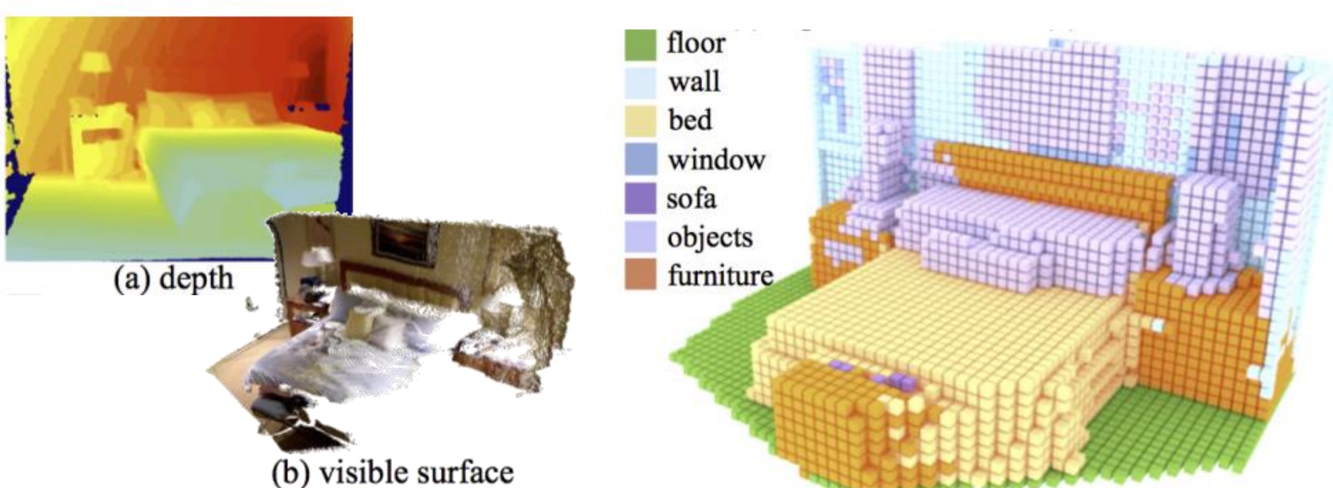
6

Learning from Synthetic Data



27

Recall: Semantic Scene Understanding

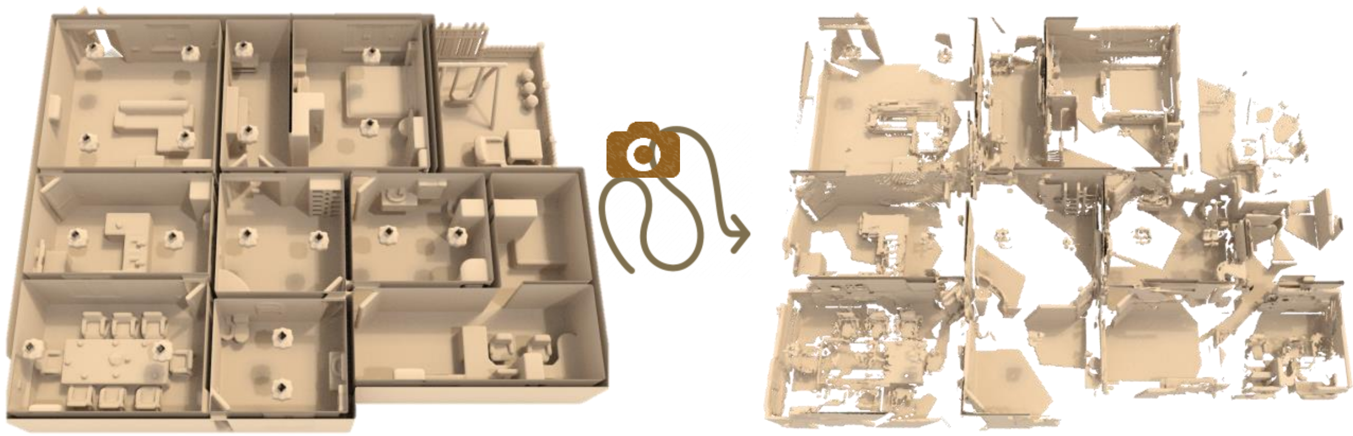


[Song et al. 2017]

28

Learning to Complete 3D Scans

(slide credit: Matthias Niessner)

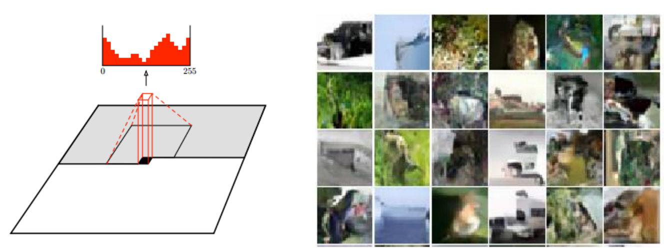


Scenes from SUNCG [Song et al. 17]

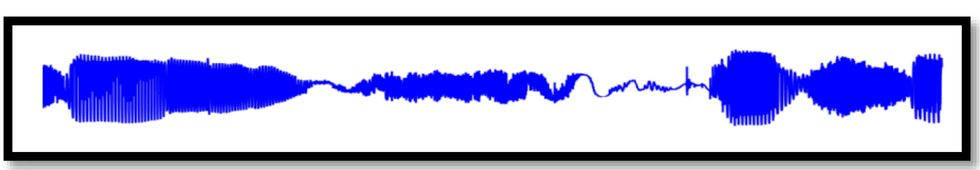
29

Dependent Predictions: Autoregressive Neural Networks

- PixelCNN [van den Oord 2015, van den Oord 2016, Reed 2017]

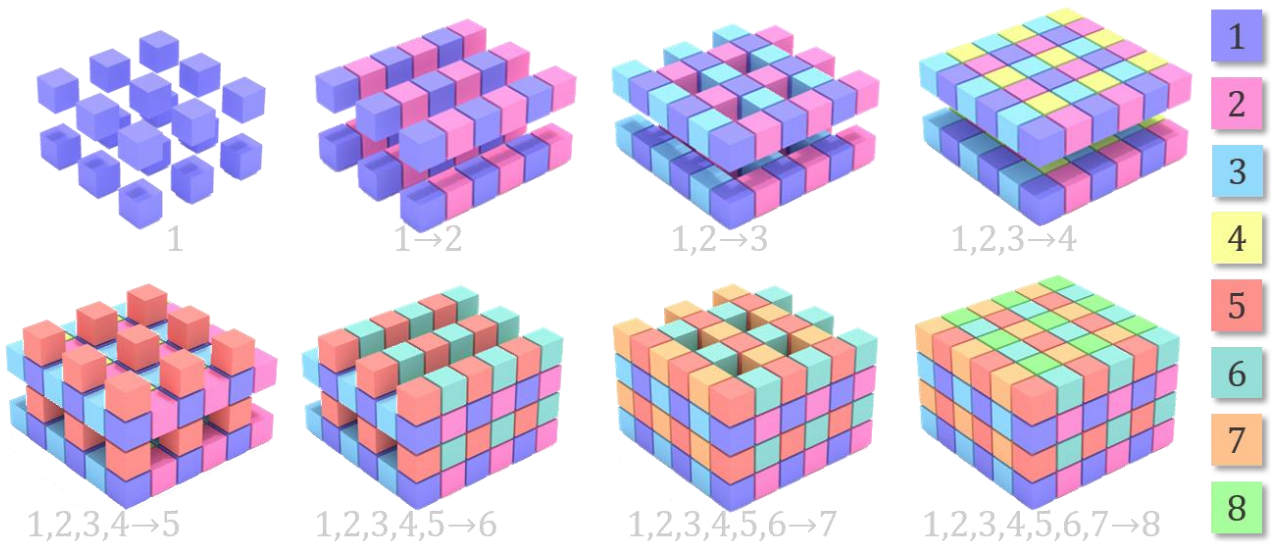


- WaveNet [van den Oord 2016]



30

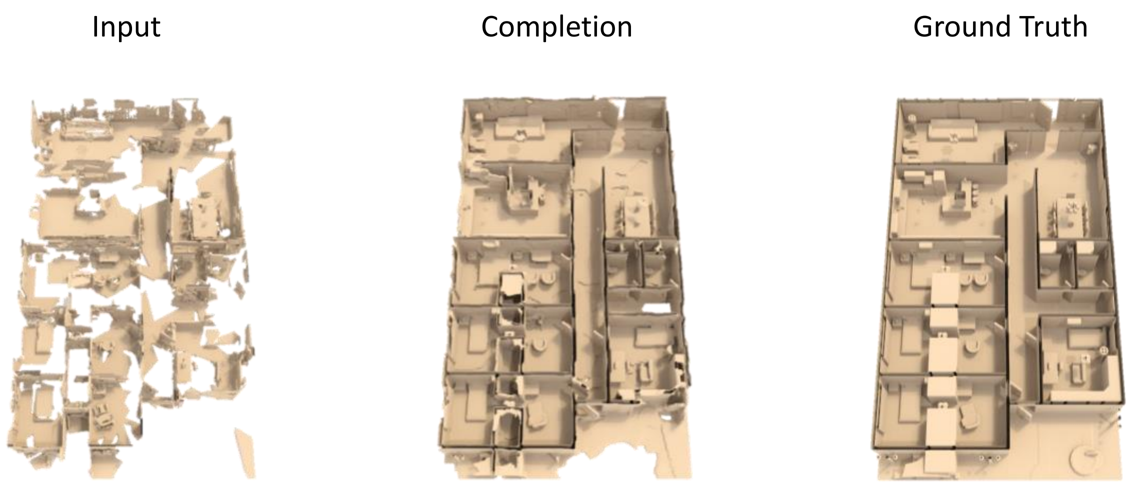
Dependent Predictions: Autoregressive Neural Networks



[Dai et al. 2018]

31

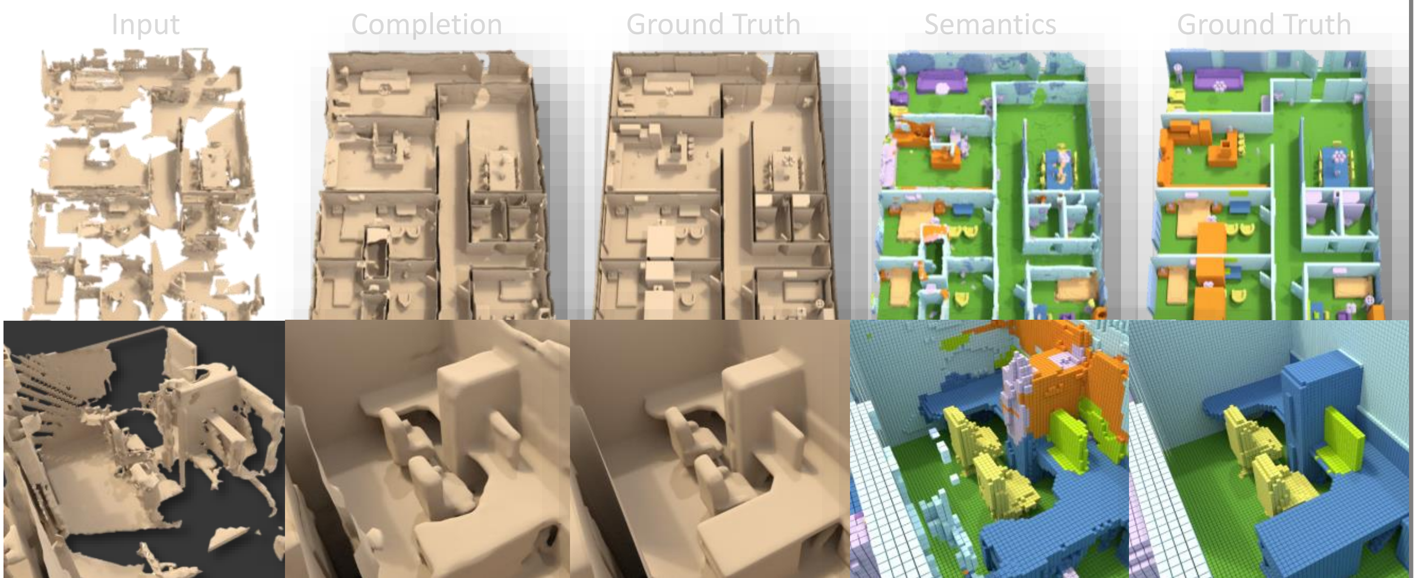
ScanComplete: Completing 3D Scans



[Dai et al. 2018]

