



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Deep Reinforcement Learning

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 13, 2020



Outline

Sequential Decision Making

Reinforcement Learning

Markov Decision Processes

Policy Iteration

Other Solution Methods

Deep Reinforcement Learning

Deep Q Learning

AlphaGo

AlphaGo Zero



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Sequential Decision Making



Sequential decision making: Multi-armed bandit problem



Action Formalize choosing a machine as **action a** at time t from a set A

Reward Action a_t has a **different¹** unknown pdf $p(r|a)$ generating reward r_t

Policy Formalize choosing an action a as pdf $\pi(a)$ which we call a **policy**

¹ This is not how gambling works

Evaluative Feedback



- Find action a producing the **maximum expected reward over time t** :

$$\max_a \mathbb{E} [p(r|a)]$$

- Difference to supervised learning: **No** feedback on **what** action to choose
- $\mathbb{E} [p(r|a)]$ is **not known in advance**
- We can form a one-hot encoded vector \mathbf{r} which reflects which action from \mathbf{a} caused the reward
- Estimate the joint pdf online as $\frac{1}{t} \sum_{i=1}^t \mathbf{r}_i := Q_t(\mathbf{a})$
- We call $Q_t(\mathbf{a})$ the action-value function, which changes with every new information

Incremental update of $Q_t(\mathbf{a})$

$$\begin{aligned}
 Q_{t+1}(\mathbf{a}) &= \frac{1}{t} \sum_{i=1}^t \mathbf{r}_i \\
 &= \frac{1}{t} \left(\mathbf{r}_t + \sum_{i=1}^{t-1} \mathbf{r}_i \right) \\
 &= \frac{1}{t} \left(\mathbf{r}_t + (t-1) \frac{1}{t-1} \sum_{i=1}^{t-1} \mathbf{r}_i \right) \\
 &= \frac{1}{t} (\mathbf{r}_t + (t-1)Q_t(\mathbf{a})) \\
 &= \frac{1}{t} (\mathbf{r}_t + t Q_t(\mathbf{a}) - Q_t(\mathbf{a})) \\
 &= Q_t(\mathbf{a}) + \frac{1}{t} (\mathbf{r}_t - Q_t(\mathbf{a}))
 \end{aligned}$$

Exploitation



Exploration



- Reward is maximized by a policy $\pi(a)$ choosing $\max_a Q_t(a)$
- We **exploit** a known good action
- This is a **deterministic**¹ policy called **greedy action selection**
- However we **need** to obtain **samples** r_a
 - This means we **cannot follow** the greedy action selection policy for learning
 - Sometimes **explore** by selecting other moves which could potentially be better

¹ If Q_t is equal for two a , the tie has to be broken e.g. randomly

We sample discrete actions a from $\pi(a)$, but what distributions can we use?

Uniform random

$$\pi(a) = \frac{1}{|A|}$$

- $|A|$ is the cardinality of the set of different actions A

Epsilon Greedy

$$\pi(a) = \begin{cases} 1 - \epsilon & \text{if } a = \max_a Q_t(a) \\ \epsilon / (n - 1) & \text{else} \end{cases}$$

Softmax

$$\pi(a) = \frac{e^{Q_t(a)/\tau_t}}{\sum_{n=1}^{|A|} e^{Q_t(a_n)/\tau_t}}$$

- τ_t is called **temperature** and used to decrease exploration over time

Summary

So far we ...

- considered **sequential decision making** in a setting known as multi-armed bandits
- found out that **estimating a function** $Q(a)$ and the **greedy action selection** policy $\pi(a) = \max_a Q(a)$ maximized our reward
- learned that **exploration** of different actions is necessary
- assumed rewards **didn't depend** on a **state** of the world
- and our action at time t **doesn't influence** the **rewards** from a at $t + 1$

NEXT TIME
ON DEEP LEARNING



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Deep Reinforcement Learning - Part 2

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 13, 2020





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Reinforcement Learning



Associativity

We extend the multi-armed bandits problem:

- We introduce a state of the world at any time t : s_t
- Rewards now additionally **depend on** the **state** s_t :

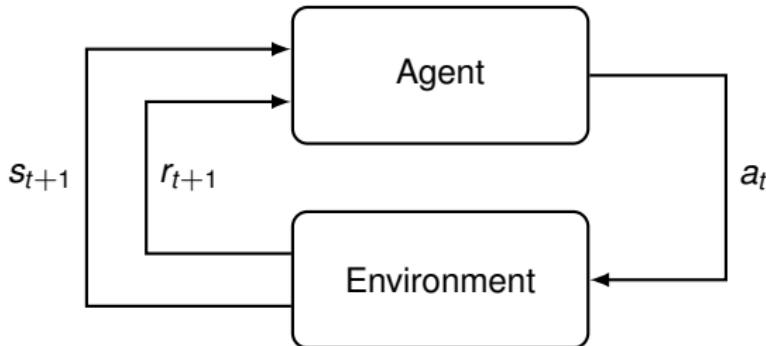
$$p(r_t | s_t, a_t)$$

- However this setting is known as **contextual bandit**
- In the full **reinforcement learning problem**, actions influence the state:

$$p(s_{t+1} | s_t, a_t)$$

Markov Decision Processes

Markov Decision Process



Action An **action** a_t at time t from a set A

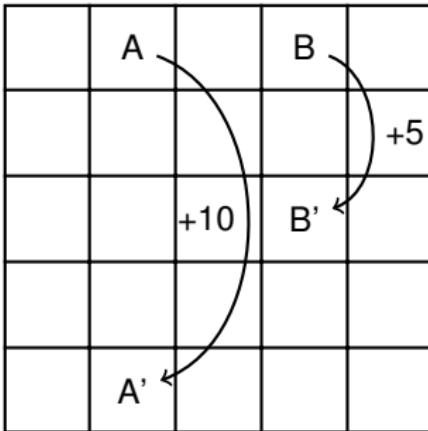
State A **state** s_t from a set S

A **state transition pdf** $p(s_{t+1}|s_t, a_t)$

Reward Transition produces reward $r_{t+1} \in R \subset \mathbb{R}$ according to $p(r_{t+1}|s_t, a_t)$

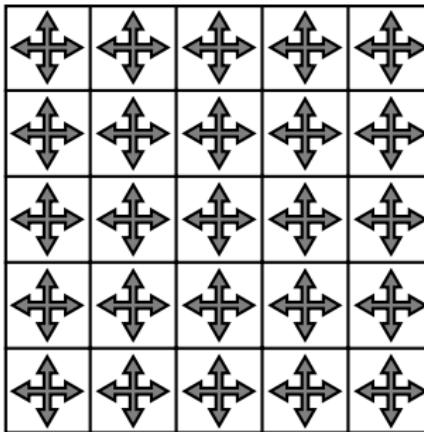
Policy Agents choose actions a_t by a policy $\pi(a|s)$

If all those sets are **finite** we call this a **finite MDP**



- Here s is the field we are currently on.
- The agent can **move** in all four directions
- Any action which would leave the grid has $p(s_{t+1}|a_t, s_t)$ equal to a δ distribution on $s_{t+1} = s_t$ and a similarly deterministic $r_t = -1$
- Every state we reach other than tile A' and B' deterministically causes $r_t = 0$
- On A or B **any** action will take us to A' or B' respectively

Example policy



- Policies now depend on s_t
- We can extend the **uniform random policy** to be independent from s_t
- However there's no reason to believe that this policy is any good
- How can we **estimate good policies?**

What is a good policy?

- We have to be precise about good
- Preliminary we have to state **two kinds** of tasks
 1. **Episodic** tasks which have an **end**
 2. **Continuing** tasks which are **infinitely** long
- **Unify them** using a **terminal state** in **episodic tasks** which only transition to themselves with deterministic $r_t = 0$
- Goal is to **maximize the future return**

$$\max_{\pi(s_t, a_t)} g_t = \sum_{k=t+1}^T \gamma^{k-t-1} r_k$$

- γ is a **discount** reducing influence of rewards **far** in the future
- $\gamma \in (0, 1]$ meaning that $\gamma = 1$ is allowed as long as $T \neq \infty$

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Deep Reinforcement Learning - Part 3

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 13, 2020



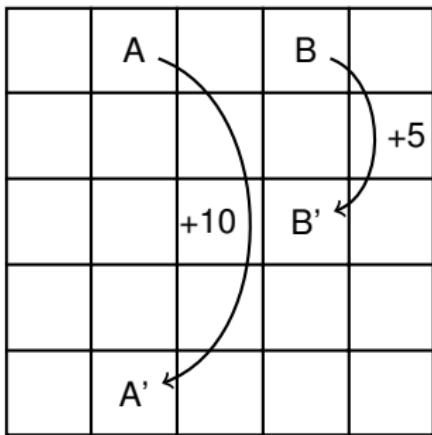
Policy Iteration



- Before we used the action-value function $Q(a)$
- Now a_t has to depend on s_t
- Use an oracle predicting the future reward g_t following $\pi(s_t, a_t)$ from s_t
- We introduce the **state-value function** $V_\pi(s)$

$$V_\pi(s) = \mathbb{E}_\pi [g_t | s_t] = \mathbb{E}_\pi \left[\sum_{k=t+1}^T \gamma^{k-t-1} r_k | s_t \right]$$

State-value Function Example



The definition of the gridworld

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

$V_\pi(s)$ for the uniform random policy

- Recall our grid example
- Some **edge tiles** are negative since the policy **can't control the move**
- What if we use the **greedy action selection** policy on this $V_\pi(s)$?
- We get a **better policy!**

Action-value function

- Before we used the action-value function $Q(a)$
- Now we introduced $V_\pi(s)$ filling a similar role
- We can also introduce the **action-value function** $Q_\pi(s, a)$
- Basically this accounts for the **transition probabilities**

$$Q_\pi(s, a) = \mathbb{E}_\pi [g_t | s_t, a_t] = \mathbb{E}_\pi \left[\sum_{k=t+1}^T \gamma^{k-t-1} r_k | s_t, a_t \right]$$

Are Value Functions Created Equal?

- No.
- There can only be one¹ **optimal** $V^*(s)$
- We can state its existence without referring to a specific policy:

$$V^*(s) = \max_{\pi} V_{\pi}(s) \quad (1)$$

- $Q^*(s, a)$ can also be defined and is related to $V^*(s_t)$ by:

$$Q^*(s, a) = \mathbb{E} [r_{t+1} + \gamma V^*(s_{t+1})] \quad (2)$$

¹ in a finite MDP

Optimal Value-function Example

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

$V_\pi(s)$ for the uniform random policy

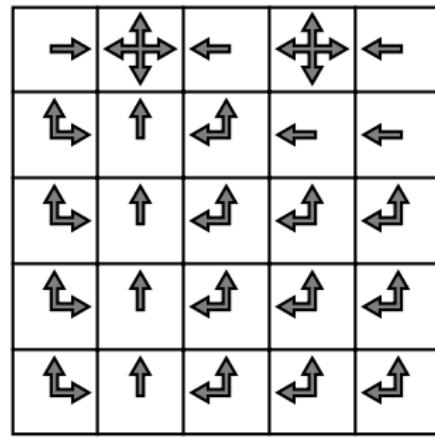
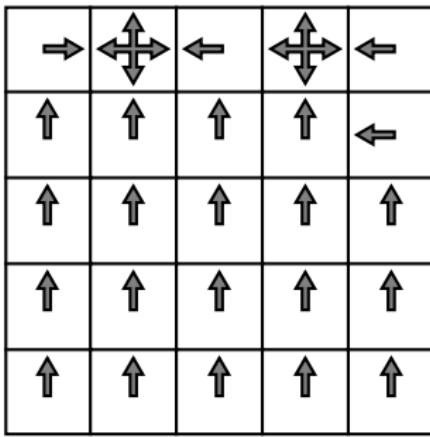
V^*

- Observe that V^* is strictly positive since it's deterministic

Optimal Policies

- Policies can now be ordered: $\pi \geq \pi'$ if and only if $V_\pi(s) \geq V_{\pi'}(s), \forall s \in S$
- **Any** policy π with $V_\pi = V^*$ is an optimal policy π^*
- This implies there might be **more than one** optimal policy
- Given either V^* or Q^* an optimal policy is directly obtained by **greedy action selection**

Greedy Action Selection on $V^*(s)$ or $Q^*(s, a)$



$\pi'(s, a) = \text{Greedy Action Selection on } V_\pi(s)$ with
 $\pi(s, a)$ being uniform random

$\pi^*(s, a) = \text{Greedy Action Selection on } V^*(s)$

A Tool to Compute Optimal Value-functions

- We **still need** to compute $V^*(s)$ and $Q^*(s, a)$
- For this the **Bellman equations** can be utilized
- They are **consistency conditions** for the value functions

Bellman equation for $V_\pi(s)$

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s_{t+1}, r} p(s_{t+1}, r|s, a) [r + \gamma V_\pi(s_{t+1})]$$

Policy Evaluation

- The Bellman equations form a **system of linear equations** which can be solved for **small** problems
- Better: **Iteratively solve**, by turning the Bellman equations into **update rules**:

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s_{t+1}, r} p(s_{t+1}, r|s, a) [r + \gamma V_k(s_{t+1})]$$

For all $s \in S$

Policy Improvement

- $V_\pi(s)$ is used to **guide our search** for good policies
- Another necessary step is to **update the policy**
- However if we use **greedy action selection** an update of $V_\pi(s)$ is **simultaneously** an update of $\pi(s)$
- Now iterate **evaluation** of the **greedy policy** on $V_\pi(s)$
- Stop iterating if the **policy stops changing**
- But is this **guaranteed** to work?

Policy Improvement Theorem

- We consider changing a single action a_t in state s_t but following π
- In general if

$$Q_\pi(s, \pi'(s)) \geq V_\pi(s), \forall s \in S \implies \pi' \geq \pi$$

- This also implies:

$$V_{\pi'}(s) \geq V_\pi(s)$$

- Because we **only select greedy** we have $Q_\pi(s, a) > V_\pi(s)$ before convergence
- So iteratively updating $V_\pi(s)$ and using **greedy action selection** is guaranteed to work here
- We terminate if the policy no longer changes
- Last remark: If we don't loop over all $s \in S$ for policy evaluation, but update the policy directly this algorithm is called **Value iteration**

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Deep Reinforcement Learning - Part 4

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 13, 2020



Other Solution Methods

Limitations

- Both policy iteration and value iteration **require** using the **updated policies** during learning to obtain better approximations to $V^*(s)$
- For this reason we call them **on-policy** algorithms
- Additionally we assumed the **state-transition** pdf and **reward** pdf are known
- Can we relax this?
- Yes. The methods differ mostly **how they perform policy evaluation**

Monte Carlo Techniques

Properties

- Only for **episodic** tasks
- Off-policy - learns $V^*(s)$ by following any **arbitrary** $\pi(s, a)$
- Does **not need** information about dynamics of the environment

Scheme

- **Generate** an **episode** by using some policy
- Loop **backwards** over the episode accumulating the expected future reward
$$g_t = g_{t+1} + r_{t+1}$$
- If a state was **not yet** visited append g_t to a list $returns(s_t)$
- Update $V_{s_t} = \frac{1}{N} \sum_{n=1}^N returns_n(s_t)$

Temporal Difference Learning

Properties

- On-policy
- Does **not need** information about dynamics of the environment

Scheme

- Loop and follow $\pi(s_t, a_t)$
- Use a from $\pi(s_t, a_t)$, observe r_t, s_{t+1}
- Update: $V_{t+1}(s) = V_t(s) + \alpha [r_t + \gamma V_t(s_{t+1}) - V_t(s_t)]$
- **Converges to the optimal solution**
- A variant of this estimates $Q_{(s,a)}$ and is known as SARSA

Q Learning

Properties

- Off-policy
- Temporal difference type of method
- Does **not need** information about dynamics of the environment

Scheme

- Loop and follow $\pi(s_t, a_t)$ derived from $Q_t(s, a)$ e.g. ϵ -greedy
- Use a from $\pi(s_t, a_t)$, observe r_t, s_{t+1}
- Update: $Q_{t+1}(s, a) = Q_t(s, a_t) + \alpha \left[r_t + \gamma \max_a Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t) \right]$