32

ScanComplete: Completing 3D Scans

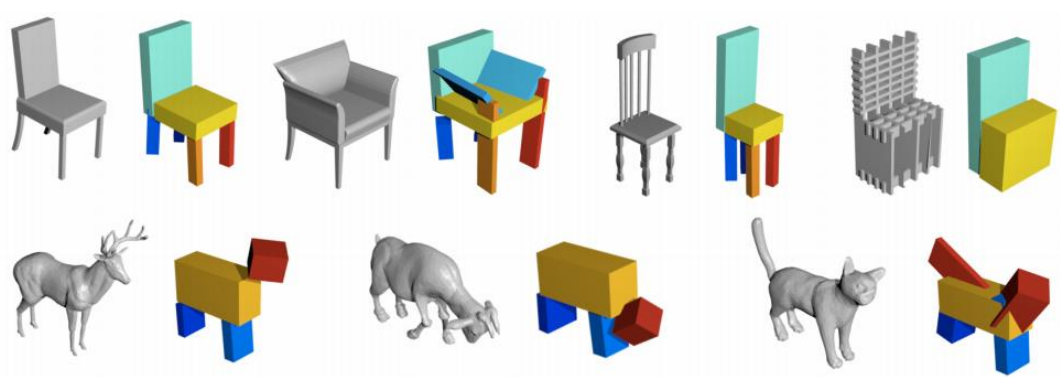


[Dai et al. 2018]

SIGGRAPH ASIA 2018 TOKYO

33

Geometry Abstraction / Simplification

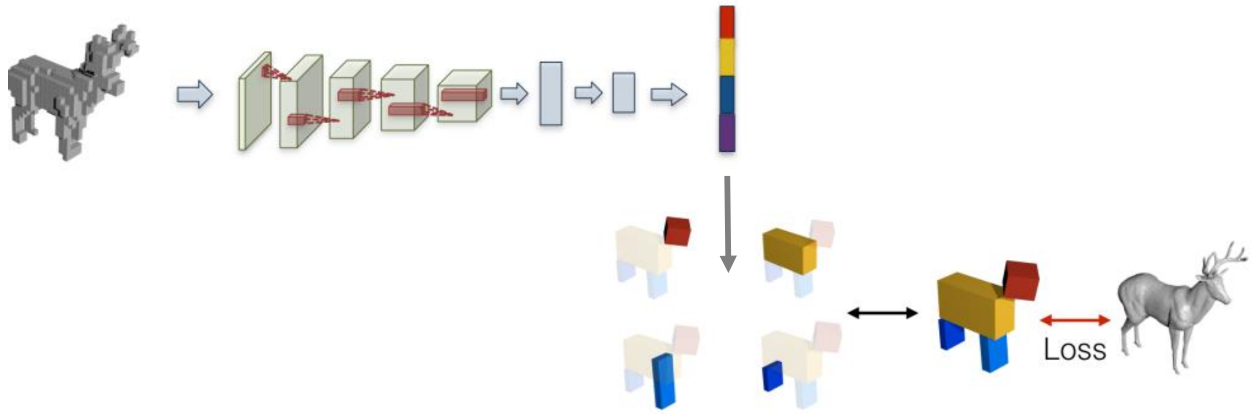


Learning Shape Abstractions by Assembling Volumetric Primitives, Tulsiani et al. 2016

SIGGRAPH ASIA 2018 TOKYO

34

Geometry Abstraction / Simplification:

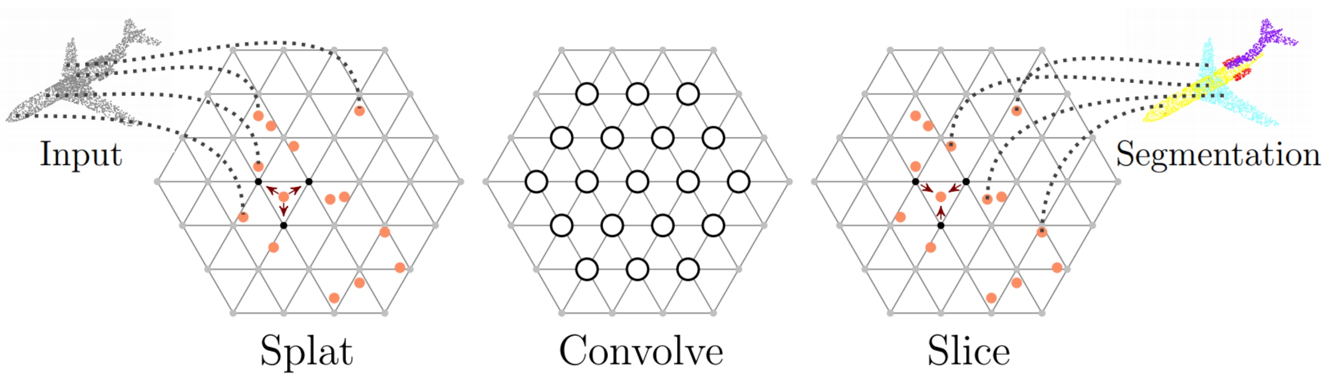


Learning Shape Abstractions by Assembling Volumetric Primitives, Tulsiani et al. 2016

SIGGRAPH ASIA 2018 TOKYO

35

SplatNet



Hang Su et al. *Splatnet: Sparse lattice networks for point cloud processing.* CVPR 2018

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics**

36

Representation for 3D

- Image-based
- Volumetric
  - **PROS**: modify image networks
  - **CONS**: special layers for hierarchical datastructures, still too coarse
- Point-based
- Surface-based
- Parametric

SIGGRAPH ASIA 2018 TOKYO

37

Representation for 3D

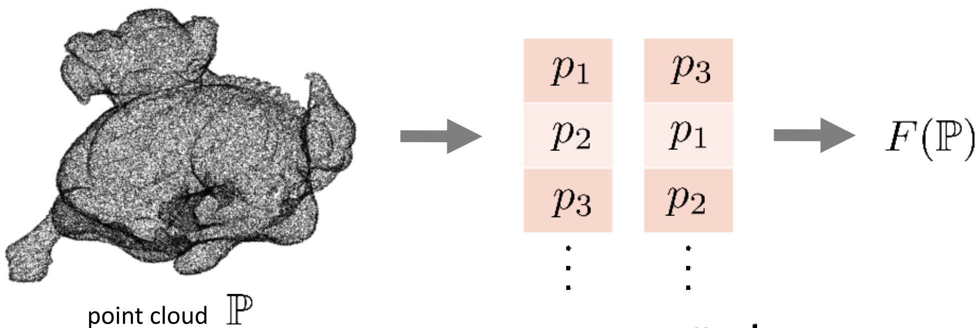
- Image-based
- Volumetric
- **Point-based**
- Surface-based
- Parametric

SIGGRAPH ASIA 2018 TOKYO

38

Point Clouds

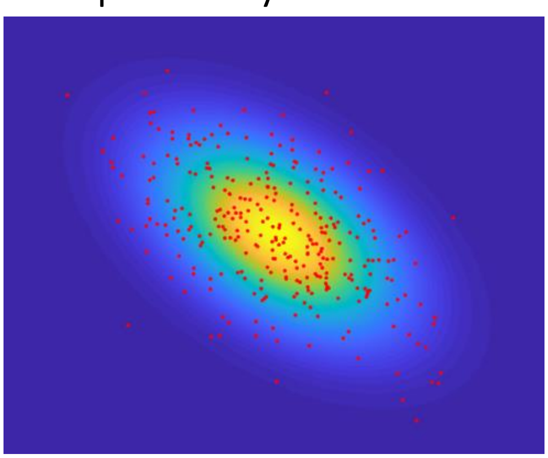
- Common representation
- Easy to obtain from meshes, depth scans, laser scans
- Difficulty: invariance to point order



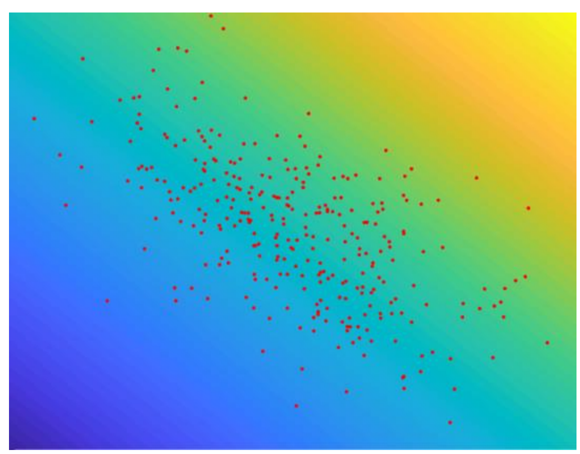
invariance to all $n!$ cases

Point Interpretation

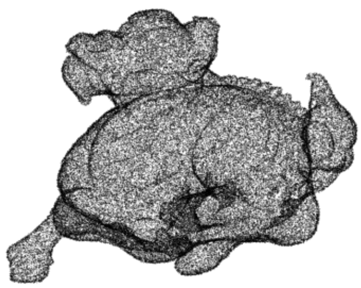
Samples from
a probability distribution



(Irregular) samples of
a continuous function



PointNet



point cloud $\mathbb{P}$

order-independent

p_1	p_3
p_2	p_1
p_3	p_2
⋮	⋮

$$f(p_3),\ f(p_1),\ f(p_2) \quad \sum_{p_j \in \mathbb{P}} g(f(p_j)) = F(\mathbb{P})$$

feature vector for a **point**

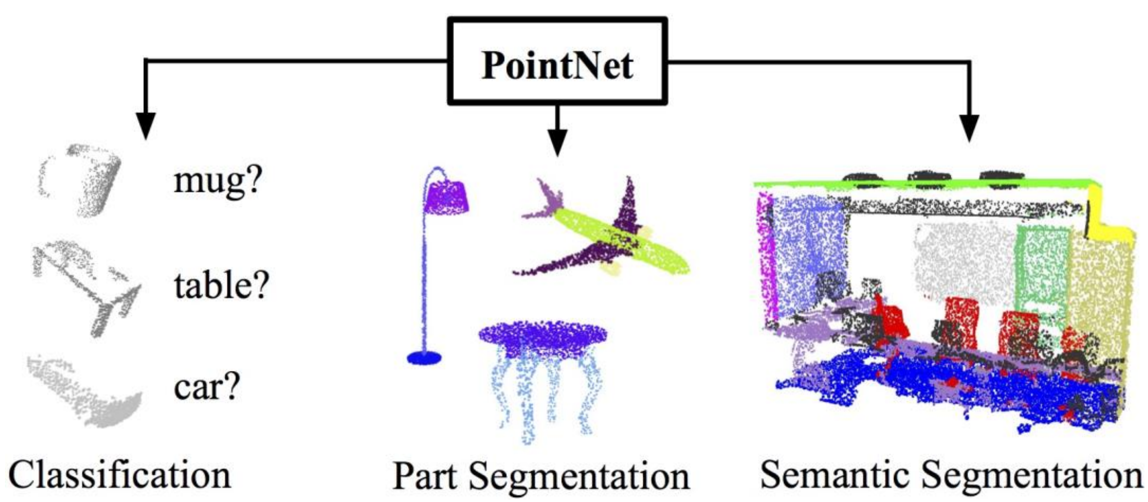
symmetric

order-independent

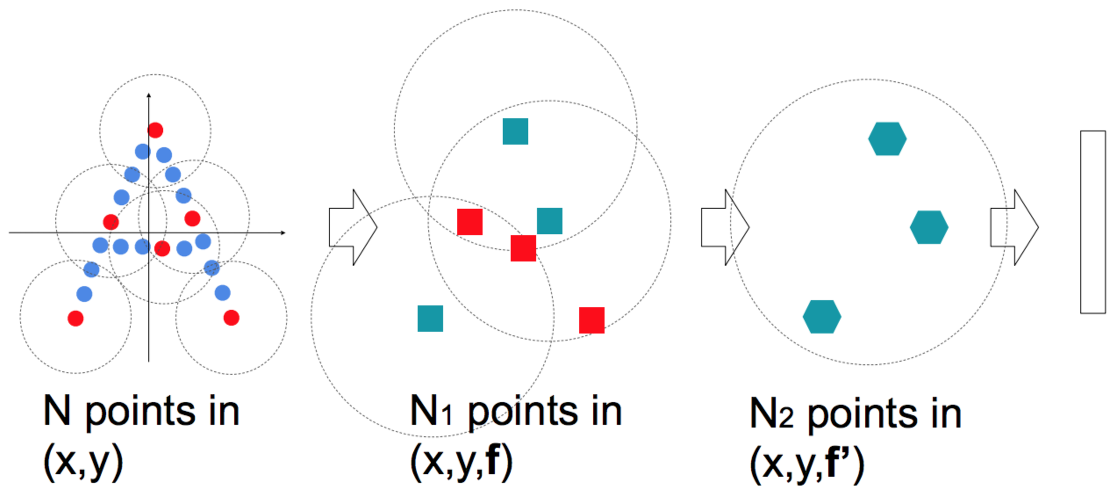
feature vector for the **point cloud**

Qi et al. *Pointnet: Deep learning on point sets for 3d classification and segmentation*. CVPR 2017

PointNet for Point Cloud Analysis



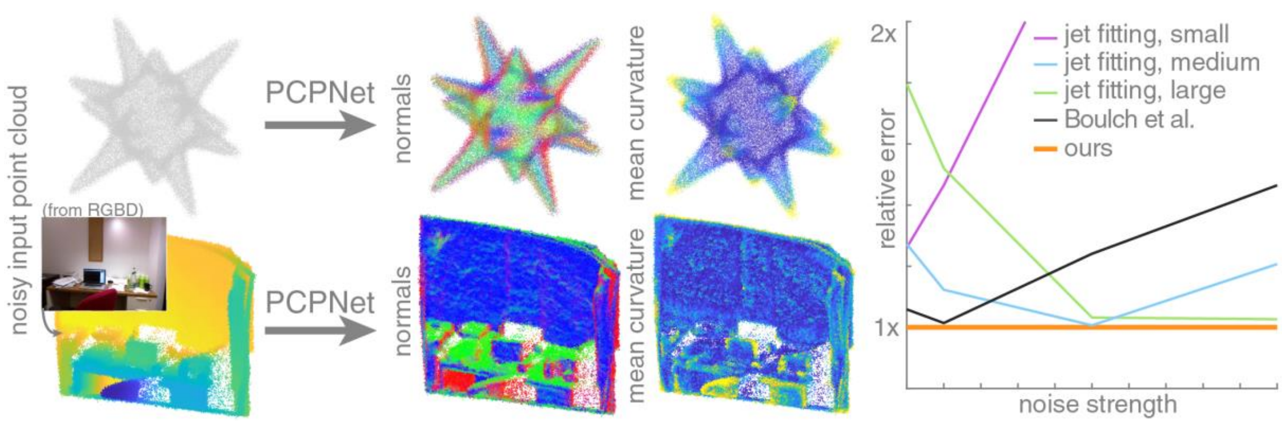
PointNet for Point Cloud Analysis: PointNet++



[Qi et al. 2018]

43

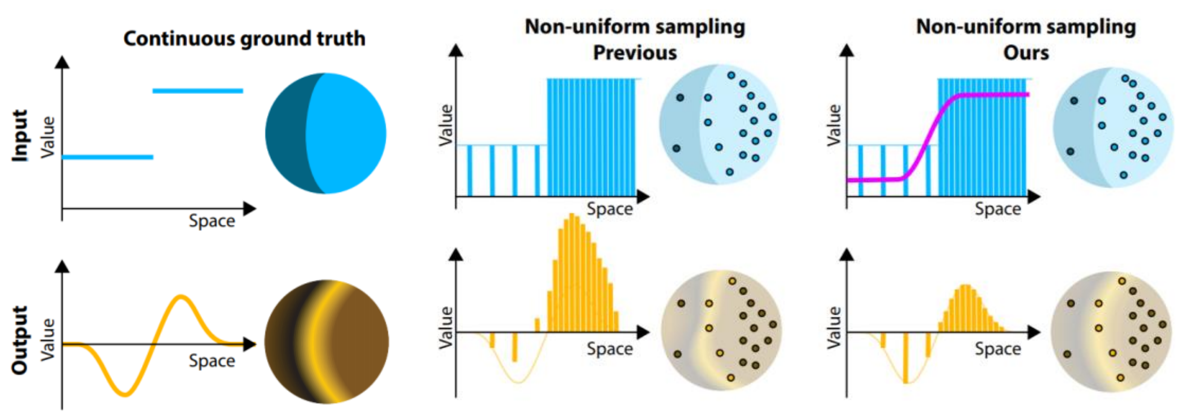
PointNet for Local Point Cloud Analysis



[Guerrero et al. 2018]

44

MCCNN



Hermosilla et al. *Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds*. SIGGRAPH Asia 2018

PointNet for Point Cloud Synthesis

generated output needs to be compare to some true shape

Earth Mover Distance as loss function

Input

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

[Su et al. 2017]

Representation for 3D

- Image-based
- Volumetric
- Point-based
- **Surface-based**

Surface models used in engineering (i.e., CAD) and computer graphics (i.e., meshes)

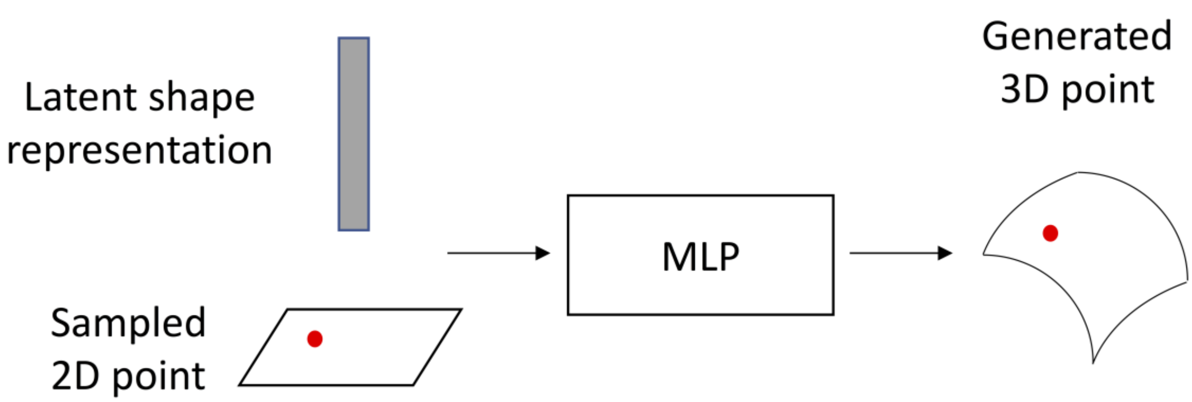
Image **Generated Volume** **Generated Points** **Generated Surface**