If you have Universal Function Approximators

- What about just parametrizing $\pi(s_t, a_t, \mathbf{w})$ by weights \mathbf{w} and use some loss-function L ?
 - Known as **policy gradient** and this instance is called REINFORCE
- Generate an episode using $\pi(s_t, a_t, \mathbf{w})$
- Go forwards in the episode: $t = 0, \dots, T - 1$
- $\mathbf{w} = \mathbf{w} + \eta \gamma^t g_t \nabla_{\mathbf{w}} \ln (\pi(a_t | s_t, \mathbf{w}))$

NEXT TIME
ON DEEP LEARNING



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Deep Reinforcement Learning - Part 5

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 13, 2020





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Deep Reinforcement Learning



Deep Q Learning

Atari Games: Human-level control through deep reinforcement learning [4]

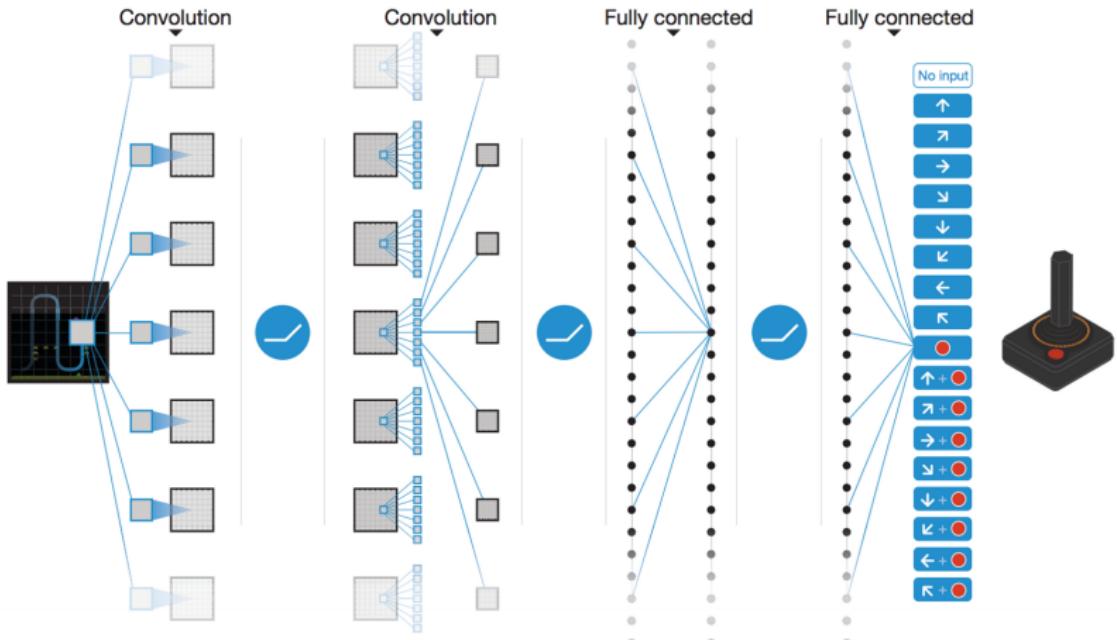
- Volodymyr Mnih et al. (Google DeepMind)
2013/2015
- Idea: Let a neural network play Atari games!
- Input: Current and three subsequent video frames from game
- Processed by network trained with reinforcement learning
- Goal: learn best controller movements
- Convolutional layers for frame processing, fully-connected for final decision making



Atari Pac-Man

Source: Human-level control through deep reinforcement learning [4]

Learning Atari Games



Source: Human-level control through deep reinforcement learning [4]

Learning Atari Games

- **Deep Q-network:** Deep network that applies Q-learning
- State s_t of the game: current + 3 previous frames (image stack)
- 18 outputs associated with an action
 - Each output estimates optimal action value for “its” action given the input
- Instead of label & cost function, update to maximize reward
- Reward: +1/-1 when game score increased/decreased, 0 otherwise
- ϵ -greedy policy with ϵ decreasing to a low value during training
- Semi-gradient form of Q-learning to update network weights w
- Uses mini-batches to accumulate weight updates

Target Network

- Weight update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[r_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a, \mathbf{w}_t) - \hat{q}(s_t, a_t, \mathbf{w}_t) \right] \cdot \nabla_{\mathbf{w}_t} \hat{q}(s_t, a_t, \mathbf{w}_t)$$

- Problem: The target $\gamma \max_a \hat{q}(s_{t+1}, a, \mathbf{w}_t)$ is a function of \mathbf{w}_t .
 - Target changes simultaneously with the weights we want to learn!
 - Training can oscillate or diverge
- Idea: Use a second **target network**:
- After each C steps, copy weights of action-value network to a duplicate network and keep them fixed
- Use output \bar{q} of “target network” as a target to stabilize:

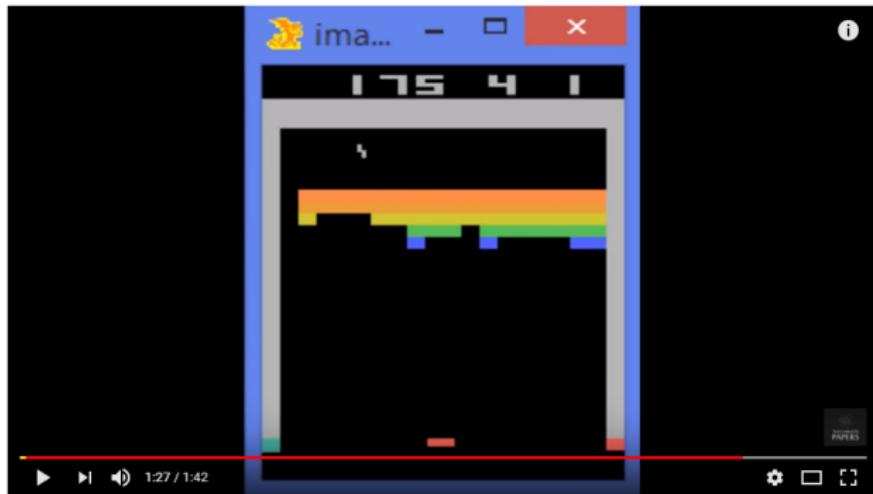
$$\gamma \max_a \bar{q}(s_{t+1}, a, \mathbf{w}_t)$$

Experience Replay

Goal: Reduce correlation between updates

- After performing action a_t for image stack s_t (state) and receiving reward r_t , add (s_t, a_t, r_t, s_{t+1}) to **replay memory**
 - Memory accumulates experiences
- To update the network, draw random samples from memory, instead of taking the most recent ones
 - Removes dependence on current weights
 - Increases stability

Atari Breakout Example



Video on learning Atari Breakout. [Click here](#)

AlphaGo

Mastering the game of Go with deep neural networks and tree search [1]

- Go is an ancient Chinese boardgame: Black plays against white for control over the board
- Simple rules but extremely high number of possible moves and situations
- Performance on par with professional human players thought years away



Traditional Go board

Source: <https://commons.wikimedia.org/wiki/File:FloorGoban.jpg>

Challenges in Go

- Go is a “perfect information” game: No hidden information and no chance
- Theoretically, we can construct a full game tree and traverse it with Minimax to find the best moves
- Problem: High number of legal moves (≈ 250 – chess ≈ 35)
- Games involve many moves (≈ 150)
- Exhaustive search is infeasible!

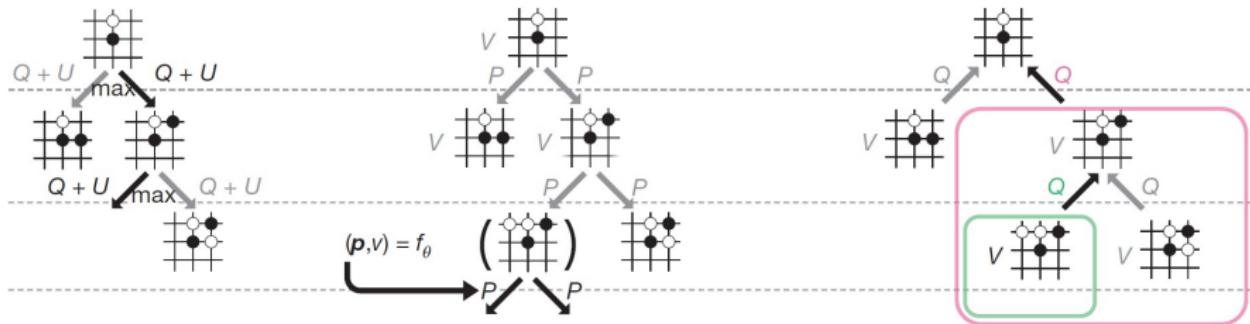
Challenges in Go (cont.)

- Search tree can be **pruned** if we have an accurate evaluation function
- For chess (DeepBlue) already extremely complex and based on massive human input
- For Go: “No simple yet reasonable evaluation function will ever be found for Go.” (Müller 2002) [5]
- Still: **AlphaGo beat Lee Sedol and Ke Jie, two of the world’s strongest players in 2016 and 2017!**

Mastering the game of Go with deep neural networks and tree search [1]

- AlphaGo was developed by Silver et al. (also Google DeepMind)
- Combination of multiple methods:
 - Deep neural networks
 - Monte Carlo tree search (MCTS)
 - Supervised learning **and**
 - Reinforcement learning
- First improvement compared to a full tree search: Monte Carlo Tree Search (MCTS)
- Networks to support efficient search through tree

Monte Carlo Tree Search



- Idea: Run many Monte Carlo simulations of episodes (=entire Go games) to select action (=where to place a stone)
- Starting from a root node representing the current state, MCTS iteratively extends the search tree

Source: Mastering the game of go without human knowledge [2]

Monte Carlo Tree Search (cont.)

Algorithm:

- **Selection:** Starting at root, traverse with tree policy to a leaf node
- **Expansion:** (Optional) add one or more child nodes to the current leaf
- **Simulation:** From the current or the child node, simulate episode with actions according to rollout policy
- **Backup:** Propagate the received reward back through the tree
- Repeat for a certain amount of time, then stop
- Then, choose action from root node according to accumulated statistics
- Start again with new root node

Monte Carlo Tree Search (cont.)

- Tree policy guides in how far successful paths are frequented more often.
- Typical exploration/exploitation trade-off.
- Problem: Estimation via MCTS not accurate enough for Go.
- Ideas in AlphaGo:
 - Control tree expansion by using a neural network to find promising actions.
 - Improve value estimation by a neural network.
- More efficient extension & evaluation of search tree → better at Go!

Deep Neural Networks for Go

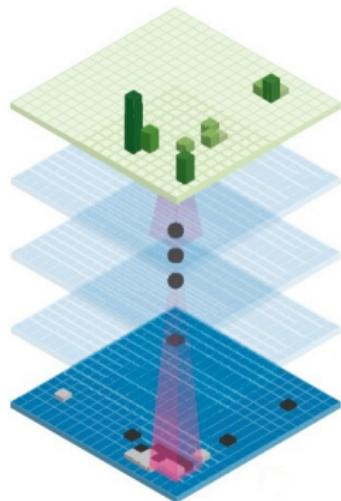
Utilization of three different networks:

- **Policy network:** Suggests the next move in leaf nodes for extension
- **Value network:** Given the current board position, get chances of winning
- **Rollout policy network:** Guide rollout action selection
- All networks are deep convolutional networks
- Input: Current board position and additional precomputed features

Source: Mastering the game of Go with deep neural networks and tree search [1]

Policy Network

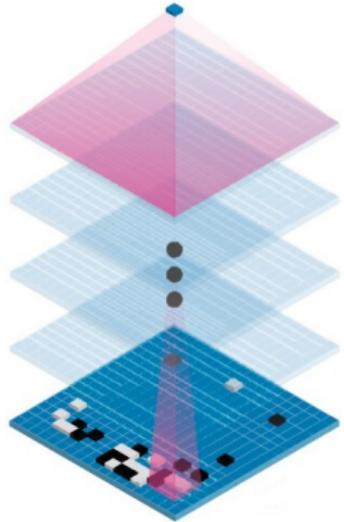
- 13 conv-layers, one output for each point on the Go board.
- Huge database of expert human moves (30 mio) available.
- Start with **supervised** learning: Train network to predict the next move in **human expert plays**
- Further train network with reinforcement learning by playing against **older versions of itself**. Reward when winning the game
- Older versions avoid correlation and instability
- Training time: 3 weeks on 50 GPUs + 1 day for RL



Source: Mastering the game of Go with deep neural networks and tree search [1]

Value network

- Same architecture as policy network but just one output node
- Goal: Estimate how likely the current state leads to a win
- Training utilized self-play games of reinforcement learned policy
- Trained using Monte-Carlo policy evaluation for 30 mio positions from these games
- Training time: 1 week on 50 GPUs



Source: Mastering the game of Go with deep neural networks and tree search [1]

Rollout policy network

- AlphaGo could use policy network to select moves during roll-out
- Problem: Inference comparatively high: 5 ms
- Solution: Train simpler, linear network on subset of data that provides actions **fast**
- Speedup of ≈ 1000 compared to policy network → more simulations possible

AlphaGo Zero

AlphaGo Zero: Do we even need humans for training?

- After minor improvements, Silver et al. proposed AlphaGo Zero:
- **Solely** trained with reinforcement learning & playing against itself!
- Simpler MCTS, no rollout policy
- Include MCTS in self-play games
- Multi-task training: Policy and value network share initial layers
- Further extension in Dec. '17: AlphaZero [3] – able to also play chess and shogi

**NEXT TIME
ON DEEP LEARNING**

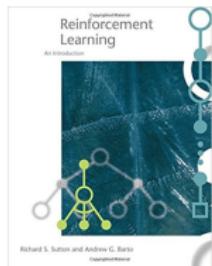
Next Time

- Algorithms to learn if we **don't even observe rewards**
- How to benefit from **adversaries**
- Extensions to perform **image processing** tasks

Comprehensive Questions

- What is a policy?
- What are value functions?
- Explain the exploitation vs exploration dilemma.
- Describe typical solutions to the dilemma.
- What is the difference of a multi armed bandit problem to the full reinforcement learning problem?
- Describe a Markov decision process.
- Is an optimal policy necessarily unique?
- What do the Bellman equations represent?
- Describe policy iteration.
- Why does policy iteration work?
- How can you beat your friends in every Atari game?
- How can one master the game of Go?

Further Reading



Reinforcement Learning

Richard Sutton

- [Link](#) - the one real reference for Reinforcement learning in its 2018 draft, including Deep Q learning and Alpha Go details



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

References



References I

- [1] David Silver, Aja Huang, Chris J Maddison, et al. "Mastering the game of Go with deep neural networks and tree search". In: [Nature](#) 529.7587 (2016), pp. 484–489.
- [2] David Silver, Julian Schrittwieser, Karen Simonyan, et al. "Mastering the game of go without human knowledge". In: [Nature](#) 550.7676 (2017), p. 354.
- [3] David Silver, Thomas Hubert, Julian Schrittwieser, et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: [arXiv preprint arXiv:1712.01815](#) (2017).
- [4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. "Human-level control through deep reinforcement learning". In: [Nature](#) 518.7540 (2015), pp. 529–533.
- [5] Martin Müller. "Computer Go". In: [Artificial Intelligence](#) 134.1 (2002), pp. 145–179.

References II

- [6] Richard S. Sutton and Andrew G. Barto.
Introduction to Reinforcement Learning. 1st. Cambridge, MA, USA: MIT Press, 1998.