SIGGRAPH ASIA 2018 TOKYO

47

AtlasNet for Surface Generation

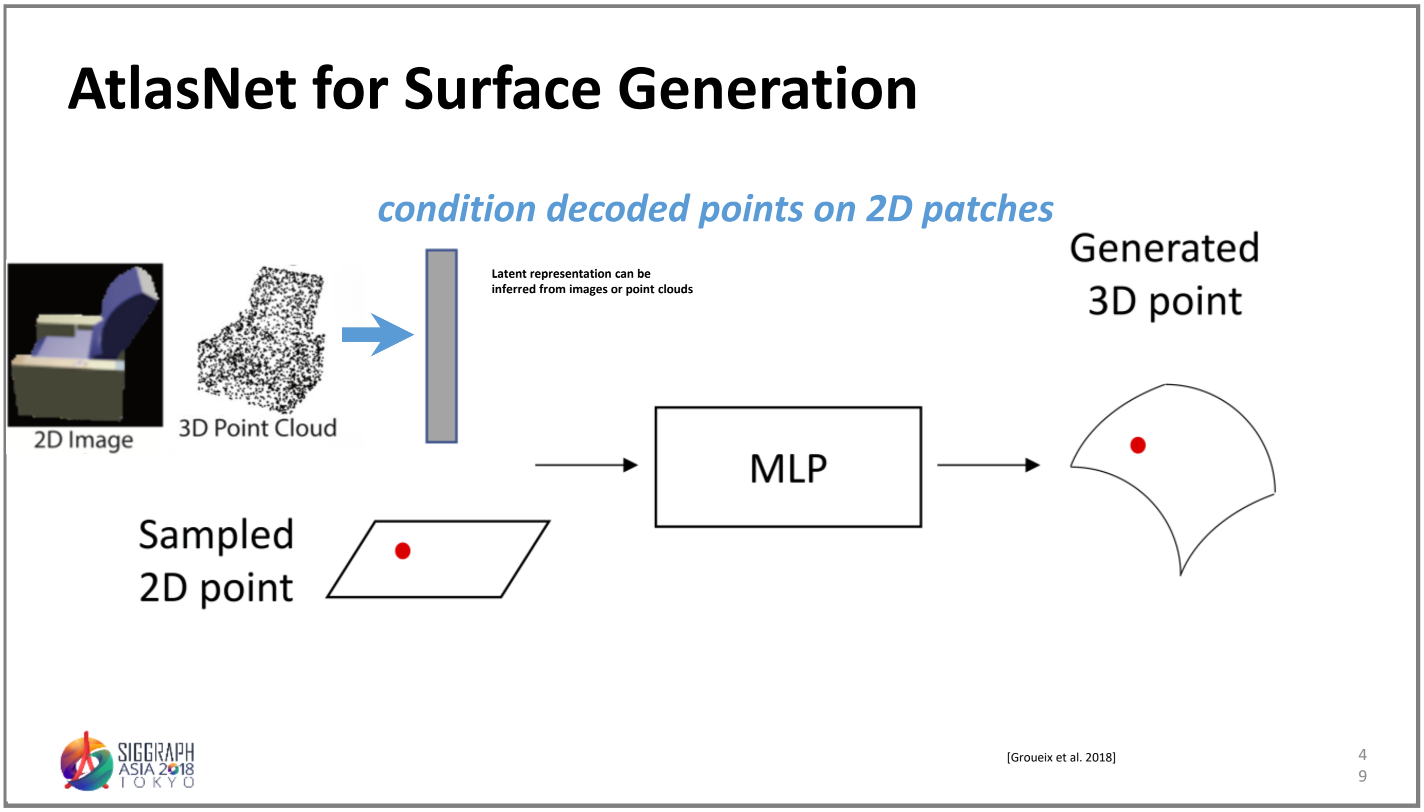
condition decoded points on 2D patches

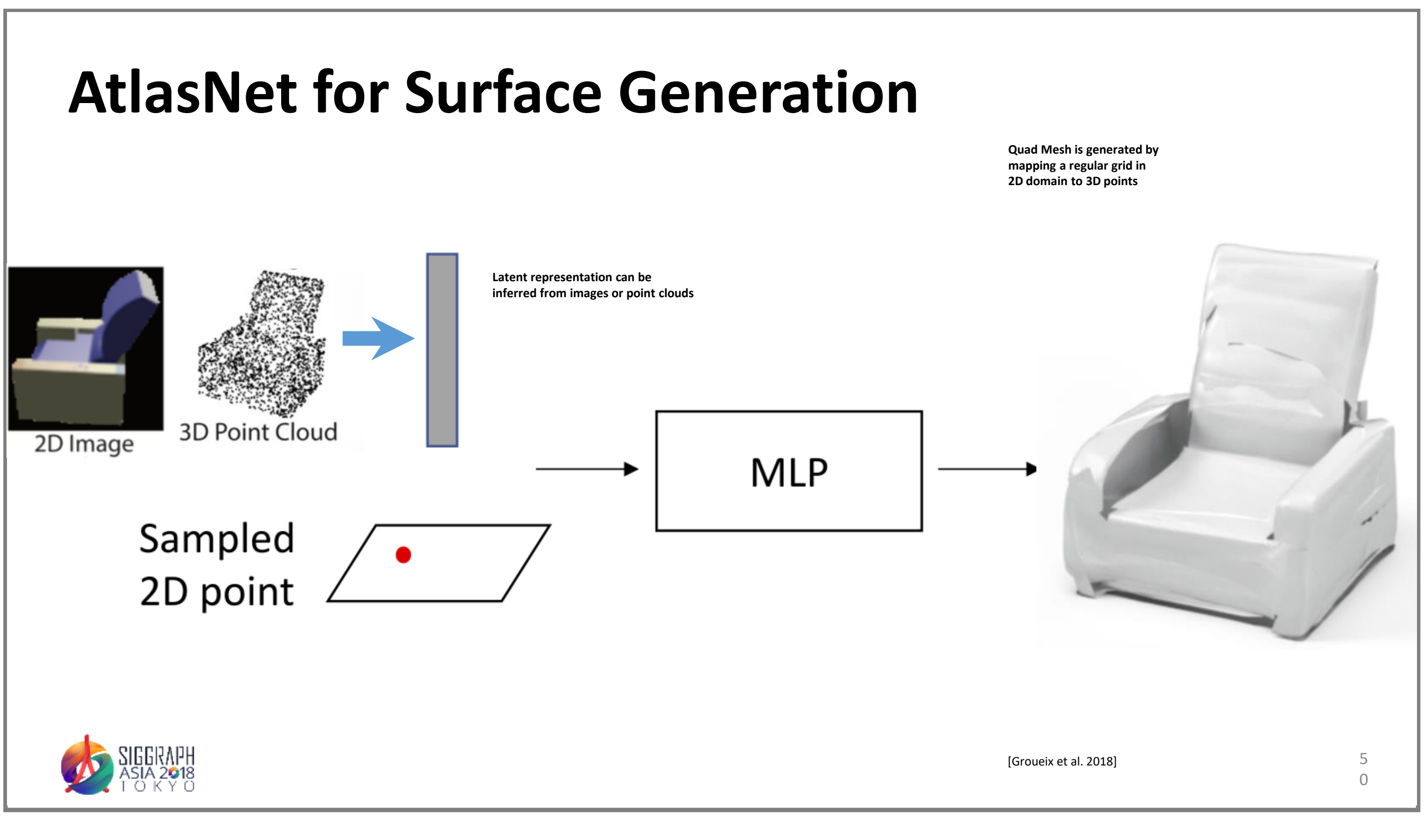


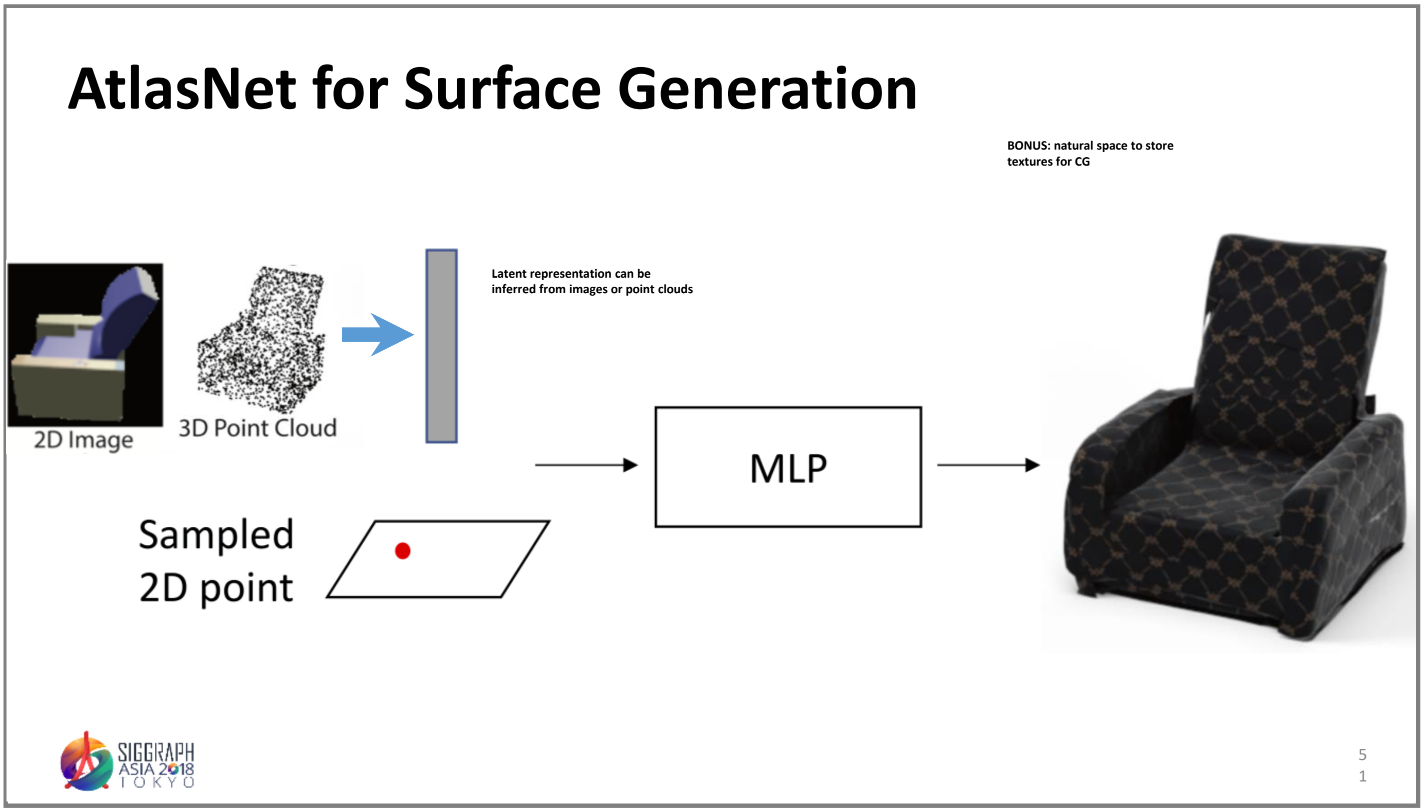
SIGGRAPH ASIA 2018 TOKYO

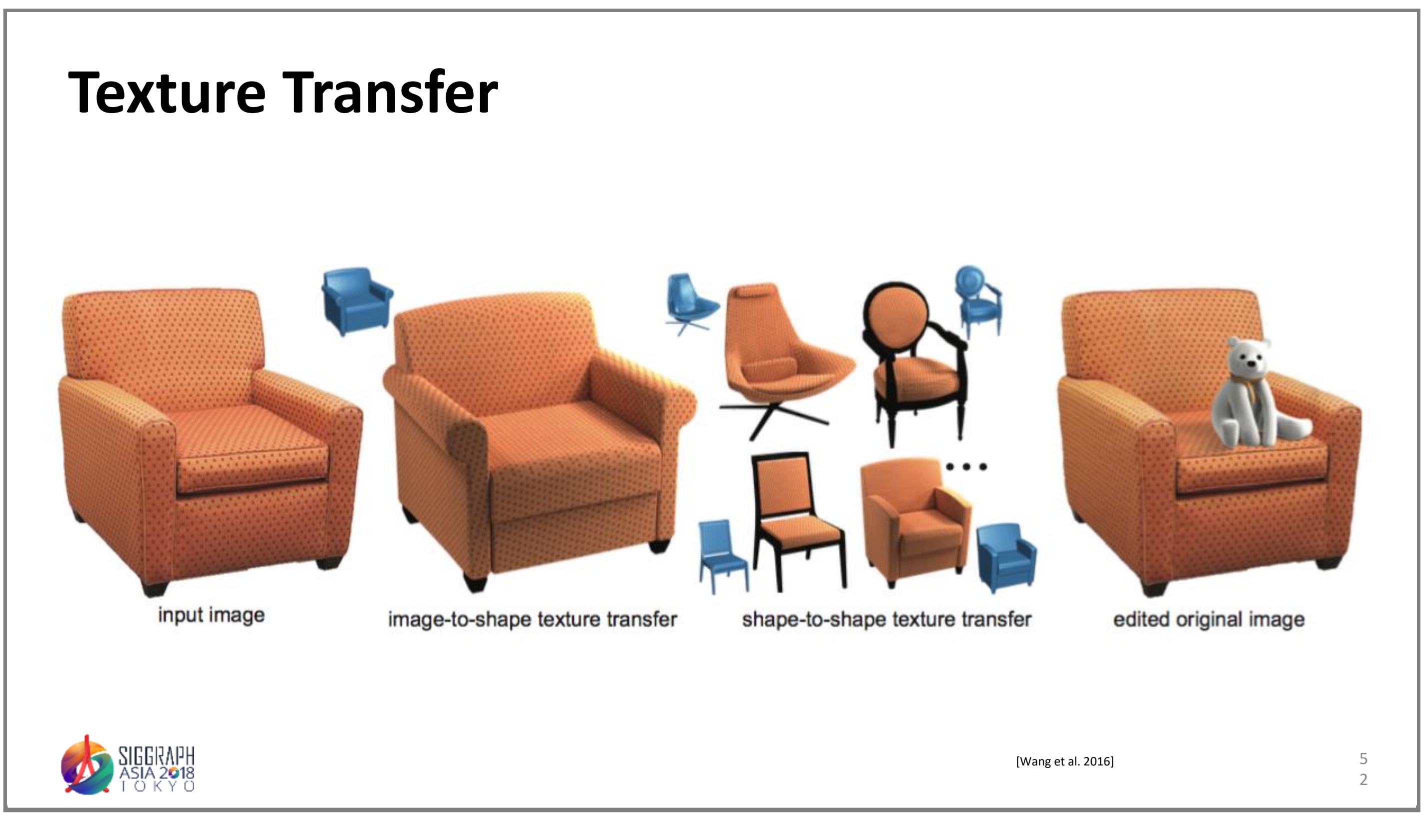
[Groueix et al. 2018]

48


AtlasNet for Surface Generation
condition decoded points on 2D patches
Generated 3D point
2D Image
3D Point Cloud
Latent representation can be inferred from images or point clouds
MLP
Sampled 2D point
SIGGRAPH ASIA 2018 TOKYO
[Groueix et al. 2018]
49

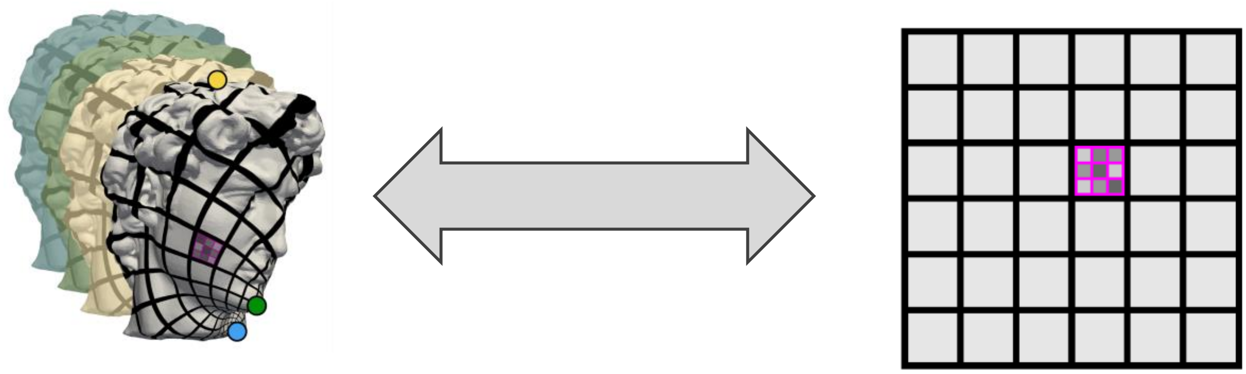

AtlasNet for Surface Generation
Quad Mesh is generated by mapping a regular grid in 2D domain to 3D points
2D Image
3D Point Cloud
Latent representation can be inferred from images or point clouds
MLP
Sampled 2D point
SIGGRAPH ASIA 2018 TOKYO
[Groueix et al. 2018]
50


AtlasNet for Surface Generation
BONUS: natural space to store textures for CG
Latent representation can be inferred from images or point clouds
2D Image
3D Point Cloud
MLP
Sampled 2D point
SIGGRAPH ASIA 2018 TOKYO
51


Texture Transfer
input image
image-to-shape texture transfer
shape-to-shape texture transfer
edited original image
[Wang et al. 2016]
SIGGRAPH ASIA 2018 TOKYO
52

Parameterization for Surface Analysis

map 3D surface to 2D domain



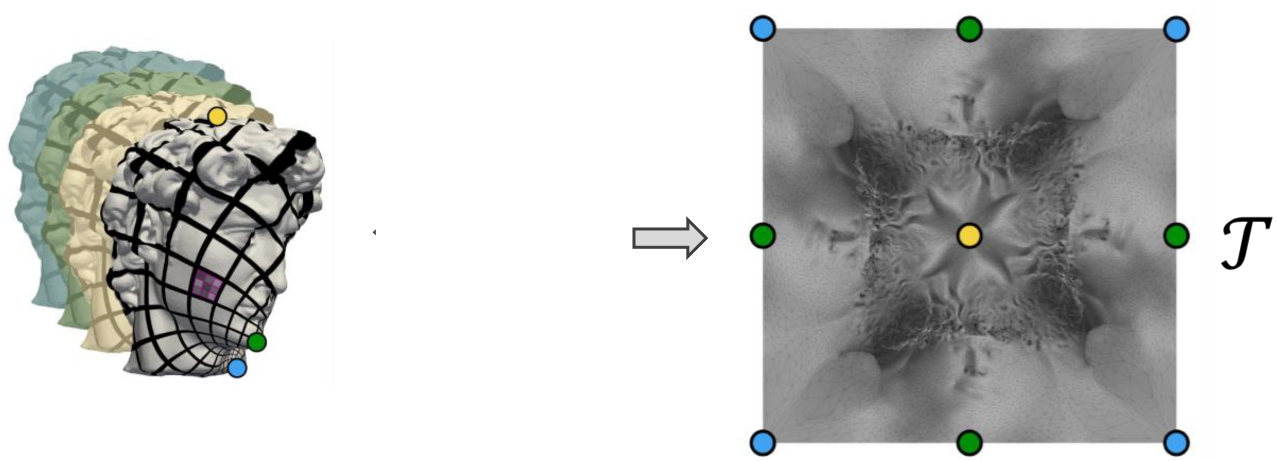
SIGGRAPH ASIA 2018 TOKYO

[Maron et al. 2017]

53

Parameterization for Surface Analysis

map 3D surface to 2D domain



SIGGRAPH ASIA 2018 TOKYO

[Maron et al. 2017]

54

Parameterization for Surface Analysis

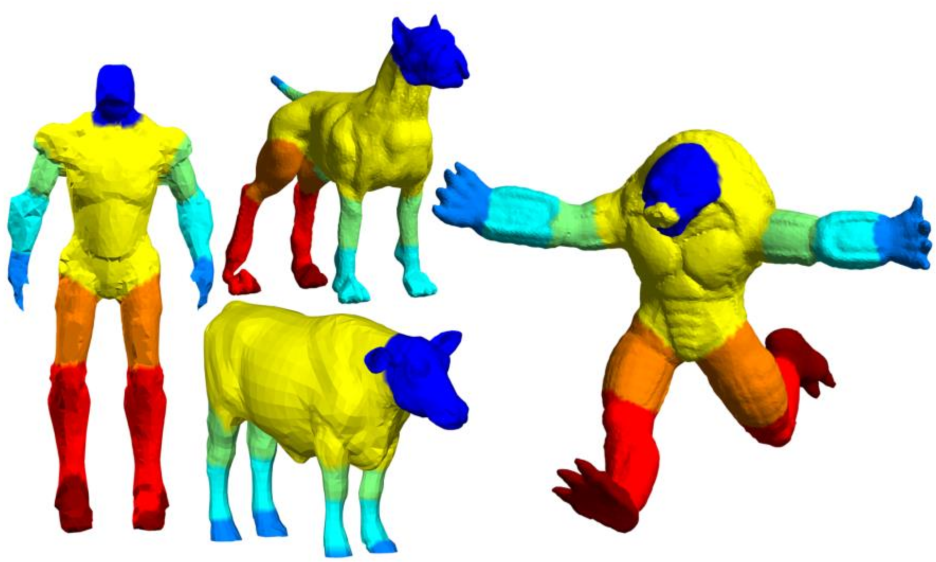
- Map 3D surface to 2D domain
  - One such mapping: flat torus (seamless => translation-invariant)
  - Many mappings exists: sample a few and average result
  - Which functions to map?
    XYZ, normals, curvature, …

SIGGRAPH ASIA 2018 TOKYO

[Maron et al. 2017]

55

Parameterization for Surface Analysis



SIGGRAPH ASIA 2018 TOKYO

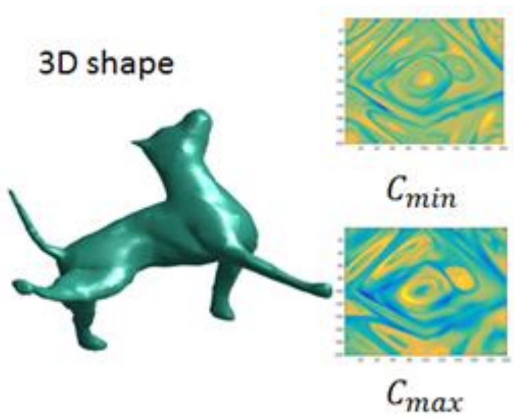
[Maron et al. 2017]

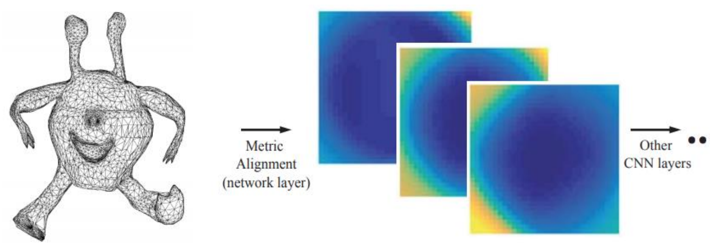
56

Other Parameterizations

Geometry Image

Metric Alignment





[Sinha et al. 2017]

[Ezuz et al. 2017]

SIGGRAPH ASIA 2018 TOKYO

57

Other Parameterizations

geodesic discs

parameterize in spectral domain

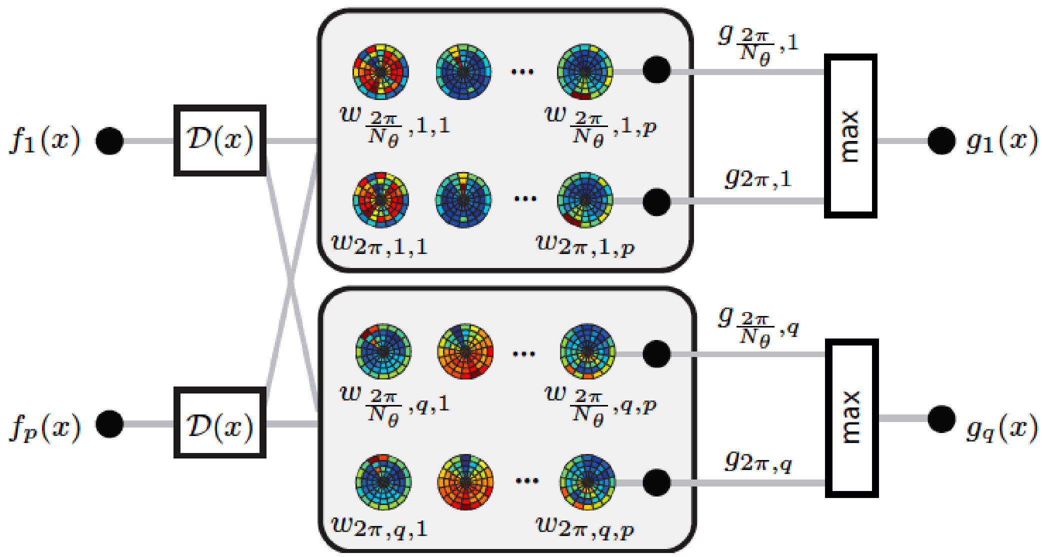
Spatial domain

Spectral domain

SIGGRAPH ASIA 2018 TOKYO

58

Other Parameterizations

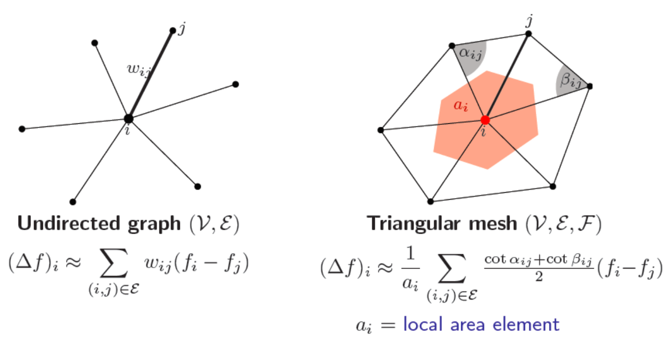


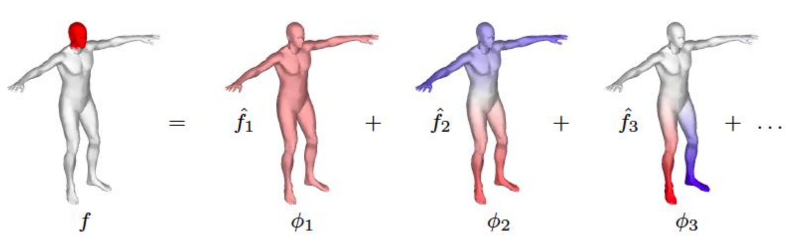
[Masci et al. 2015]

5
9

Discrete Laplacian

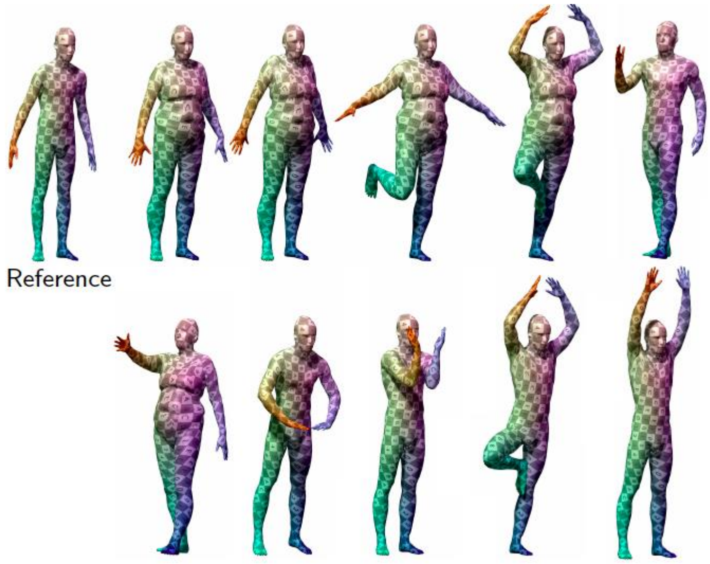
(slide credit: Michael Bronstein)





6
0

Transferring Correspondence



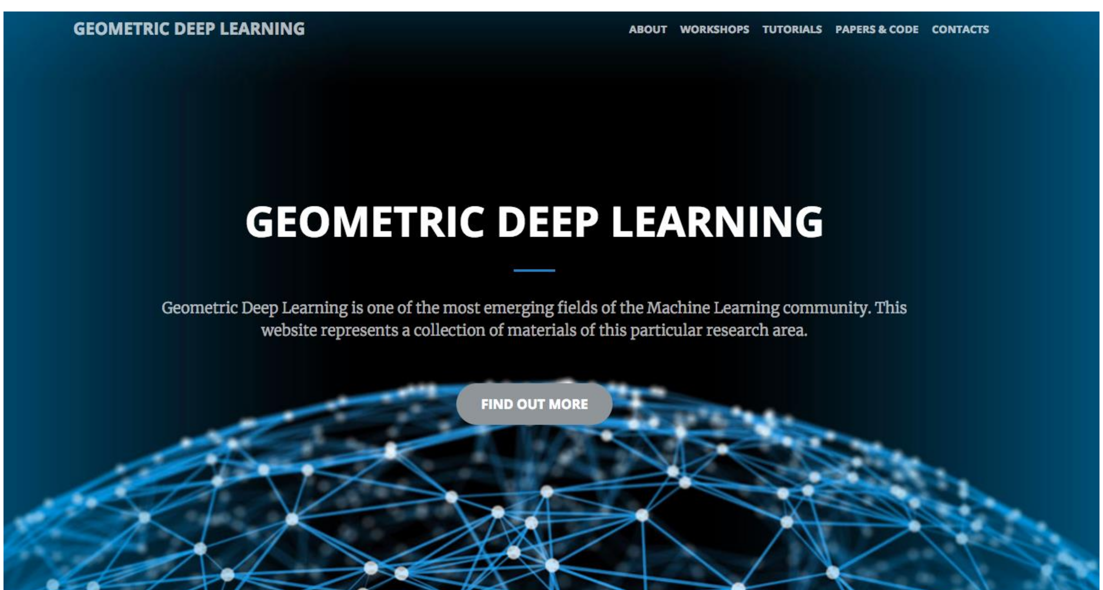
Texture transferred from reference to query shapes

[Monti et al. 2016]

61

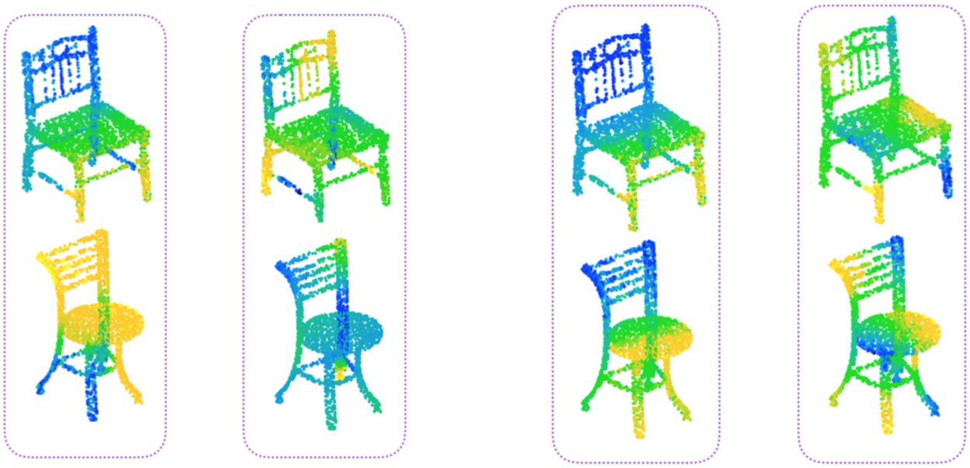
Spectral Methods

(slide credit: Michael Bronstein)