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Unsupervised Deep Learning

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Motivation



Motivation

Dataset variation

	So far	Medical imaging
Dataset size	Huge (up to millions!)	Small (30-100 patients)
Variation	Many objects	One complex object
Modalities	Few modalities	Many modalities

Example

- Germany: 65 CT scans per 1000 inhabitants
- 5 million CT scans in 2014!
- ... but it is highly **sensitive** data
- Trend to make this data available
- ... but annotation is **expensive** to obtain

Source: <http://www.cancerimagingarchive.net>, <https://data.oecd.org/healthcare/computed-tomography-ct-exams.htm>

Solutions

- **Weakly supervised** learning: label for related task

Example: object localization with class labels



Brushing teeth



Cutting trees

- **Semi-supervised** learning: partial (typical: little) labeled data
- **Unsupervised** learning: no labeled data

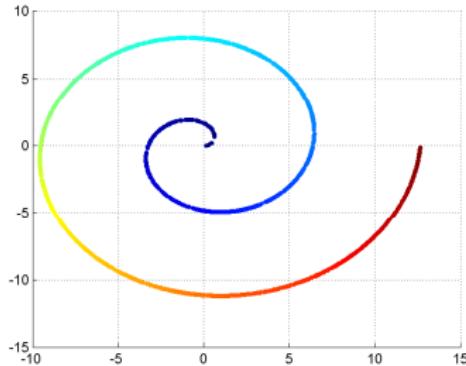
Source: [21]

Label-free Learning

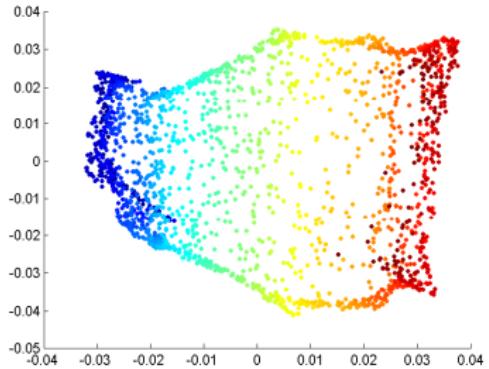
Applications as

- Dimensionality Reduction

Data on a manifold



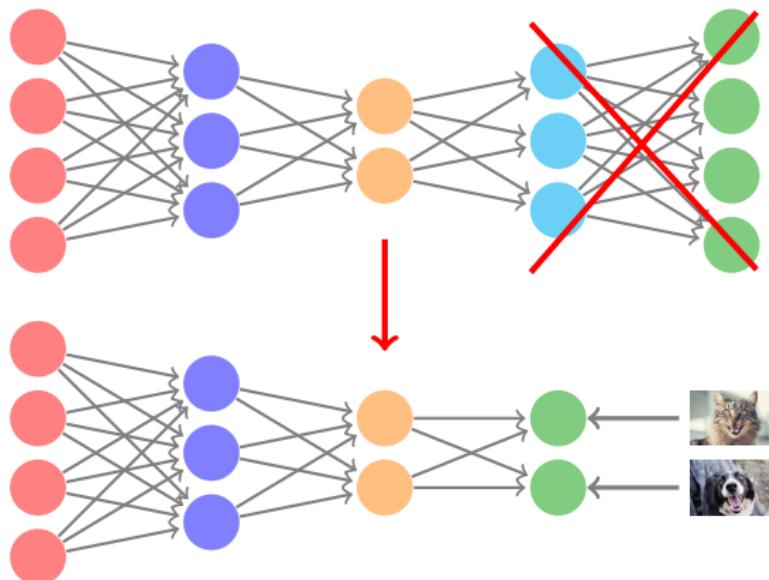
Dimensionality reduced



Label-free Learning

Applications as

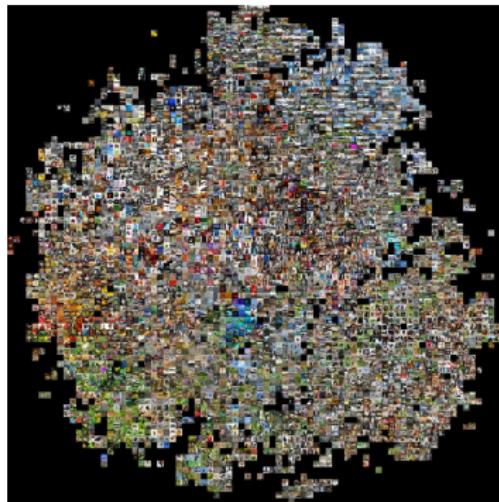
- Network Initialization (related: transfer learning)



Label-free Learning

Applications as

- Representation Learning
- e.g. for clustering

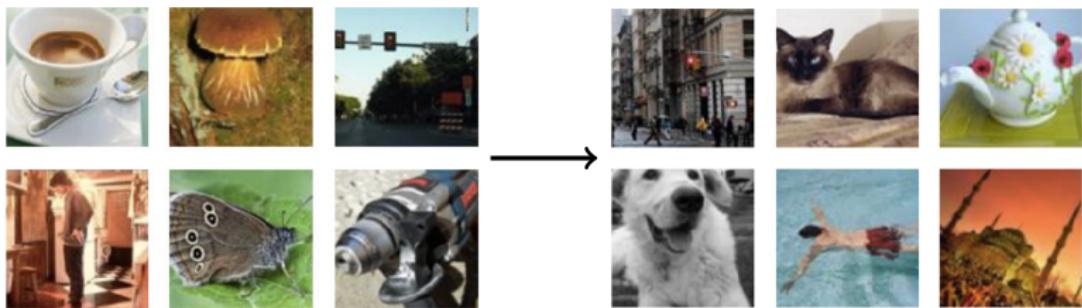


Source: <http://cs.stanford.edu/people/karpathy/cnnembed/>

Label-free Learning

Applications as

- Generative Models
 - Realistic generation tasks
 - Missing Data → semi-supervised learning
 - Image to image translation
 - Simulate possible futures → reinforcement learning



Training examples

Model samples

Source: [7]

Label-free Learning

Today

- Restricted Boltzmann Machines
 - Historically important but nowadays rarely used
 - Building blocks of Deep Belief Networks
- Autoencoders
 - Non-linear dimensionality reduction
 - Extensions to generative models, e.g. variational autoencoders
- Generative Adversarial Networks
 - Currently most widely used generative model
 - Many applications of the general concept, e.g. for segmentation, reconstruction, semi-supervised learning and more



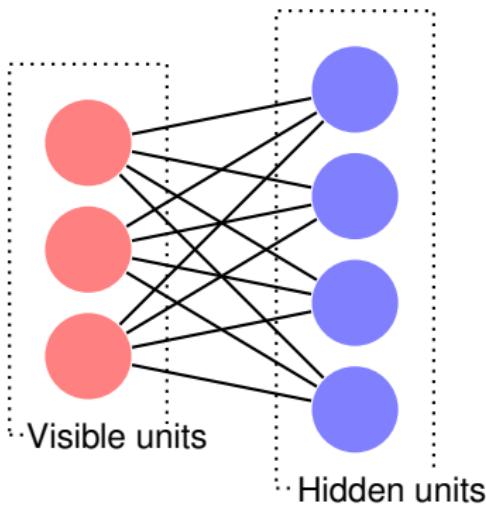
FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Restricted Boltzmann Machines



Restricted Boltzmann Machine (RBM)



- Binary (Bernoulli) visible units \mathbf{v} represent observed data
- Binary (Bernoulli) hidden units \mathbf{h} capture dependencies (latent variables), learn representation of the input

Restricted Boltzmann Machine (RBM)

- The RBM is an energy-based model with a joint probability function $p(\mathbf{v}, \mathbf{h})$ defined in terms of an energy function:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}$$

b, c: biases; **W:** connection matrix

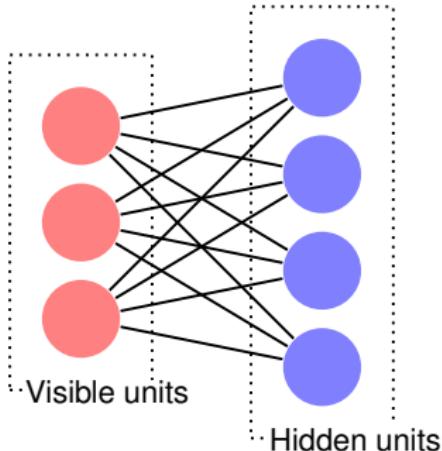
Z: “partition function” (normalization constant),
sum over all possible hidden/visible pairs:

$$Z = \sum_{\mathbf{v}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$$

- $p(\mathbf{v}, \mathbf{h})$ is the so called Boltzmann distribution - closely related to the softmax function!
- Note: RBM is not a FC layer! **Not feed-forward**
- RBM’s “hidden layer” models the input layer in a **stochastic** manner and is **trained unsupervised**

Training RBMs

- Visible and hidden units: bipartite graph
- RBMs are Markov Random Fields (MRFs) with hidden variables
- We want to find \mathbf{W} such that $p(\mathbf{v}, \mathbf{h})$ is high for low energy states and vice versa.
- Learning based on gradient descent on the negative log-likelihood



Training RBMs

θ : model parameters

$$\log L(\theta | \mathbf{v}) = p(\mathbf{v} | \theta)$$

$$= \log \frac{1}{Z} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$$

$$= \log \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) - \log \left(\sum_{\mathbf{v}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \right)$$

$$\frac{\partial \log L(\theta | \mathbf{v})}{\partial \theta} = \dots \quad \text{see [5]}$$

$$= \underbrace{\sum_{\mathbf{h}} p(\mathbf{h}, \mathbf{v}) \frac{-\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}}_{\mathbb{E}_{\text{data}}} - \underbrace{\sum_{\mathbf{v}, \mathbf{h}} p(\mathbf{v}, \mathbf{h}) \frac{-\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}}_{\mathbb{E}_{\text{model}}}$$

$\mathbb{E}_{\text{model}}$ generally intractable

→ Approximate via contrastive divergence

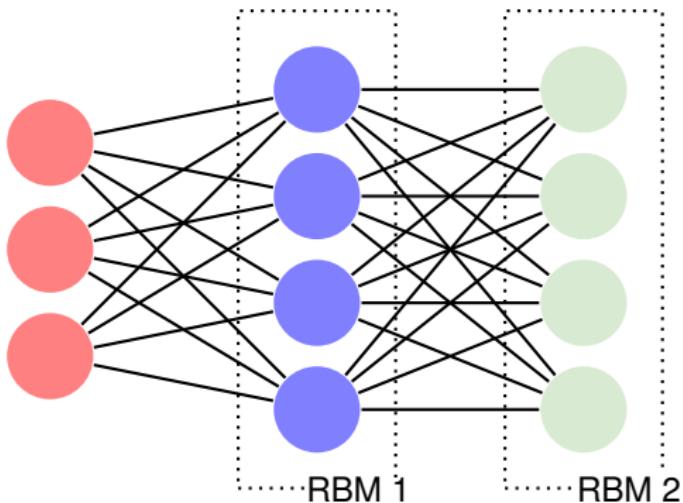
Training RBMs

Contrastive divergence

1. Take any training example as \mathbf{v}
2. Set binary states of the hidden units: $p(h_i = 1 | \mathbf{v}) = \sigma(\sum_j w_{ij} v_j + c_i)$
3. Run k Gibbs sampling steps:
 - 3.1 Sample reconstruction $\tilde{\mathbf{v}}$ by $p(v_j = 1 | \mathbf{h}) = \sigma(\sum_i w_{ij} h_i + b_j)$
 - 3.2 Resample $\tilde{\mathbf{h}}$
4. $\Delta w_{ij} = \eta(\mathbf{v}\mathbf{h}^\top - \tilde{\mathbf{v}}\tilde{\mathbf{h}}^\top)$
 $\Delta b = \eta(\mathbf{v} - \tilde{\mathbf{v}})$
 $\Delta c = \eta(\mathbf{h} - \tilde{\mathbf{h}})$

The more iterations of Gibbs sampling, the less biased the estimate of the gradient. In practice: $k = 1$

Deep Belief Network (DBN)



- Stack multiple RBMs and train them layer by layer
- Last layer can also be fine-tuned for a classification task
- One of the first successful deep architectures [9]
- Sparked the “deep learning renaissance” (nowadays rarely used anymore)

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Unsupervised Deep Learning - Part 2

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020





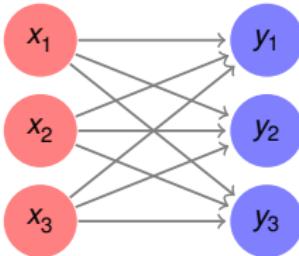
FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Autoencoder



Autoencoder (AE)



Concept

- Special case of feed-forward neural network
- **Encoder:** $y = f(x)$
- ... but how do we get a **loss**?
- **Decoder:** $\hat{x} = g(y)$
- So we train for $\hat{x} = x$
- AE tries to learn an approximation of the identity

Autoencoder – Loss Functions

$$\rightarrow L(\mathbf{x}, \mathbf{x}') \propto -\log p(\mathbf{x}|\mathbf{x}')$$

Squared L₂ norm

$$p(X|\mathbf{x}') \sim \mathcal{N}(\mathbf{x}', \sigma^2 \mathbf{I})$$

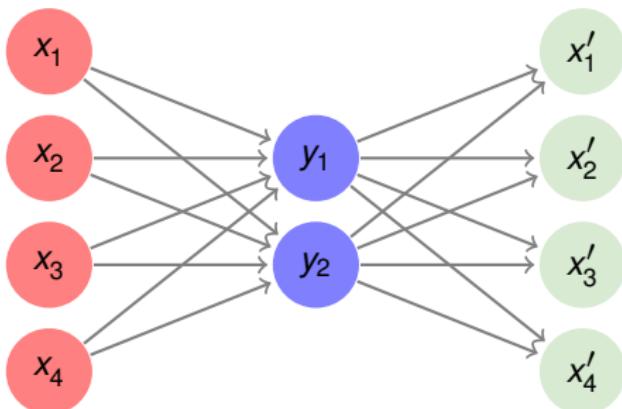
$$L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$$

Cross entropy

$$p(X|\mathbf{x}') \sim \mathcal{B}(\mathbf{x}')$$

$$L(\mathbf{x}, \mathbf{x}') = -\sum_{i=1}^n x_i \log(x'_i) + (1 - x_i) \log(1 - x'_i)$$

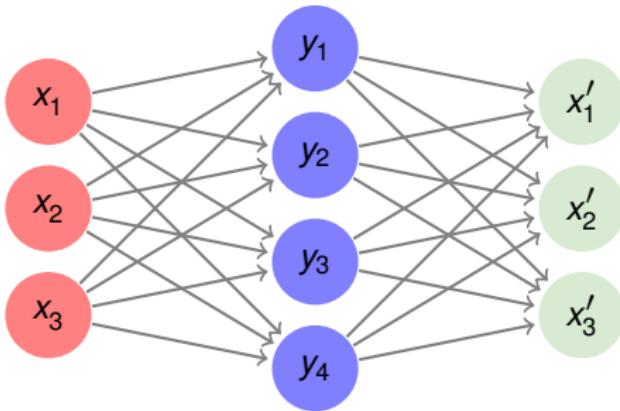
Enforce Information Compression



Undercomplete AE

- Prevent network from simply copying the input
- AE with linear layers and squared L₂ norm as loss learns a PCA
- AE with nonlinear layers: nonlinear PCA generalization

Sparse Autoencoder



- May include **more** hidden units compared to input units (rather than fewer)
- Bottleneck by enforcing sparsity in the **activations** (as opposed to weights!):

$$L_{\text{SAE}}(\mathbf{x}, \mathbf{x}', \mathbf{w}) = L(\mathbf{x}, \mathbf{x}', \mathbf{w}) + \Omega(\mathbf{y})$$

y: activations of the bottleneck layer

Ω: sparsity penalty on **y**, e.g., L_1/L_2 norm or defined via KL-divergence [18]

Autoencoder Variations / Important Terms

Convolutional Autoencoder

Replace fully connected layer with convolutional layer (optionally add pooling layers)

Denoising Autoencoder (DAE)

- Corrupt the input
 - Noise models $C(\hat{\mathbf{x}}|\mathbf{x})$:
 - $C(\hat{\mathbf{x}}|\mathbf{x}) \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})$
 - Set random elements to zero
- Denoising
- Regularization (similar to dropout but applied to input layers)