62

SyncSpecCNN

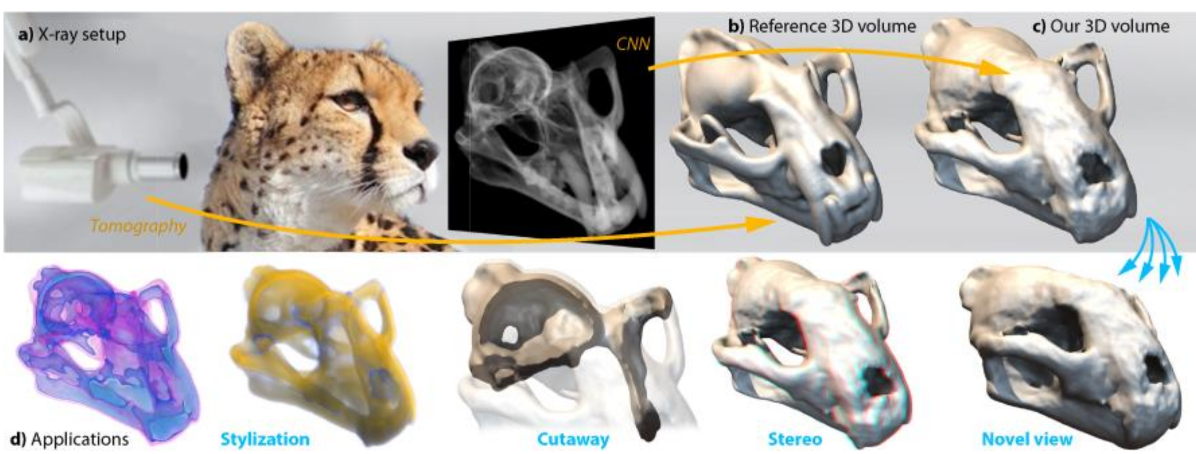


before synchronization   after synchronization

[Yi et al. 2017]

63

3D volumes form Xrays



Single-Image Tomography: 3D Volumes from 2D Cranial X-Rays. Henzler et al. **EG 2018**

64

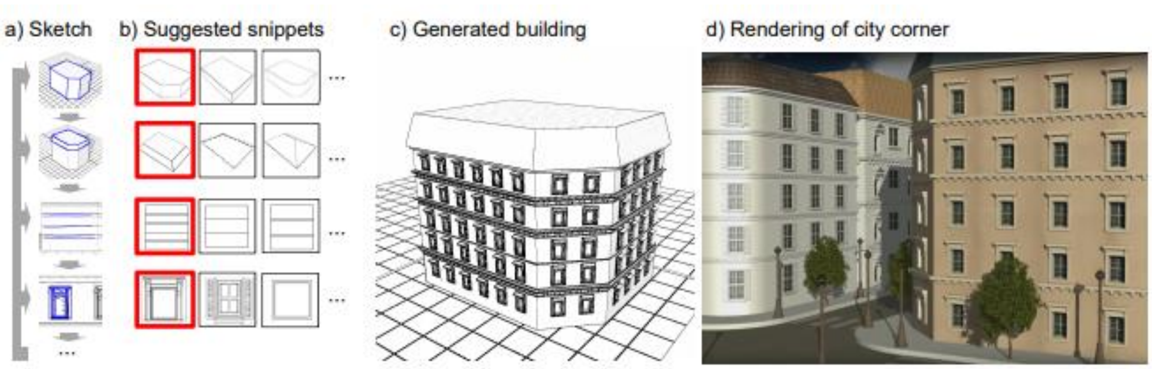
Representation for 3D

- Image-based
- Volumetric
- Point-based
- Surface-based

Parametric

6
5

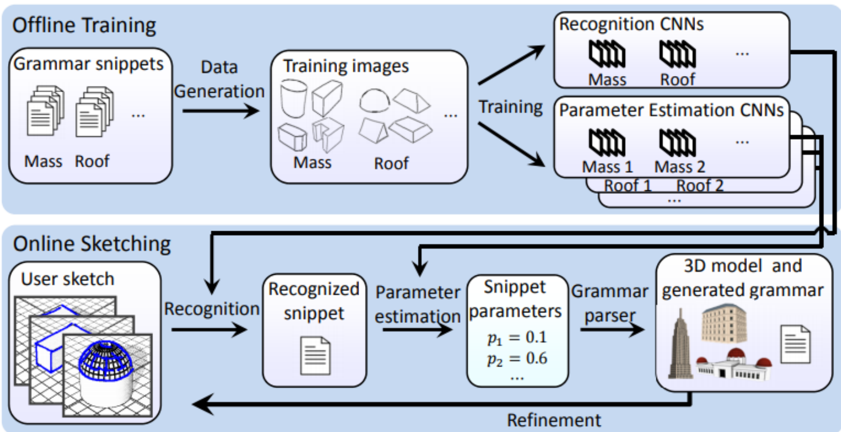
Procedural Parameter Estimation



Interactive Sketching of Urban Procedural Models, Nishida et al. 2016

66

Procedural Parameter Estimation: *Interactive Sketching of Urban Procedural Models*



Interactive Sketching of Urban Procedural Models, **Nishida et al.**

SIGGRAPH ASIA 2018 TOKYO

67

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH ASIA 2018 TOKYO

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics**

SIGGRAPH ASIA 2018 TOKYO

CreativeAI: Deep Learning for Graphics

Motion and Physics

Niloy Mitra UCL

Iasonas Kokkinos UCL/Facebook

Paul Guerrero UCL

Nils Thuerey TU Munich

Tobias Ritschel UCL

UCL — facebook Artificial Intelligence Research — TUM Technische Universität München

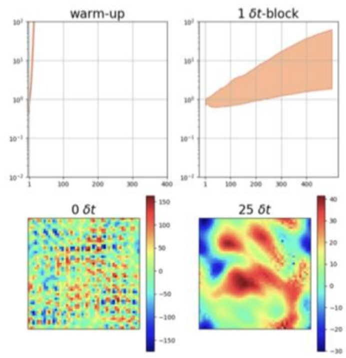
Timetable

		Niloy	Iasonas	Paul	Nils	Tobias
Theory and Basics	Introduction	X	X	X	X	X
	Theory	X			X	
	NN Basics	X	X			
	Alternatives to Direct Supervision			X		
	15 min. break					
State of the Art	Feature Visualization					X
	Image Domains		X			X
	3D Domains			X		X
	Motion and Physics	X			X	

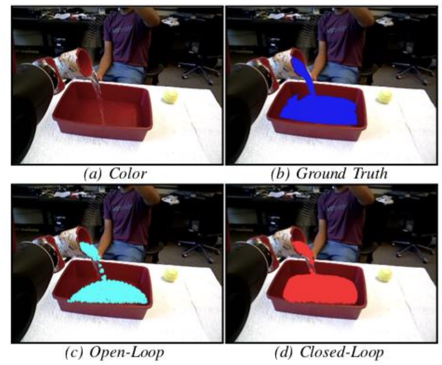
Deep Learning for Fluids

(slide credit: Nils Thuerey)