Autoencoder Variations / Important Terms

DAE as Generative Model

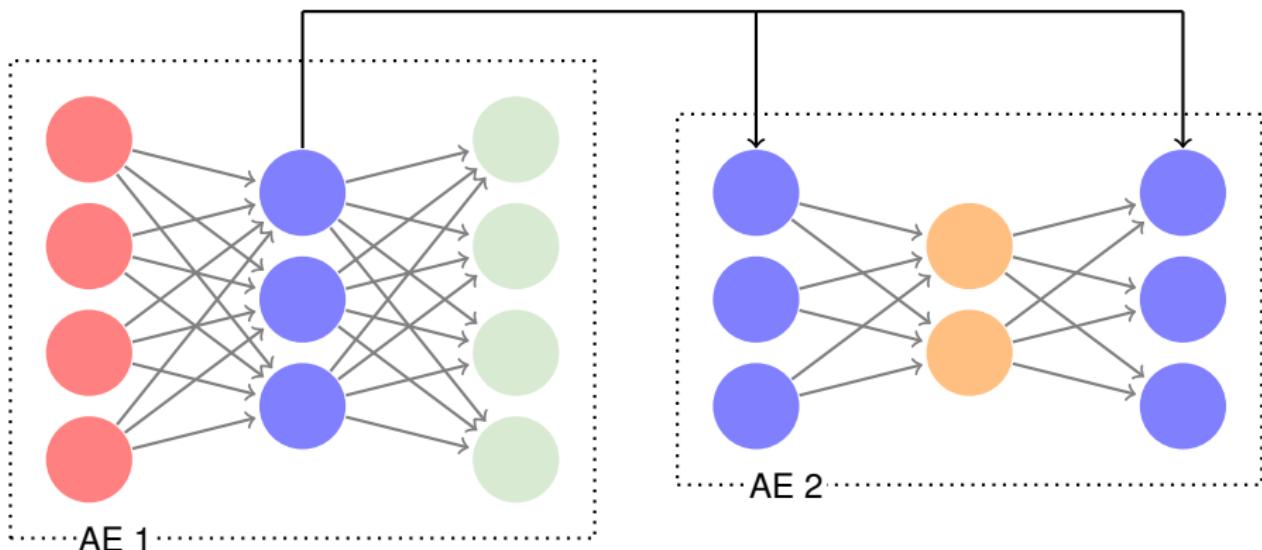
- DAEs implicitly estimate the underlying data-generating process
- Given $C(\hat{\mathbf{x}}|\mathbf{x})$ the DAE learns $p(\mathbf{x}|\hat{\mathbf{x}})$
- Intuition: If \mathbf{x} is typical sample, then iteratively applying the noise and the denoising will reproduce the sample very often
- Estimator $p(\mathbf{x})$: Markov chain sampling by alternating denoising model $p(\mathbf{x}|\hat{\mathbf{x}})$ and corruption process $C(\hat{\mathbf{x}}, \mathbf{x})$
- Converging MC yields estimator $p(\mathbf{x})$

Often expensive and hard to assess convergence

→ Variational AEs much more common!

Autoencoder Variations / Important Terms

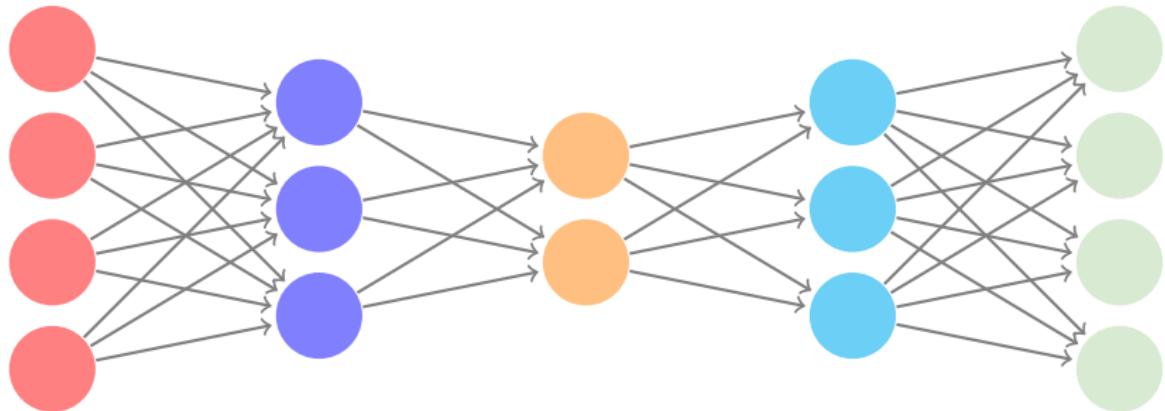
Stacked Autoencoder



Option 1: Train stacked AE layer by layer (originally proposed)

Autoencoder Variations / Important Terms

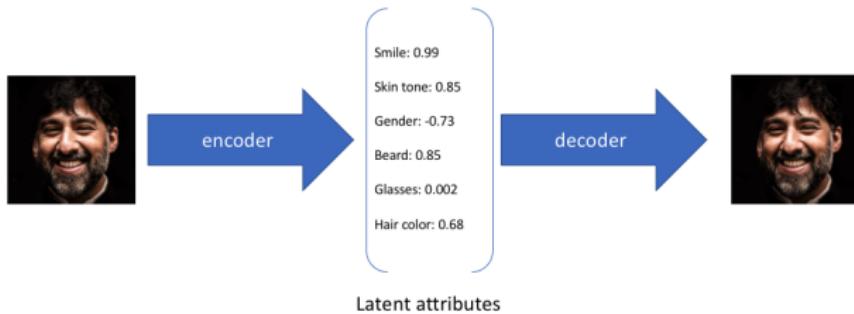
Stacked Autoencoder



Option 2: Train stacked AE by means of multiple layers

Variational Autoencoders

- “Traditional” AEs compute a **deterministic** feature vector describing the attributes of the input in latent space

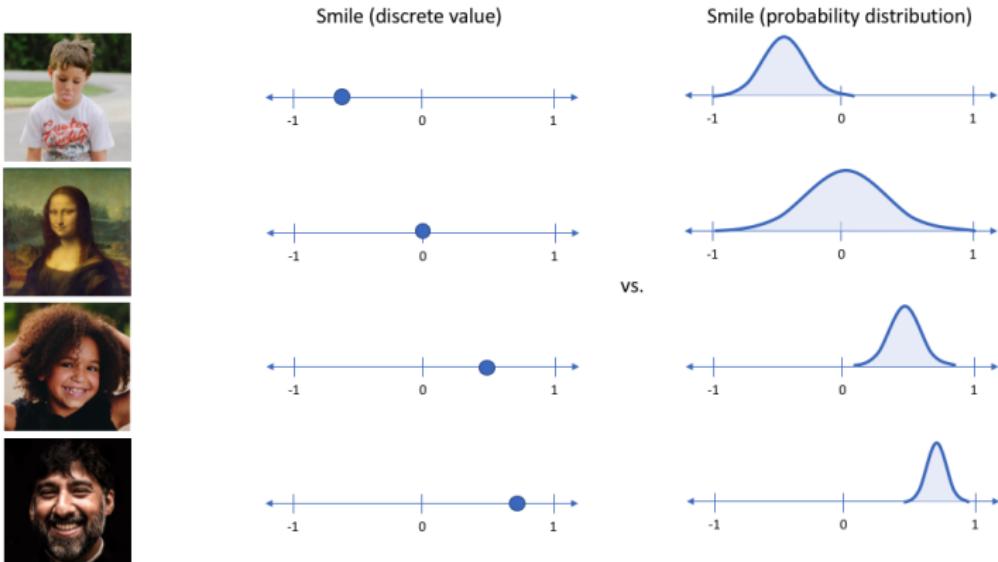


- Key difference in **variational autoencoders**:
 - Uses a variational approach to learn the latent representation
 - Allows to describe observation in latent space in **probabilistic manner**

Source: <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders: Motivation

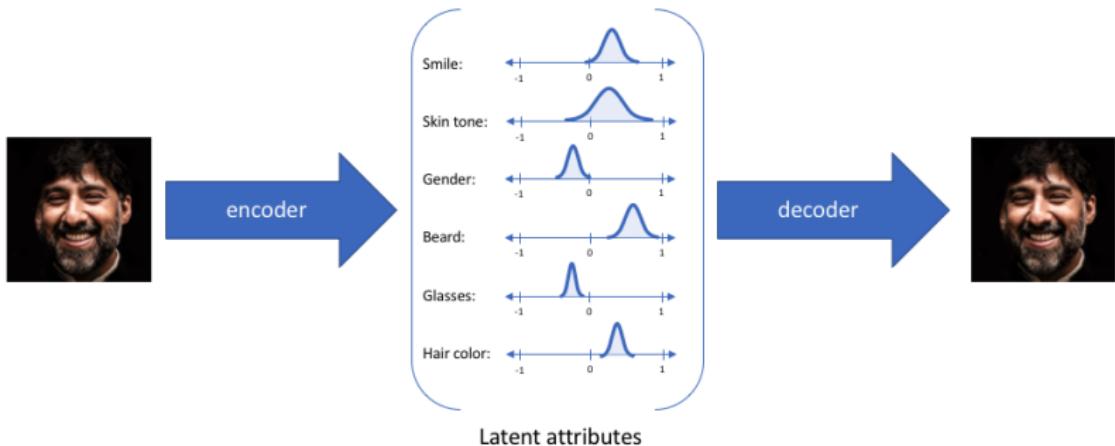
- Describe each latent attribute as **probability distribution**
- Allows to model uncertainty in the input data



Source: <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders

- Decoding: **Sample** from latent space as input for decoder model



- Representation as probability distribution enforces a continuous, smooth latent space representation
- Similar latent space vectors should correspond to similar reconstructions

Source: <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders: Statistical Motivation

- Assumption: Hidden (latent) variable z that generates an observation x
- Training a variational autoencoder: determining the distribution of z
- Computing arbitrary $p(z|x)$ - usually intractable
- Approximate $p(z|x)$ by tractable distribution $q(z|x)$ → Determine parameters of $q(z|x)$
- Minimize Kullback-Leibler-divergence between $p(z|x)$ and $q(z|x)$:

$$\min \text{KL}(p(z|x), q(z|x)) \quad (1)$$

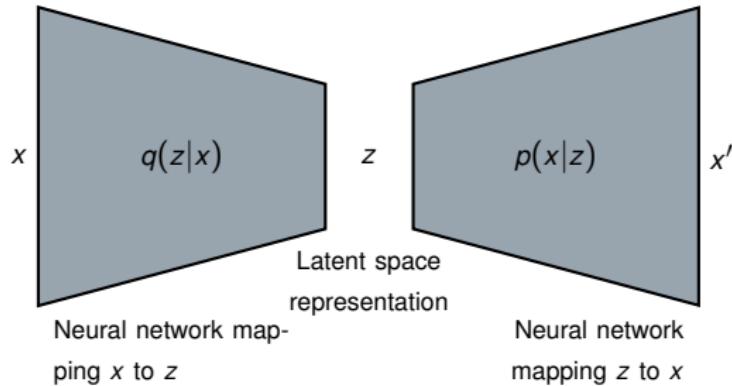
which is equivalent to

$$\max \underbrace{\mathbb{E}_{q(z|x)} \log p(x|z)}_{\text{reconstruction likelihood}} - \text{KL}(q(z|x), p(z)) \quad (2)$$

which forces $q(z|x)$ to be similar to true prior distribution $p(z)$

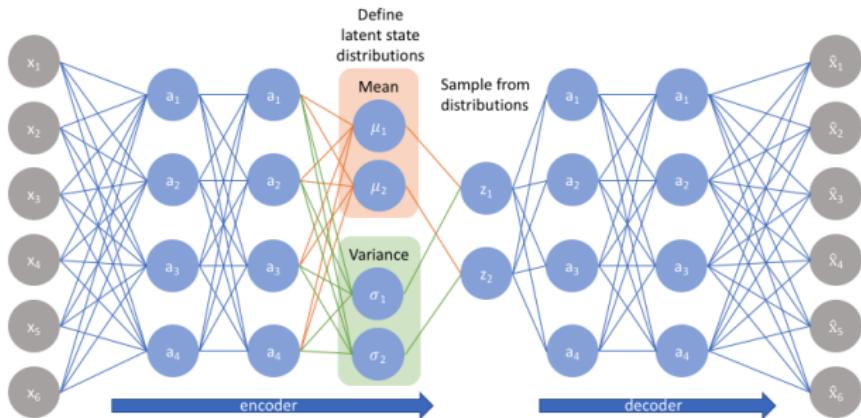
Variational Autoencoders: Statistical Motivation

- $p(z)$ often assumed to be (isotropic) Gaussian distribution
- Determining $q(z|x)$ boils down to estimating μ and σ
- Use **neural network** to estimate $q(z|x)$ and $p(x|z)$



Source: After <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders: Training

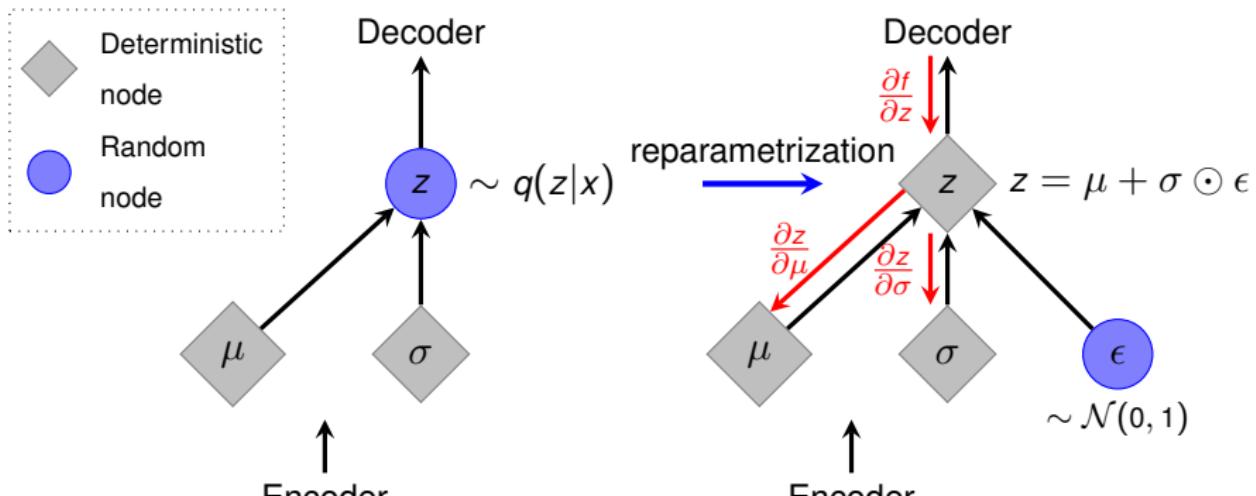


- Loss function: $L(\theta, \phi; x, z) = \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \text{KL}(q_\phi(z|x), p(z))$
- Problem: Network contains **sampling** operator → we cannot backpropagate through!

Source: <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders: Reparametrization Trick

- We cannot backpropagate through random sampling - what now?
- “Push” random sampling out of backpropagation path by reparametrization:

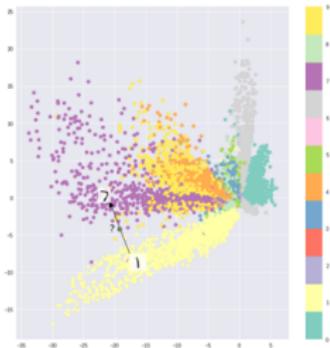


→ Deterministic backpropagation

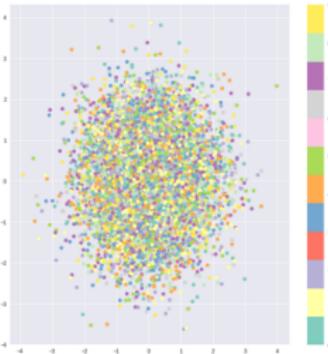
Source: after: <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders: Latent Space Visualization

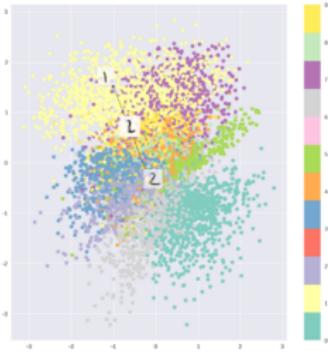
Only reconstruction loss



Only KL divergence



Combination



Samples well separated but no smooth transition

Not able to describe original data

Simultaneous optimization allows to describe input with distributions close to prior

Source: <https://www.jeremyjordan.me/variational-autoencoders/>

Variational Autoencoders as Generative Models

- New data can be generated by sampling from distributions in the latent space
→ reconstructed by decoder
- Diagonal prior enforces independent latent variables
→ Can encode different factors of variation
- Example: Smoothly varying degree of smile and head pose [12]



Source: Kingma and Welling [12]

Variational Autoencoders: Summary

- Probabilistic models → allow to generate data
- Intractable density → optimize variational lower bound instead
- Trained via back propagation by using reparametrization
- Pros:
 - Principled approach to generative models
 - Latent space representation can be useful for other tasks
- Cons:
 - Only maximizes lower bound of likelihood
 - Samples in standard models often of lower quality compared to GANs
- Active area of research!