Tompson et. al 2017



Long et. al 2017

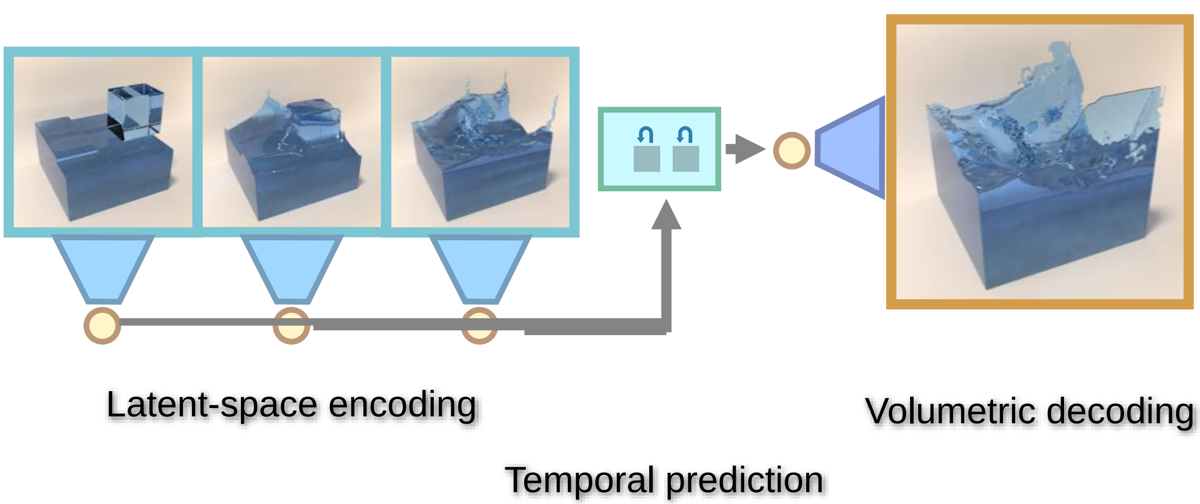


Schenck et. al 2017

SIGGRAPH ASIA 2018 TOKYO 3

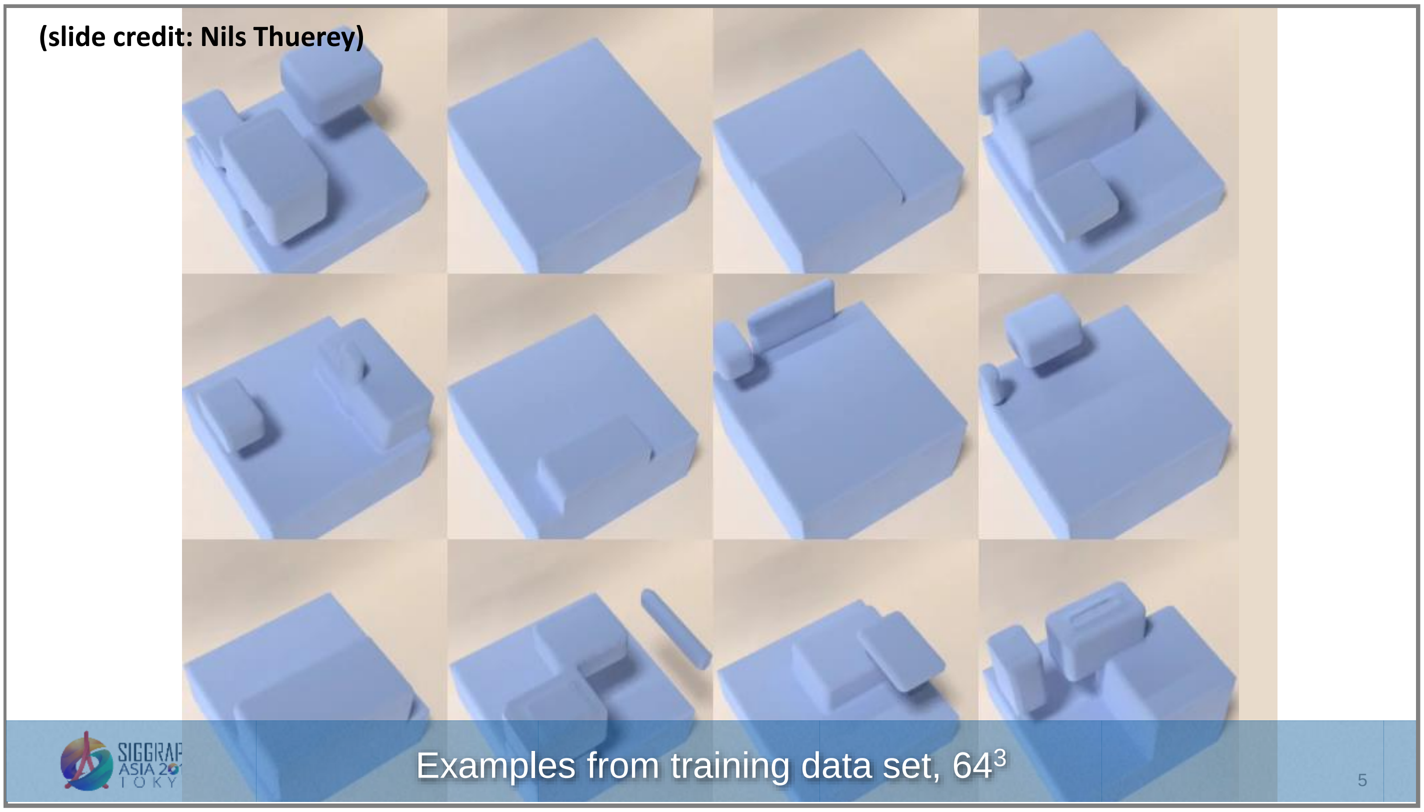
High Resolution Simulation of Liquids

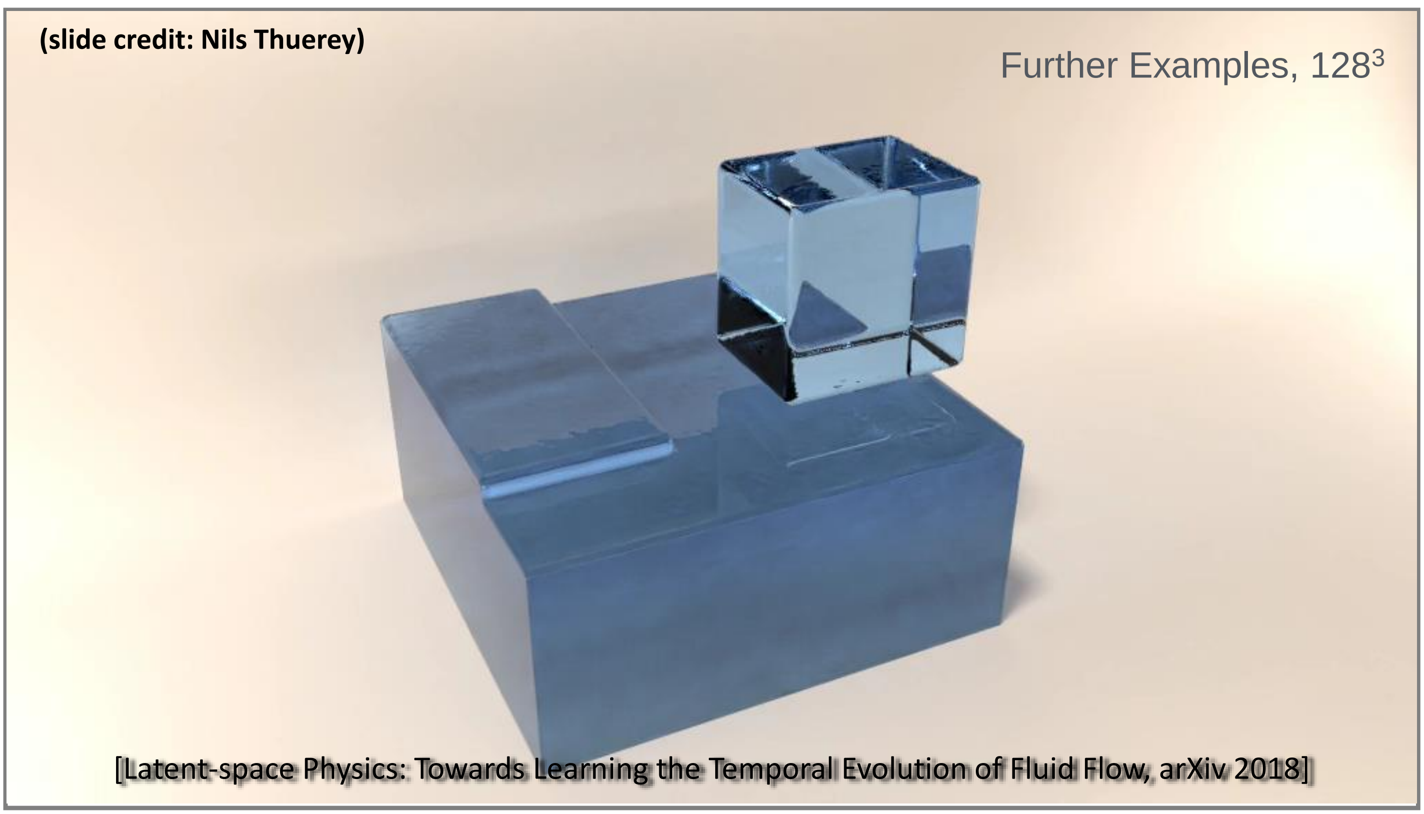
(slide credit: Nils Thuerey)



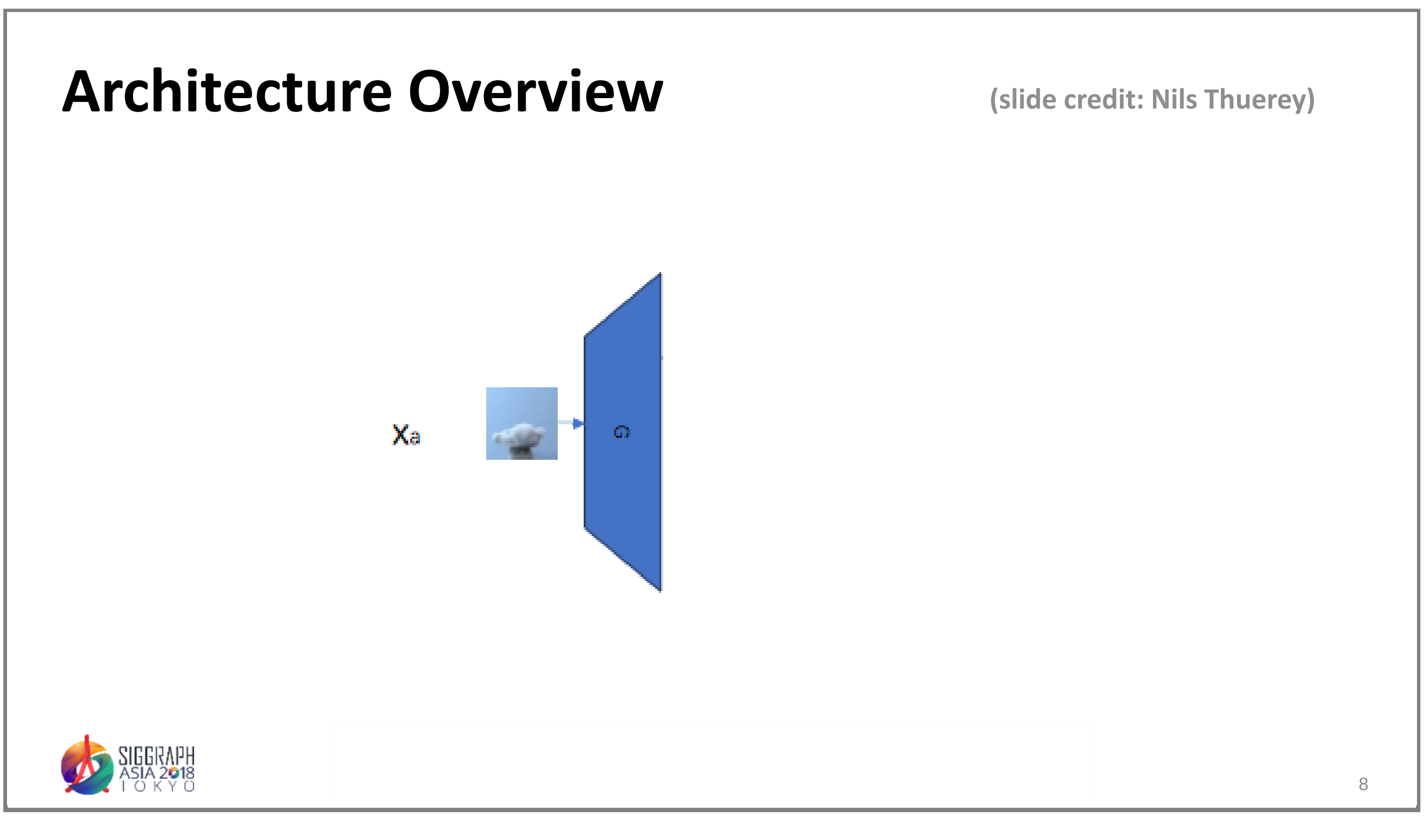
[Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow, arXiv 2018]

SIGGRAPH ASIA 2018 TOKYO 4


(slide credit: Nils Thuerey)
Examples from training data set, 64^3
5

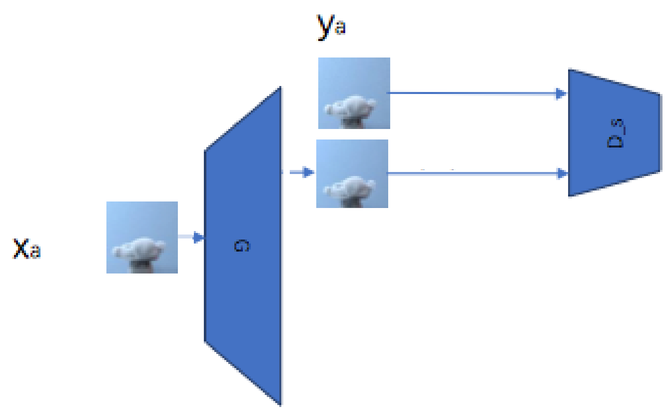

(slide credit: Nils Thuerey)
Further Examples, 128^3
[Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow, arXiv 2018]


Example input
(slide credit: Nils Thuerey)
Example target (4x)
Down-sample
[tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow , SIGGRAPH 2018]


Architecture Overview
(slide credit: Nils Thuerey)
Xa
G
SIGGRAPH ASIA 2018 TOKYO
8

Architecture Overview

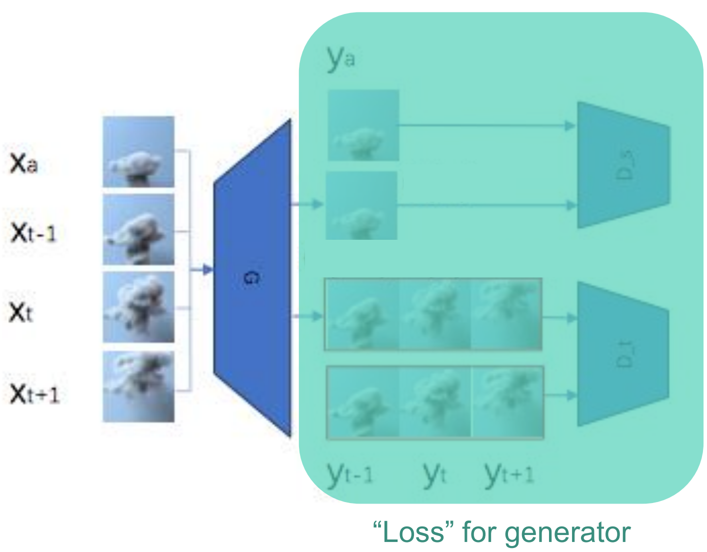
(slide credit: Nils Thuerey)