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Unsupervised Deep Learning - Part 3

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Generative Adversarial Networks

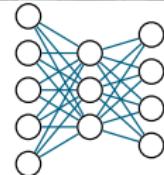


Let's Play a Game (or the Principle of GANs)

Discriminator (D)



Generator ($G; z$)



real:/ fake: 0



...

Training GANs – Minimax Game

Alternate between:

1. Train D : minimize

$$L^D(\theta^D, \theta^G) = -\mathbb{E}_{x \sim p_{\text{data}}} \log D(\underbrace{x}_{\text{real}}) - \mathbb{E}_{z \sim p_z} \log(1 - \underbrace{D(G(z))}_{\text{fake}})$$

→ Trained to distinguish real data samples from fake ones

2. Train G : minimize $L^G = -L^D$

Generator minimizes log-probability of the discriminator being correct

→ Trained to generate data domain images and fool D

- Optional: run k steps of one player for every step of the other player
- Equilibrium is a saddle point of the discriminator loss

Training GANs – Minimax Game (cont.)

$$L^D = -\frac{1}{2} \mathbb{E}_{\substack{\mathbf{x} \sim p_{\text{data}}}} \log D(\underbrace{\mathbf{x}}_{\text{real}}) - \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_z} \log(1 - \underbrace{D(G(\mathbf{z})))}_{\text{fake}})$$

$$L^G = -L^D$$

- L^G is tied directly to $-L^D$
- Summarize game with a **value function** specifying the discriminator's payoff:

$$V(\theta^D, \theta^G) = -L^D(\theta^D, \theta^G)$$

Minimax game:

$$\hat{\theta}^G = \arg \min_{\theta^G} \max_{\theta^D} V(\theta^D, \theta^G)$$

Training GANs – Optimal Discriminator

- Assumption: both densities are nonzero everywhere
- Otherwise some input values are never trained → some $D(\mathbf{x})$ have undetermined behavior
- Solve for: $\frac{\partial L^D}{\partial D(\mathbf{x})} = 0$
- Optimal $D^*(\mathbf{x})$ for any $p_{\text{data}}(\mathbf{x})$ and $p_{\text{model}}(\mathbf{x})$:

$$D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

- Unfortunately this optimal discriminator is theoretical and unachievable
- GAN's key approximation mechanism!
- GANs use supervised learning to estimate this ratio
- Underfitting / Overfitting

Non-Saturating Games – Modify Generator's Loss

$$L^D = -\frac{1}{2} \mathbb{E}_{\substack{\mathbf{x} \sim p_{\text{data}}}} \log D(\underbrace{\mathbf{x}}_{\text{real}}) - \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_z} \log \left(1 - \underbrace{D(G(\mathbf{z}))}_{\text{fake}} \right)$$

$$L^G = -\frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_z} \log D(G(\mathbf{z}))$$

- In Minimax: G minimizes log-probability of D being correct
Here: G maximizes log-probability of D being mistaken
- Heuristically motivated: fights vanishing gradient of G when D is “too smart” (esp. in the beginning)
- Equilibrium no longer describable with single loss

Other Popular Loss Functions

Feature matching loss / perceptual loss

- G trained to match expected value of features $f(\mathbf{x})$ of intermediate layer of D :

$$L^G = \|\mathbb{E}_{\mathbf{x} \sim p_{data}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_z} f(G(\mathbf{z}))\|_2^2$$

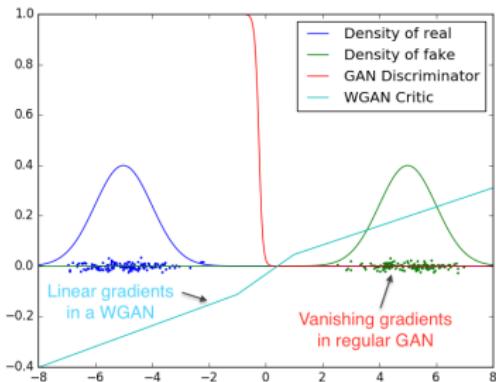
- Prevent “overtraining” of G on current D
- Popular loss also in other domains

Other Popular Loss Functions

Wasserstein Loss

- Derived from Wasserstein distance a.k.a. Earth Movers Distance
- Learn discriminator D that maximizes the discrepancy between real and fake and whose gradient remains beyond a limit:

$$\max_{\|D\|_L \leq 1} \mathbb{E}_{x \sim p_{\text{data}}} (D(x)) - \mathbb{E}_{x \sim p_{\text{model}}} (D(x))$$



- If weights could grow arbitrarily large, the gradient would be unbounded
- Clip D 's weights → Critic lies within space of 1-Lipschitz functions
- Helps to counter vanishing gradients in D
- Many more loss functions exist, e. g. KL divergence (then GAN do ML)
- **But the approximation strategy matters more than the loss**

How to Evaluate GANs

Inception Score [17]

- Goals:
 1. Generated image should be recognizable (standard: Inception v-3 pre-trained on Imagenet)
 - Score distribution should be dominated by one class
 - Image-wise class distribution should have low entropy
 2. Generated images should be diverse
 - Class distribution uniform
 - Entropy should be high
- KL distance between distributions (higher: better):

$$\text{IS}(\mathbf{x}) = \exp \left\{ \mathbb{E} [\text{KL}(p(y|\mathbf{x}) \| p(y))] \right\}$$

How to Evaluate GANs

Fréchet Inception Distance (FID) [8]

- Use intermediate layer (last pooling layer of Inception-v3 pre-trained on ImageNet)
- Model data distribution by multivariate Gaussians (μ , Σ)
- FID score between real images \mathbf{x} and generated images \mathbf{g} (lower: better)

$$\text{FID}(\mathbf{x}, \mathbf{g}) = \|\mu_{\mathbf{x}} - \mu_{\mathbf{g}}\|^2 + \text{Tr}(\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{g}} - 2\sqrt{(\Sigma_{\mathbf{x}}\Sigma_{\mathbf{g}})})$$

- + More robust to noise than IS
- + No class concept needed

GANs in Comparison to Other Generative Models

- Ability to generate samples in parallel
- Very few restrictions (e.g. compared to Boltzman machines)
- No Markov chain needed!
- No variational bound is needed

GANs known to be asymptotically consistent since the model families are universal function approximators

NEXT TIME
ON DEEP LEARNING



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Unsupervised Deep Learning - Part 4

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020



Conditional GANs

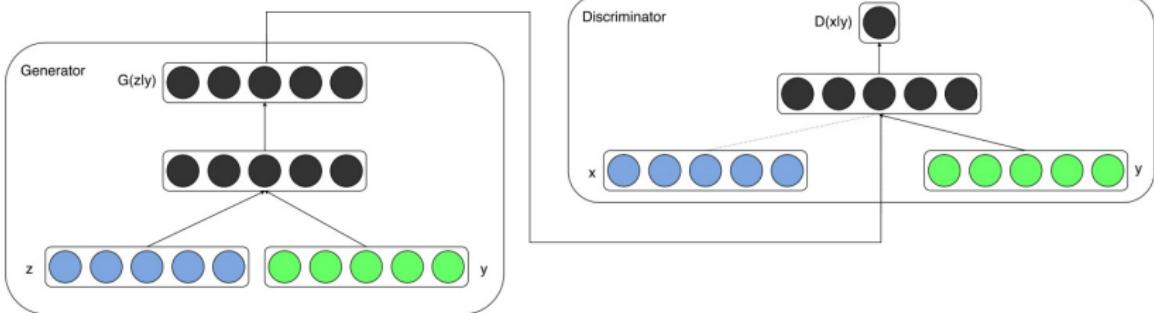
- Problem: Generator creates a “fake” generic image → is not specific for a certain condition/characteristic
- Example: **text to image generation** – image should **depend** on the text
- Idea: Provide additional vector **y** to networks to encode **conditioning** [15]



Generated samples conditioned on one label (digit)

Source: Mirza et al. [15]

Conditional GANs (cont.)



Generator and discriminator in CGANs.

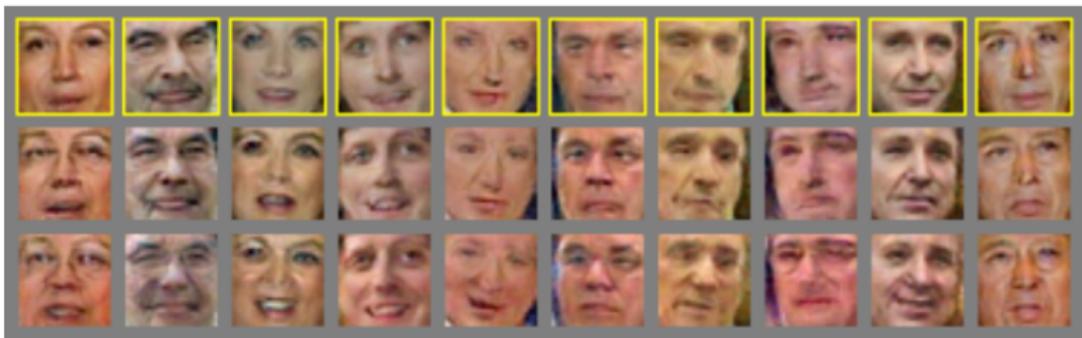
- Generator G receives the latent vector \mathbf{z} and a conditioning vector \mathbf{y}
- Discriminator D receives \mathbf{x} and also \mathbf{y}
- The objective function of a two-player minimax game changes to:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))]$$

Source: Mirza et al. [15]

Example: Conditional GANs for Face Generation

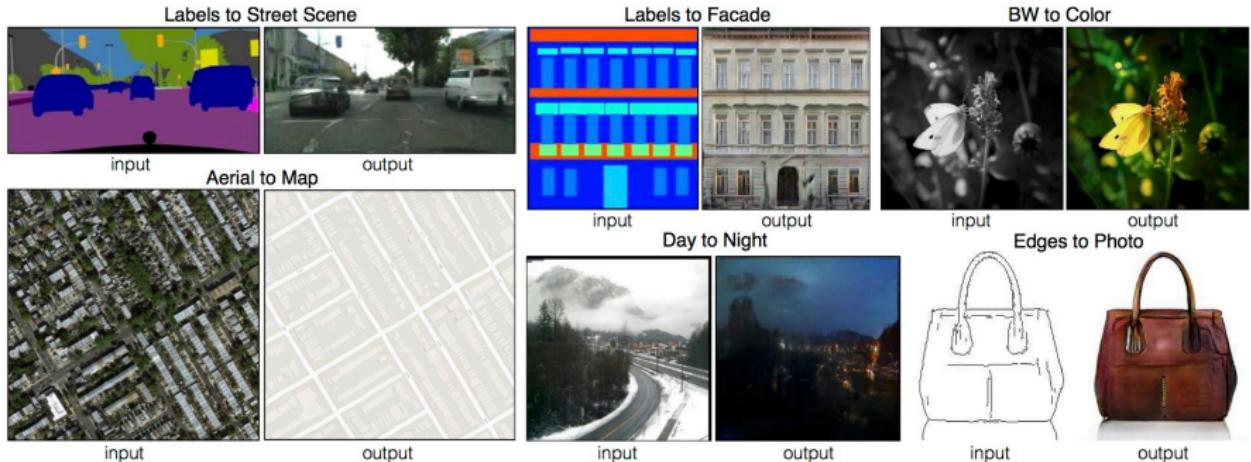
- Add conditional feature (e.g., smiling, gender, old age, ...)
- Generator/Discriminator learn to operate in **modes**:
 - Generator learns to generate a face with a certain attribute
 - Discriminator learns to decide whether the face contains attribute



First row: random samples, second row: $y \sim \text{old age}$, third row: $y \sim \text{old age} + \text{smiling}$.

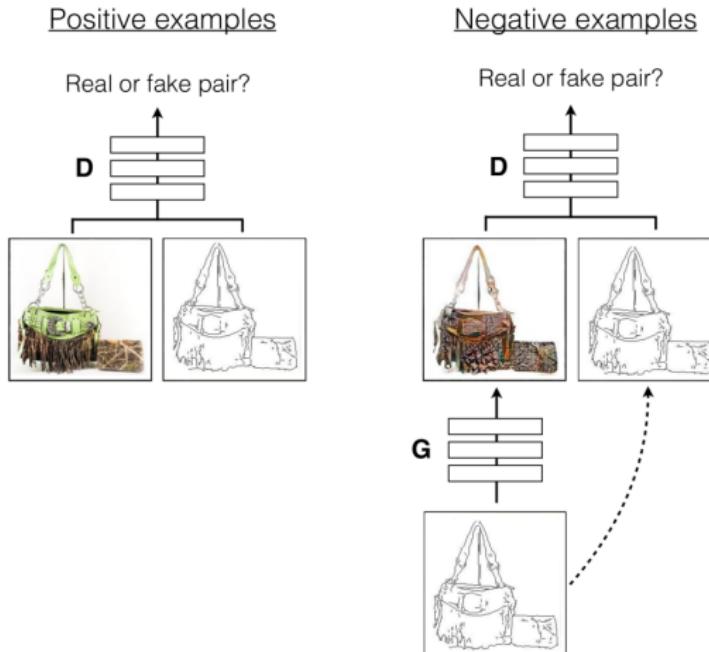
Source: Gauthier [6]

Image To Image



Source: [11]

Image To Image - Just a Conditional GAN!

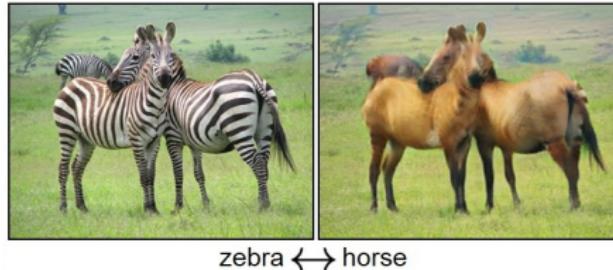


Source: [11]

Cycle Consistent GANs [22]

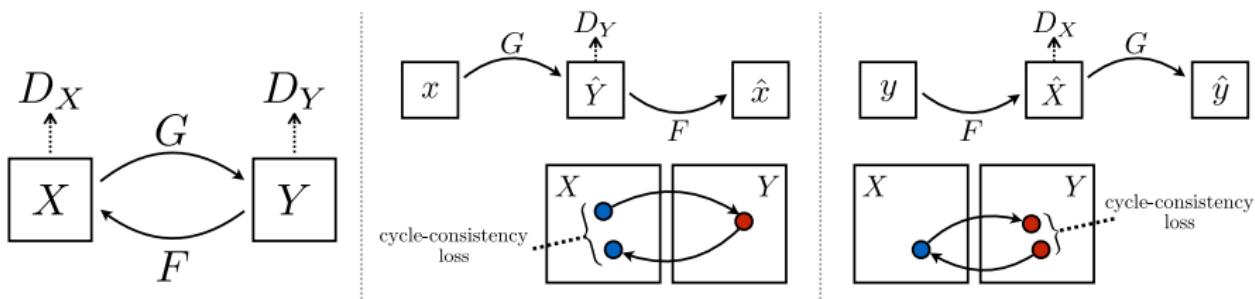
- Image to Image GAN should generate plausible results **w.r.t. input**
- **Paired data** difficult/impossible to obtain
- Cycle consistency loss: Couple GAN with trainable **inverse** mapping F such that

$$F(G(\mathbf{x})) \approx \mathbf{x} \quad \text{and} \quad G(F(\mathbf{y})) \approx \mathbf{y} \quad (3)$$



Source: Adapted from [22]

Cycle Consistency Loss [22]



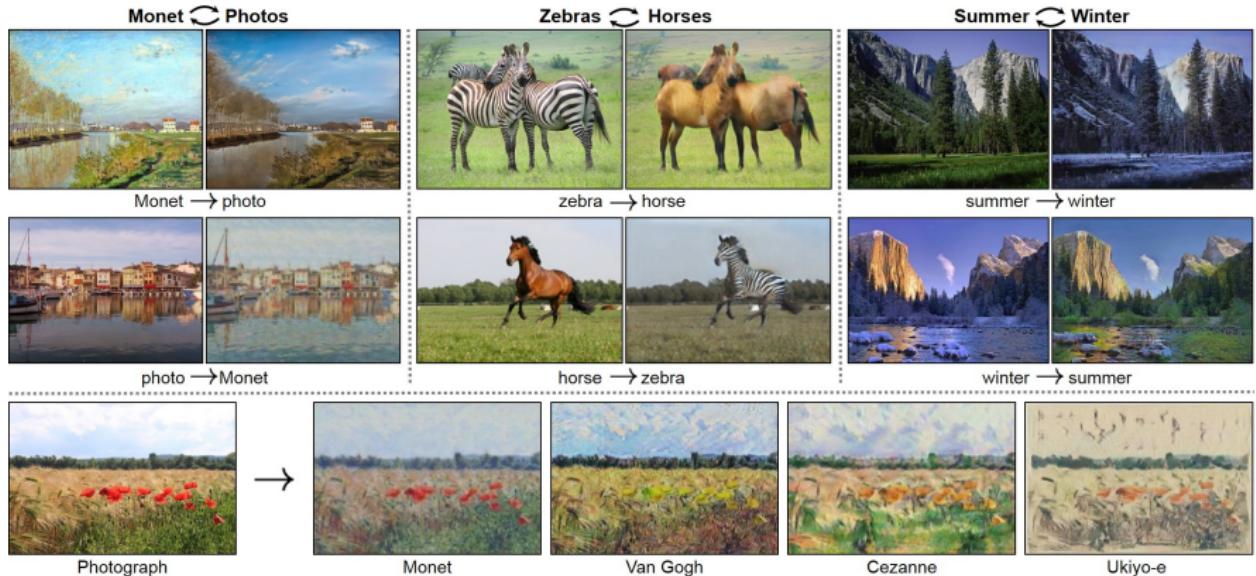
- Two discriminators D_Y and D_X
- Cycle consistency loss for **two generators** G, F :

$$\begin{aligned} L_{\text{cyc}}(G, F) = & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\|F(G(\mathbf{x})) - \mathbf{x}\|_1] \\ & + \mathbb{E}_{\mathbf{y} \sim p_{\text{data}}(\mathbf{y})} [\|G(F(\mathbf{y})) - \mathbf{y}\|_1] \end{aligned}$$

- Total loss:
- $$L(G, F, D_X, D_Y) = L_{\text{GAN}}(G, D_Y, X, Y) + L_{\text{GAN}}(F, D_X, Y, X) + \lambda L_{\text{cyc}}(G, F)$$

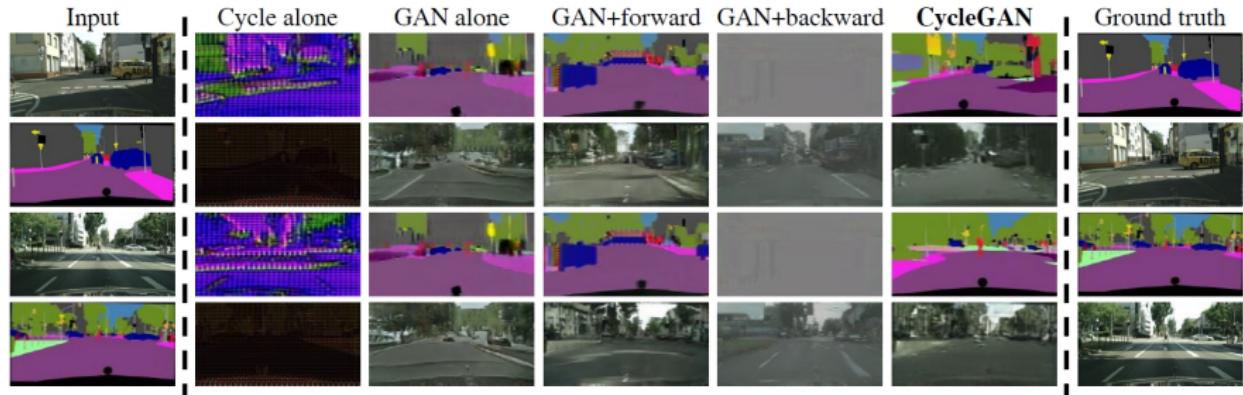
Source: Adapted from [22]

CycleGAN: Examples



Source: [22]

CycleGAN: Examples (cont.)



Ablation study for CycleGAN

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Unsupervised Deep Learning - Part 5

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

More Tricks of the Trade



One-sided Label Smoothing

- Replace targets of the real samples with a smoothed version
 - replace 1 with 0.9
- Do **not** do the same for fake samples (don't change 0 label)
Otherwise D will reinforce incorrect behavior → G will produce samples that resemble the data or samples it already makes
- Benefits
 - Prevents D from giving very large gradient signal to G
 - Prevents extrapolating to encourage extreme samples

Balancing G and D necessary?

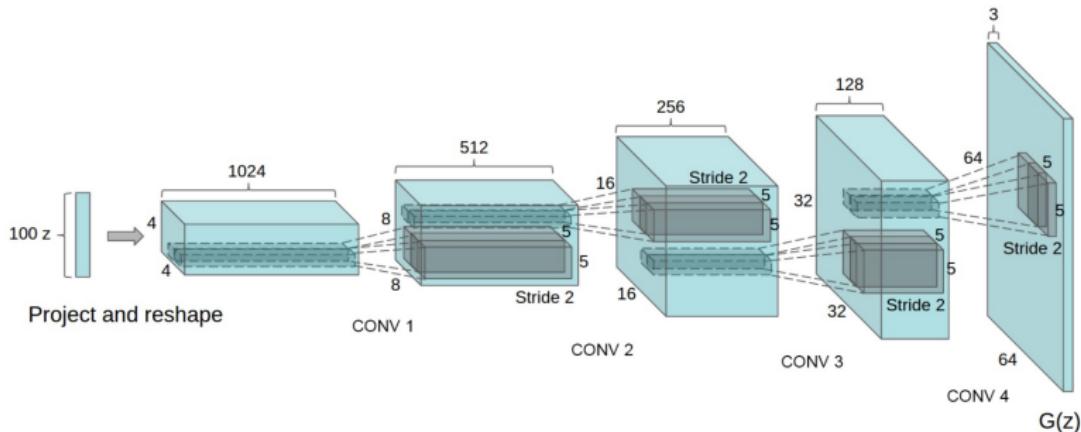
No.

- GANs work by estimating ratio of data and model density
 - Ratio estimated correctly only when D is optimal
 - Fine if D overpowers G

But when D gets too good

- G 's gradient may vanish
 - Use non-saturating loss
- G 's gradient may get too large
 - Use label smoothing

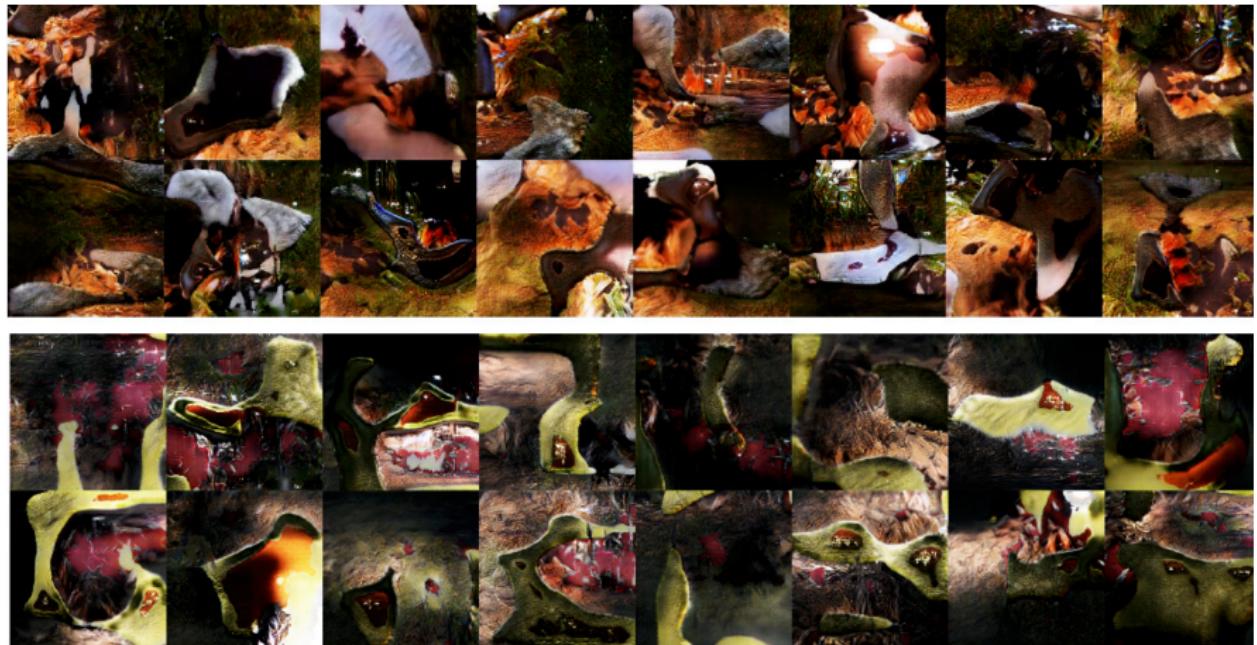
Deep Convolutional GANs (DCGAN)



- Replace any pooling layer with strided convolutions (D) and transposed convolution (G)
- Remove fully connected hidden layers for deeper architectures
- G : Use ReLU activation except for output layer which uses tanh
- D : LeakyReLU activation for all layers
- Use batch normalization

Source: [16]

Problem of Batch Normalization in G



→ Causes strong intra-batch correlation

Virtual Batch Normalization VBN

- Don't use one BN instance for both minibatches (containing only real / fake)!
- Use two separate BN or better use VBN
- If BN/VBN is too expensive → choose instance normalization (for each sample subtract mean and divide by standard deviation)

Virtual Batch Normalization

- Create **reference batch** R of random samples chosen and fixed once at the start of training
- For each \mathbf{x}_i of the current mini-batch
 - Create a new **virtual batch** $V_i = R \cup \mathbf{x}_i$
 - Compute mean and standard deviation of V_i
 - We always need to propagate R forward **in addition** to the current batch
 - Normalize \mathbf{x}_i with these statistics

Historical Averaging

- Add penalty term that punishes weights which are rather far away from their historical average:

$$\| \theta - \frac{1}{T} \sum_{i=1}^T \theta[i] \| ^2$$

where $\theta[i]$ is the value of parameters at past time i

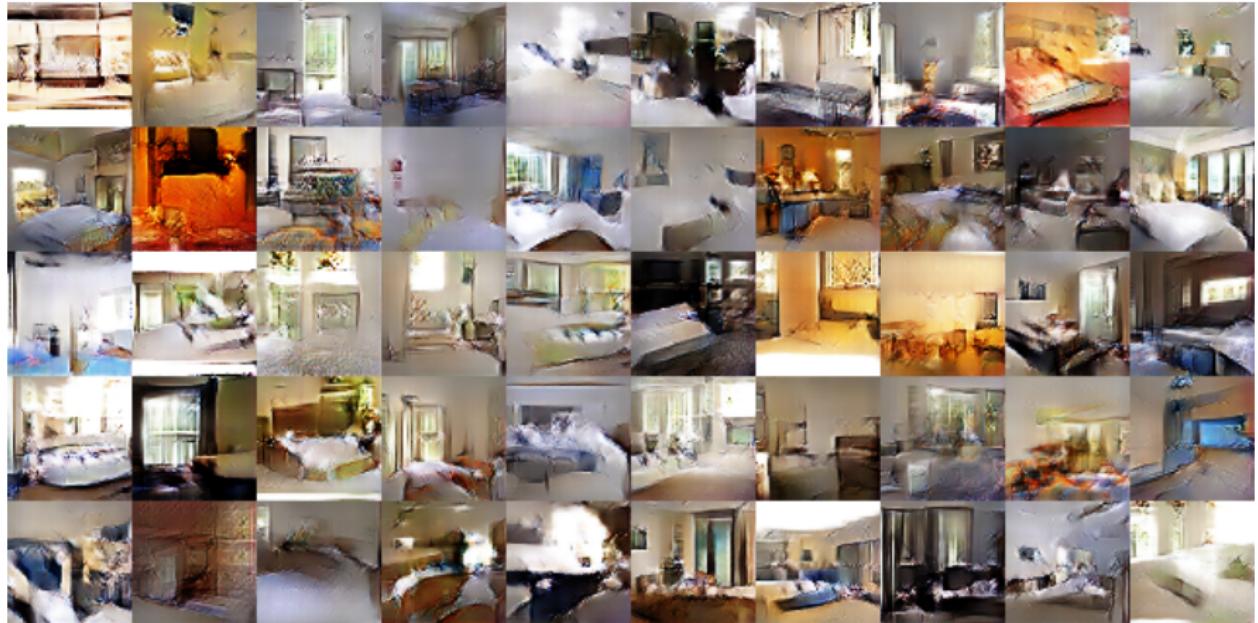
- Historical average of the parameters can be updated in an online fashion

Similar tricks from RL also work for GANs, e.g. experience replay:

- Keep a replay buffer of past generations and occasionally show them
- Keep checkpoints from the past of G and D and occasionally swap them out for a few iterations

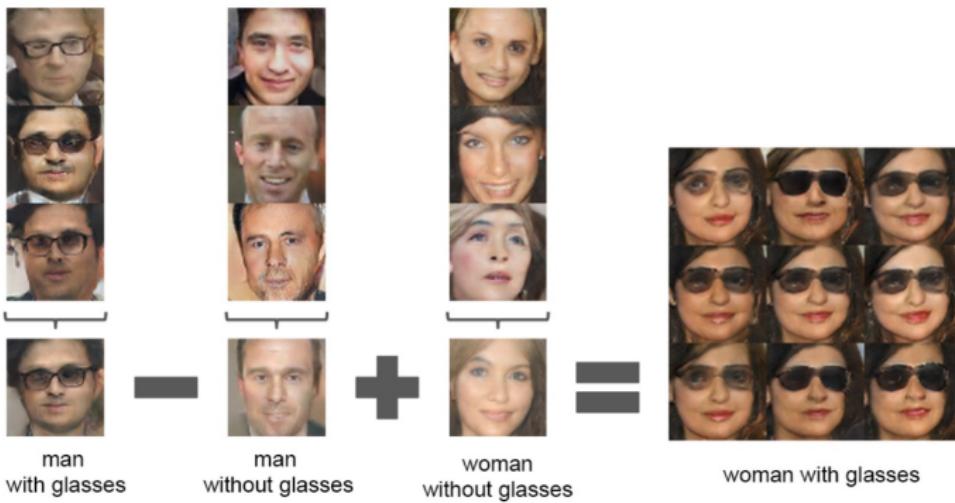
DCGAN examples

Bedrooms after 1 epoch



Source: [16]

Vector Arithmetic



- Average three latents codes z and apply operation
- GANs learn a distributed representation that disentangles the concept of gender from the concept of wearing glass
- See also “InfoGAN” [1]

Source: [16]