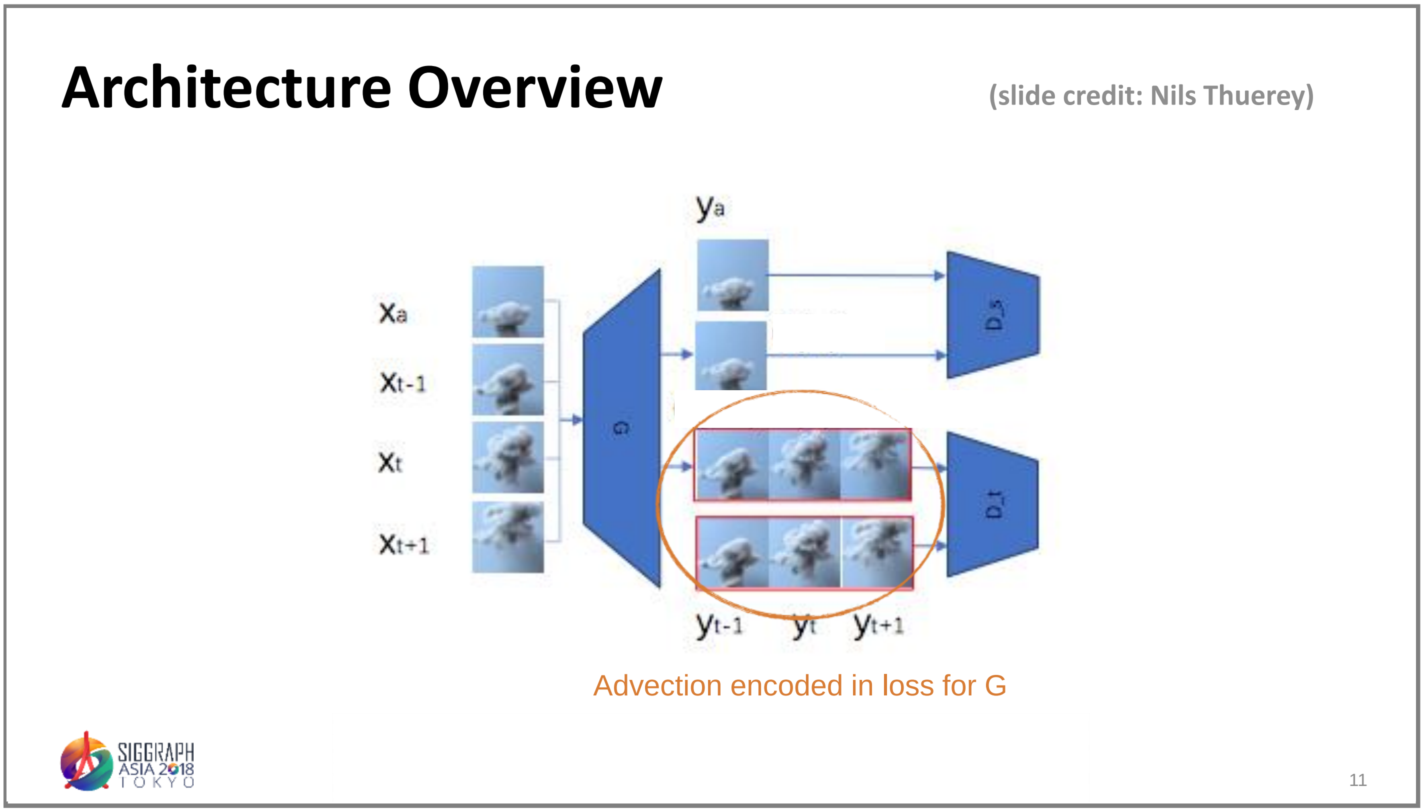
9

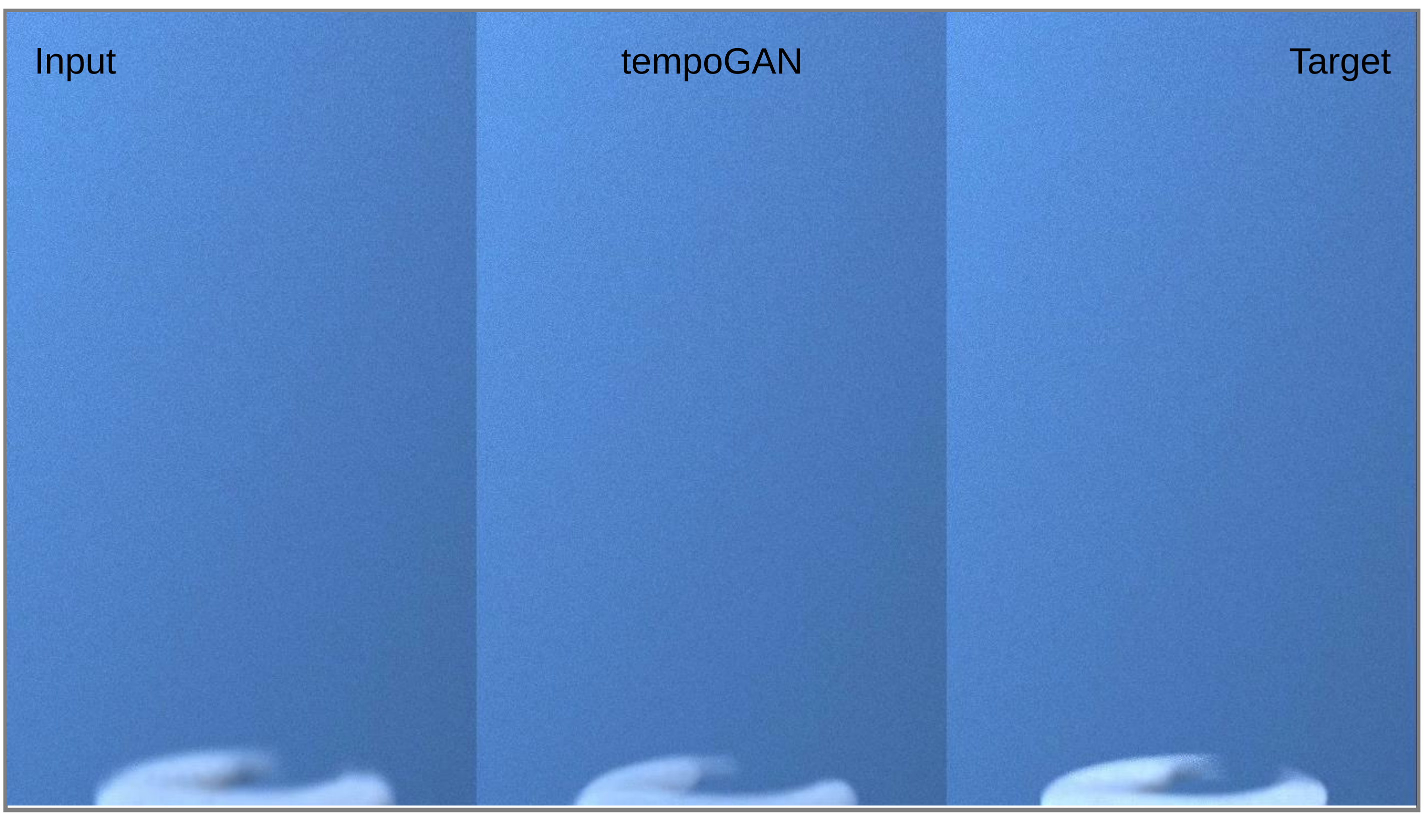
Architecture Overview

(slide credit: Nils Thuerey)

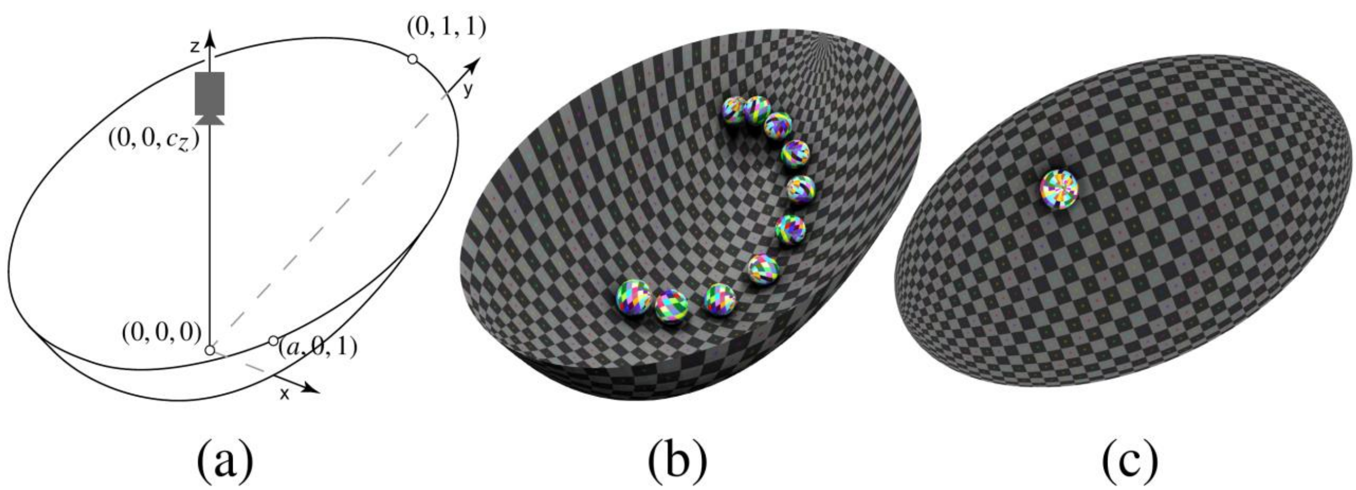


10


Architecture Overview
(slide credit: Nils Thuerey)
ya
Xa
Xt-1
Xt
Xt+1
G
D_s
D_t
yt-1
yt
yt+1
Advection encoded in loss for G
SIGGRAPH ASIA 2018 TOKYO
11

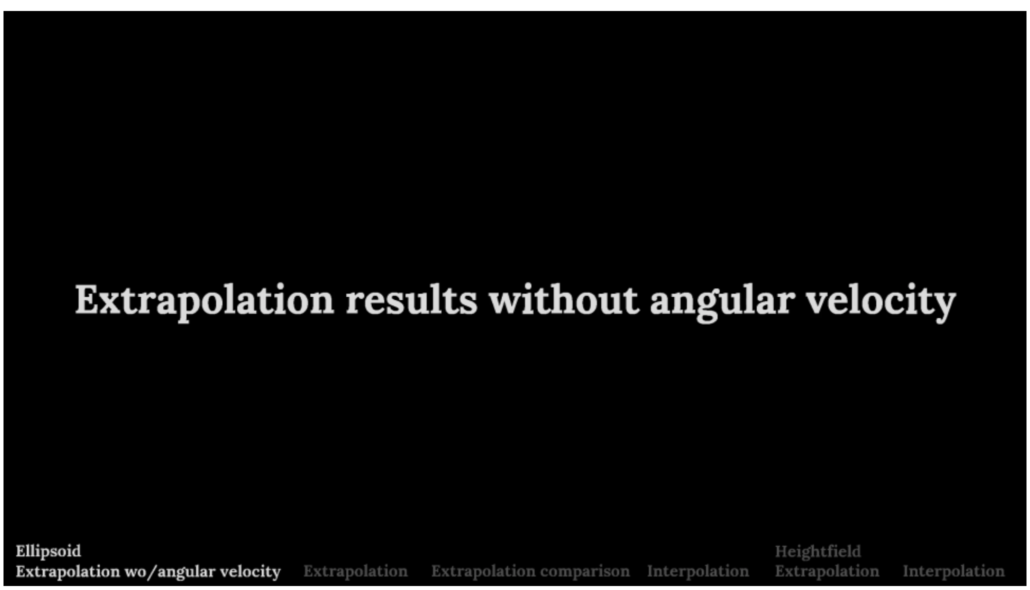

Input
tempoGAN
Target

Learning Rolling Motion



13

Learning Rolling Motion



14

Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, Daniel Zoran (DeepMind): **Visual Interaction Networks**, NIPS 2017

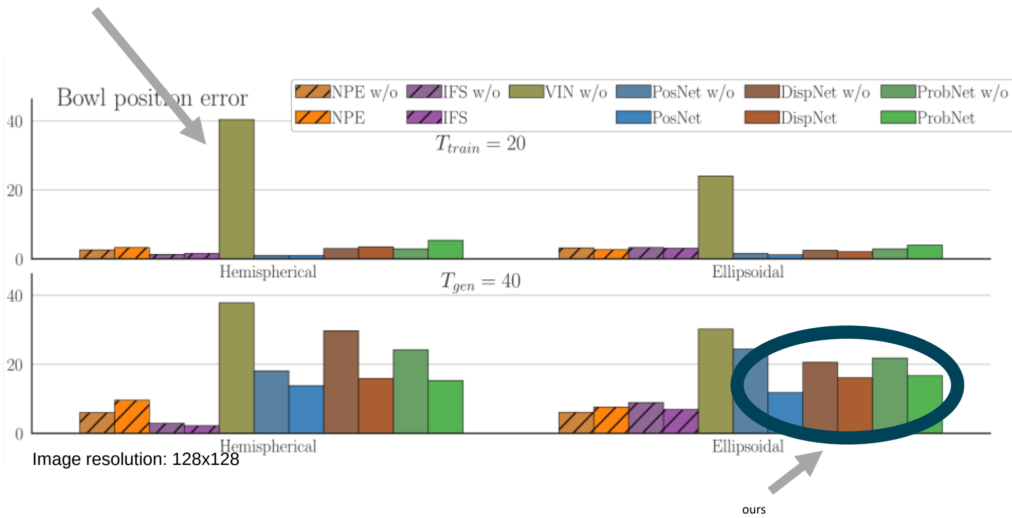


Image resolution: 128x128

15

Course Information (slides/code/comments)

http://geometry.cs.ucl.ac.uk/creativeai/

SIGGRAPH Asia Course **CreativeAI: Deep Learning for Graphics**