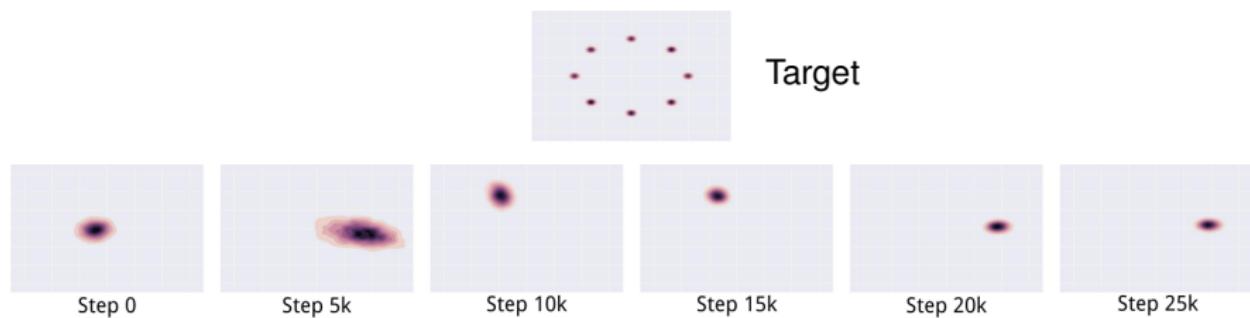
FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Advanced GAN Methods



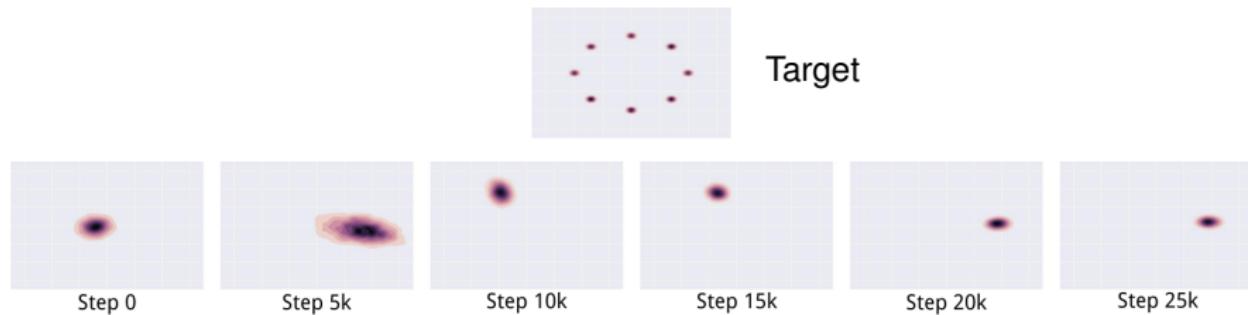
Mode Collapse



- G rotates through the modes of the data distribution
- Never converges to a fixed distribution

Source: [14]

Mode Collapse



Possible Reason

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$$

- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point
- In practice: Simultaneous SGD of both networks → both effects can appear
- Solutions: Minibatch discrimination or unrolled GANs

Source: [14]

Minibatch Discrimination

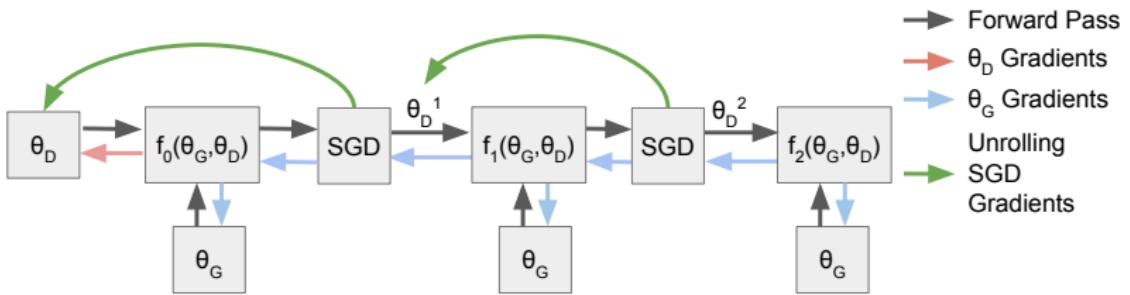
Intuition: Allow D to look at multiple samples in combination to help G avoid collapsing

1. Extract features from intermediate layer
 2. Add **minibatch layer** that computes for each feature
 1. a similarity to all other samples of the mini-batch
 2. concatenate similarity vector to each feature
- Compute these minibatch features separately for samples from G and from training data
 - D still outputs 0/1 but now uses the similarity to all samples in the mini-batch as side information

Unrolled GAN

- Ideally: $G^* = \min_G \max_D V(G, D)$
- But essentially, we ignore the max operation when computing G 's gradient
- Idea: Regard $\max_D V(G, D)$ as cost for $G \rightarrow$ need to back-propagate through the maximization operation

Unrolled GAN (cont.)

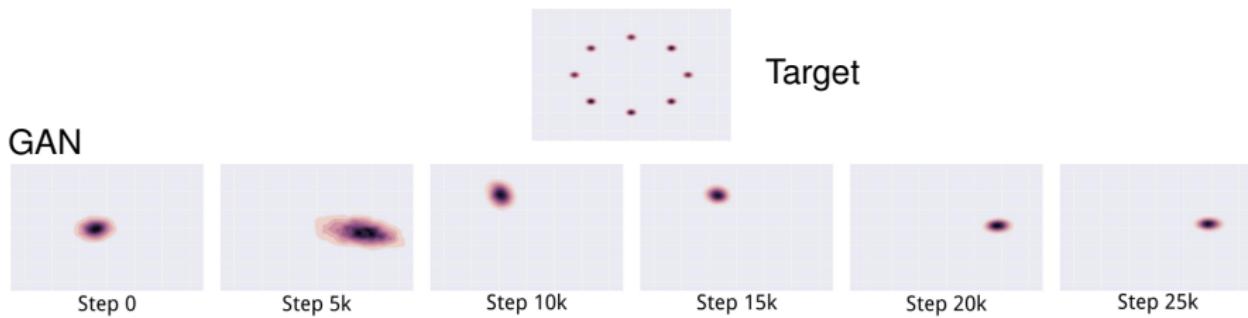


Unrolled GAN with $k = 3$

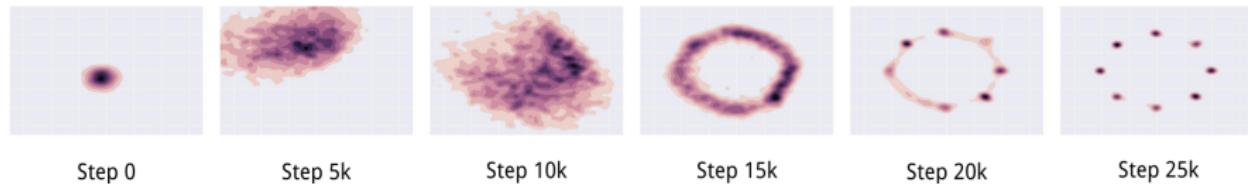
1. Build computational graph describing k steps of learning in D
2. Back-propagate through all k steps when computing G 's gradient
 - Fully maximizing D 's value function intractable
 - Even a low number of $k = 10$ already substantially reduces mode collapse

Source: [14]

Unrolled GAN (cont.)



Unrolled GAN



Source: [14]

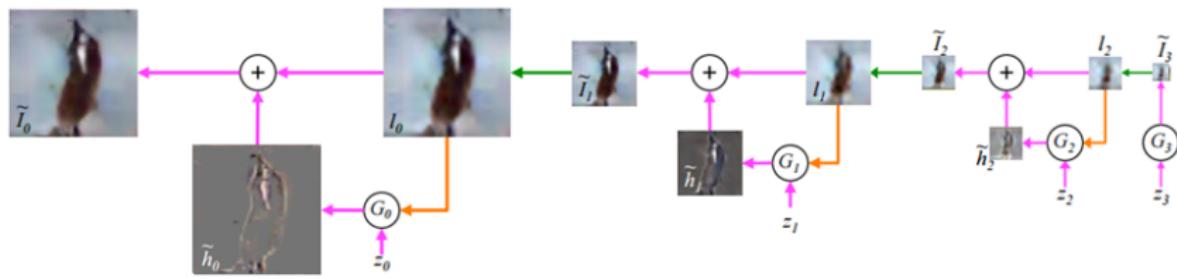
GANs for Semi-supervised Learning

- Idea: Use GANs by turning the problem from a K class problem into a problem with $K + 1$ classes
- “True classes”: Target classes for supervised learning
- Additional class: Fake inputs generated by the generator
- Probability of image being real: sum of “real class” probabilities
- Discriminator is used as a classifier within the GAN game

Source: [14]

Multi-scale approaches: Laplacian Pyramid of GANs

- GANs are pretty good at generating low-resolution images, but
- High-resolution images are much more difficult!
- **Subsequently** increase resolution and add detail with pyramid of GANs
- **Input:** Noise + upsampled output from previous G as **conditioning variable**
- **Output:** **Difference image** for current resolution

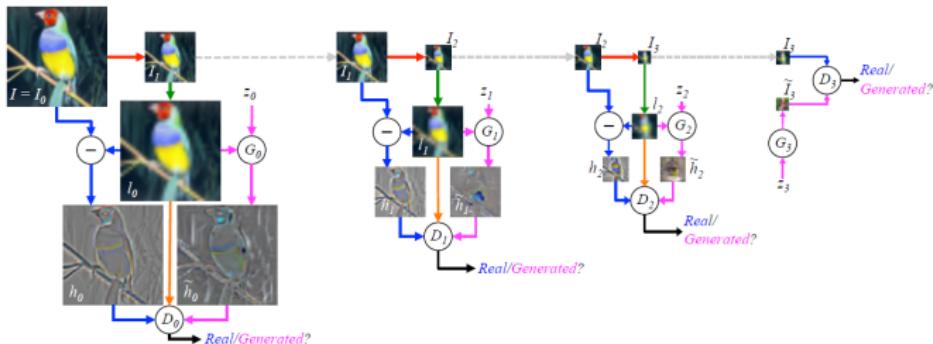


Generator hierarchy. Read right to left!

Source: Denton et al. [3]

Multi-scale approaches: Laplacian Pyramid of GANs (cont.)

- Train generators by training a discriminator on each level
- Input for discriminator: difference images



LAPGAN training.

- But still only 64×64 pixels!

Source: Denton et al. [3]

StackGANs: Text to Photo-realistic Image synthesis

- Task: Given some text, generate a fitting image
- Decompose problem: Sketch-refinement using a two-stage conditional GANs
- Analogon: Human painting – sketch and fine details
- Stage-I GAN: Draws low resolution images
 - Conditioned on text descriptions
 - Sketching rough shape / basic colors from the given text
 - Paints the background
- Stage-II GAN: Adds Generates high resolution images
 - Conditioned on Stage-I result and text descriptions
 - Corrects defects and add details

StackGANs: Text to Photo-realistic Image synthesis (cont.)

Text description	This bird is blue with white and has a very short beak	This bird has wings that are brown and has a yellow belly	A white bird with a black crown and yellow beak	This bird is white, black, and brown in color, with a brown beak	The bird has small beak, with reddish brown crown and gray belly	This is a small, black bird with a white breast and white on the wingbars.	This bird is white black and yellow in color, with a short black beak
Stage-I images							
Stage-II images							

Samples generated by StackGAN [20]

- Stage-I images are blurry with defects and missing details in foreground object
- Stage-II images have $4\times$ higher resolution and “plausible” details

Source: Zhang et al. [19]

Summary

GANs

- are generative models that use supervised learning to approximate an intractable cost function
- can simulate many cost functions
- hard to find equilibrium between D and G
- cannot generate discrete data
- can also be used for
 - (semi-)supervised classification
 - transfer learning
 - multi-modal outputs
 - ...

**NEXT TIME
ON DEEP LEARNING**

Coming Up:

- How to adapt neural networks to localize objects?
- Neural networks detecting object classes.
- Instead of classes detect specific instances of classes.
- Methods to go even finer and segment outlines.
- A neural network worth checking its citations every day.

Comprehensive Questions

- What is the basic idea of contrastive divergence?
- What is the defining characteristic of an autoencoder?
- How do denoising autoencoders work?
- What does an optimal discriminator for GANs learn?
- What are the advantages of GANs in comparison to other generative models?
- What is “mode collapse”?
- Explain feature matching/perceptual loss.

Further Reading

- Variational Autoencoders: http://dpmkingma.com/wordpress/wp-content/uploads/2015/12/talk_nips_workshop_2015.pdf
- NIPS 2016 GAN Tutorial of Goodfellow
<https://www.youtube.com/watch?v=AJVyzd0rqdc>
- How to train a GAN? Tips and tricks to make GANs work (careful, not everything is true anymore!) <https://github.com/soumith/ganhacks>
- Ever wondered about how to name your GAN?
<https://github.com/hindupuravinash/the-gan-zoo>



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

References



References I

- [1] Xi Chen, Xi Chen, Yan Duan, et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: Advances in Neural Information Processing Systems 29. Curran Associates, Inc., 2016, pp. 2172–2180.
- [2] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: Journal of Machine Learning Research 11. Dec (2010), pp. 3371–3408.
- [3] Emily L. Denton, Soumith Chintala, Arthur Szlam, et al. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". In: CoRR abs/1506.05751 (2015). arXiv: 1506.05751.
- [4] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern classification. 2nd ed. New York: Wiley-Interscience, Nov. 2000.

References II

- [5] Asja Fischer and Christian Igel. "Training restricted Boltzmann machines: An introduction". In: Pattern Recognition 47.1 (2014), pp. 25–39.
- [6] John Gauthier.
Conditional generative adversarial networks for face generation. Mar. 17, 2015. URL:
<http://www.foldl.me/2015/conditional-gans-face-generation/>
(visited on 01/22/2018).
- [7] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. 2016. eprint: arXiv:1701.00160.
- [8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: Advances in Neural Information Processing Systems 30. Curran Associates, Inc., 2017, pp. 6626–6637.

References III

- [9] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks.". In: Science 313.5786 (July 2006), pp. 504–507. arXiv: 20.
- [10] Geoffrey E. Hinton. "A Practical Guide to Training Restricted Boltzmann Machines". In: Neural Networks: Tricks of the Trade: Second Edition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619.
- [11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: (2016). eprint: arXiv:1611.07004.
- [12] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: arXiv e-prints, arXiv:1312.6114 (Dec. 2013), arXiv:1312.6114. arXiv: 1312.6114 [stat.ML].

References IV

- [13] Jonathan Masci, Ueli Meier, Dan Ciresan, et al. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". In: Artificial Neural Networks and Machine Learning – ICANN 2011. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 52–59.
- [14] Luke Metz, Ben Poole, David Pfau, et al. "Unrolled Generative Adversarial Networks". In: International Conference on Learning Representations. Apr. 2017. eprint: arXiv:1611.02163.
- [15] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: CoRR abs/1411.1784 (2014). arXiv: 1411.1784.
- [16] Alec Radford, Luke Metz, and Soumith Chintala.
Unsupervised Representation Learning with Deep Convolutional Generative Models. 2015. eprint: arXiv:1511.06434.

References V

- [17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, et al. "Improved Techniques for Training GANs". In: Advances in Neural Information Processing Systems 29. Curran Associates, Inc., 2016, pp. 2234–2242.
- [18] Andrew Ng. "CS294A Lecture notes". In: 2011.
- [19] Han Zhang, Tao Xu, Hongsheng Li, et al. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". In: CoRR abs/1612.03242 (2016). arXiv: 1612.03242.
- [20] Han Zhang, Tao Xu, Hongsheng Li, et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: arXiv preprint arXiv:1612.03242 (2016).

References VI

- [21] Bolei Zhou, Aditya Khosla, Agata Lapedriza, et al. "Learning Deep Features for Discriminative Localization". In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, June 2016, pp. 2921–2929. arXiv: 1512.04150.
- [22] Jun-Yan Zhu, Taesung Park, Phillip Isola, et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: CoRR abs/1703.10593 (2017). arXiv: 1703.10593.



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Segmentation and Object Detection

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2020



Outline

Introduction

Segmentation

Motivation

Fully Convolutional Networks for Segmentation

Upsampling

Integrating Context Knowledge

Advanced Topics

Object Detection

Motivation and Background

Region-based Detectors

Single-Shot Detectors



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Introduction



Today's Topics

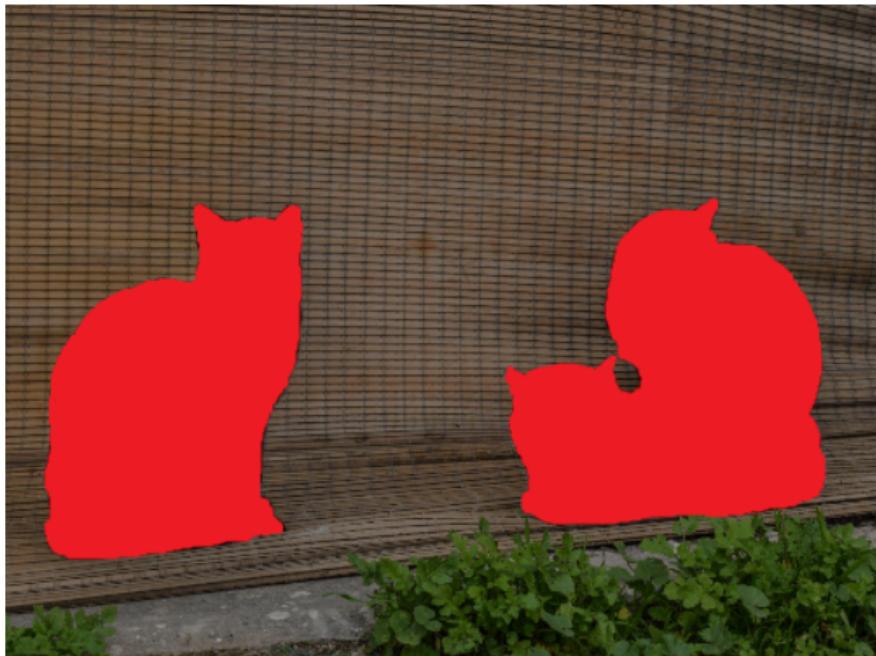


Today's Topics



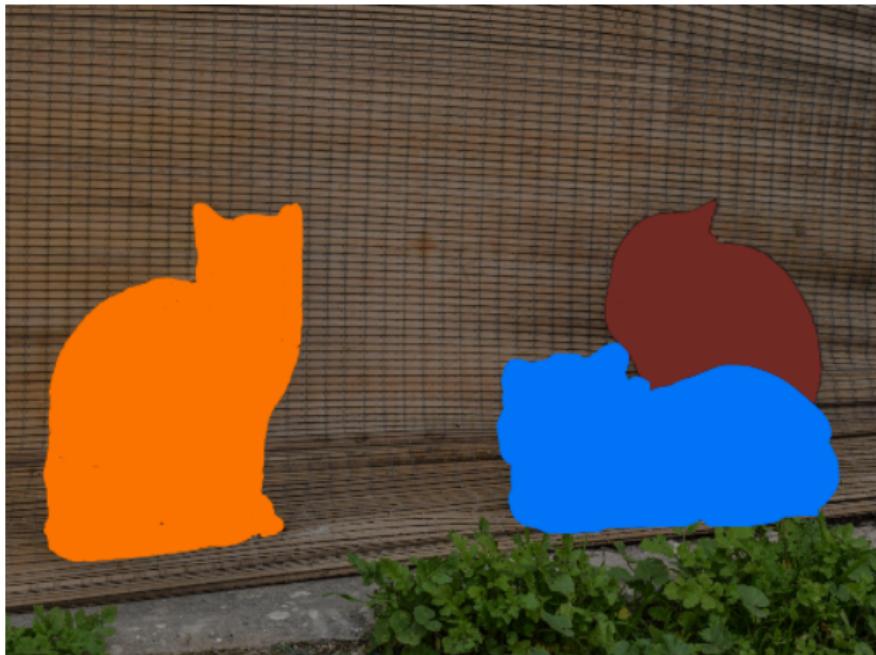
Topic Part II: Object detection = instance recognition + localization

Today's Topics



Topic Part I: Semantic segmentation

Today's Topics



Topic Part III: Semantic instance segmentation



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Segmentation



Motivation

What is Semantic Segmentation?



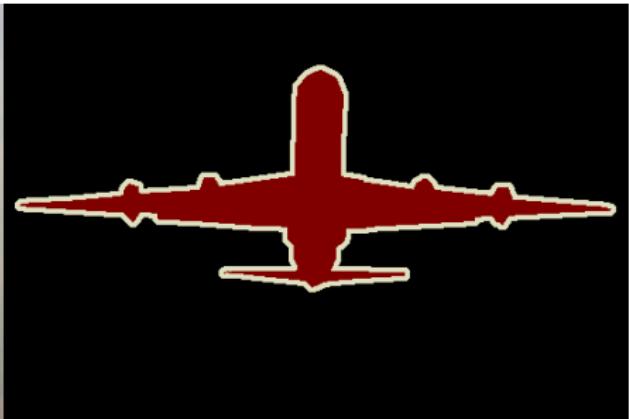
Source: Pascal VOC 2012

- Goal: partition images into **different segments**
- Segments are regions that delineate **meaningful objects**
- Label regions with an **object category label**
- Each pixel gets a semantic class
→ pixel-wise dense classification
- Concepts can be applied to other signals, e.g., sound

What is Semantic Segmentation?



Source: Pascal VOC 2012



Source: Pascal VOC 2012

Applications of Segmentation

- Medical Imaging
 - Organs
 - Vessels
 - Cells
- Autonomous driving
 - Traffic signs
 - Pedestrian detection
 - Street detection
- Aerial imaging, robotics, image editing etc.



Source: Cityscapes dataset

Evaluation Metrics

- Usefulness of segmentation depends on many factors:
Execution time, memory footprint and quality
- Quality of a method can be assessed by different metrics
- Main problem: Classes are not equally distributed → have to account for that

Assumptions:

- $k + 1$ classes including void or background
- p_{ij} is the amount of pixels of class i inferred to belong to class j
→ p_{ii} represents the number of true positive

Evaluation Metrics

1. **Pixel Accuracy (PA):** Ratio between the amount of correctly classified pixels and the total number of pixels

$$PA = \frac{\sum_{i=0}^k p_{ii}}{\sum_{i=0}^k \sum_{j=0}^k p_{ij}}$$

2. **Mean Pixel Accuracy (MPA):** Average ratio of correctly classified pixels per-class basis

$$PA = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij}}$$

Evaluation Metrics

3. **Mean Intersection over Union (MIoU):** Ratio ratio between the intersection and the union of two sets

$$PA = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij} + \sum_{j=0}^k p_{ji} - p_{ii}}$$

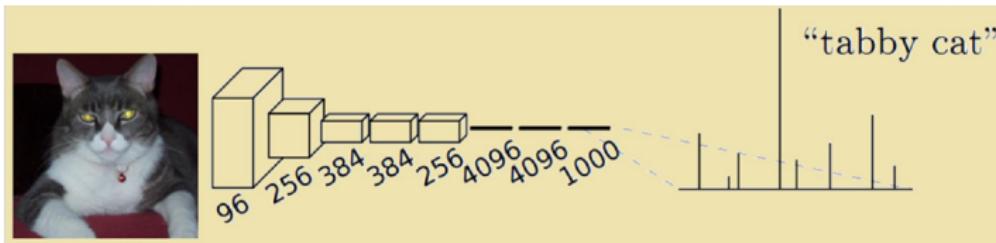
4. **Frequency Weighted Intersection over Union (FWIoU):** balanced MIoU – weights each class depending on its frequency

$$PA = \frac{1}{\sum_{i=0}^k \sum_{j=0}^k p_{ij}} \sum_{i=0}^k \frac{\sum_{j=0}^k p_{ij}p_{ii}}{\sum_{j=0}^k p_{ij} + \sum_{j=0}^k p_{ji} - p_{ii}}$$

Fully Convolutional Networks for Segmentation

Reminder: Fully Convolutional Networks

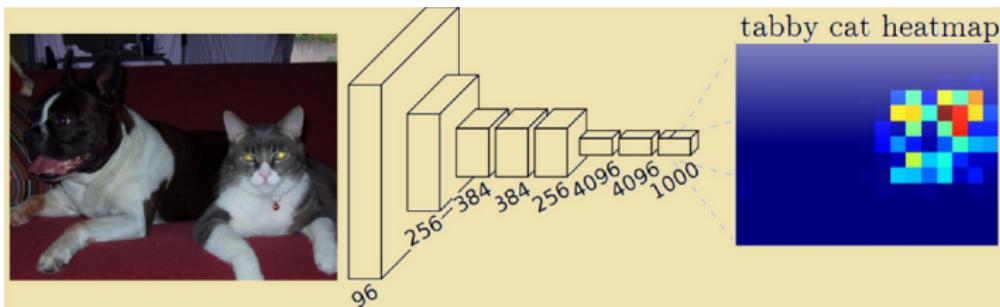
- Convolutional neural networks with **fully connected layers** used for classification:
 - CNN layers for feature extraction
 - Output dimensions reduced because of subsampling
 - Fully connected layers** have fixed inputs and remove spatial information
 - Output: vector encoding class probabilities



Source: Long et al., 2015

Fully Convolutional Networks (cont.)

- Fully convolutional networks for segmentation [13]
 - Transform fully connected layers to **convolutional** layers
 - Output independent of size → heat map
 - Pooling operations coarsen the output of the network

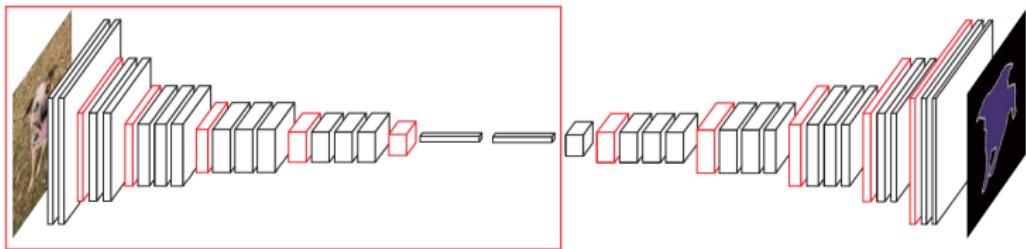


- Main problem: Segmentation is coarse (compared to input resolution)
- Main question: How can we increase the resolution and keep global context?

Source: Long et al., 2015

Encoder and Decoder

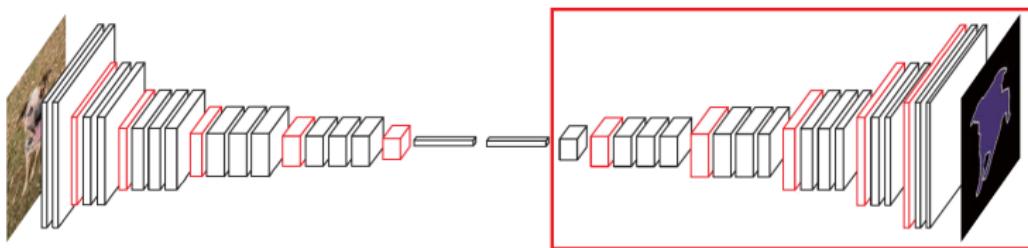
- Most methods use a classification network as basis
- If it can pickup the class, it should be able to learn the correct features
- **Encoder** part of the network



Source: modified from: Long et al. 2015

Encoder and Decoder (cont.)

- Network has to learn to **decode** (map) the output of the encoder to pixel-wise predictions
- **Decoder** part of the network



Example networks with different decoders:

- Long et al.'s Fully Convolutional Networks [13]
- SegNet [1]
- U-net [21]

Source: modified from: Long et al. 2015

Upsampling

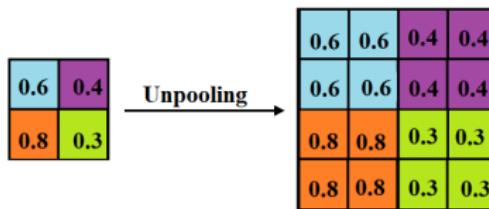
Upsampling Techniques

- Decoder requires upsampling to enable pixel-wise prediction
- Different options possible:
- Unpooling
- Transpose convolution

Upsampling Techniques (cont.)

Nearest neighbor unpooling

Duplicate the value of the element for the region



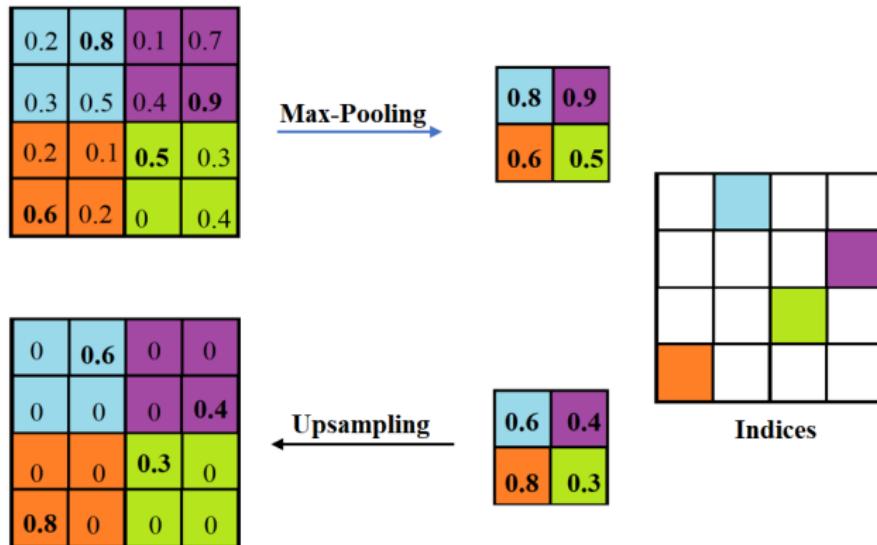
Bed of Nails

Take the one value and make the others zero



Upsampling Techniques

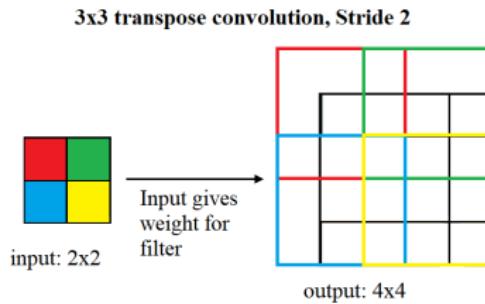
Using max pooling indices



Source: Modified from Semantic Segmentation using FCN over the years

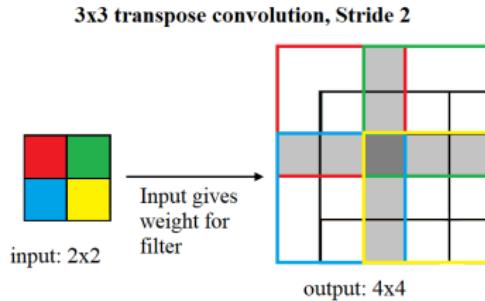
Upsampling Techniques: Transpose convolution

- Learnable upsampling (sometimes misleadingly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride
- Transpose convolution is backward pass of normal convolution
- Stride results in upsampling ("fractionally strided convolution")



Upsampling Techniques: Checkerboard Artifacts

- Transpose convolution has uneven overlap when the kernel size is not divisible by the stride
- The uneven overlaps on the two axes multiply, creating a characteristic checkerboard artifact
- In principle, network could learn weights to avoid this but ...
- ... in practice, networks struggle to avoid it completely



Checkerboard Artifacts

How to avoid

- Choose appropriate kernel size: use a kernel size that is divisible by the stride, avoiding the overlap issue
- Separate upsampling from convolution to compute features
- For example:
 - Resize the image (e.g., using NN or bilinear interpolation)
 - Add a convolutional layer

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Segmentation and Object Detection - Part 2

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2020



Integrating Context Knowledge

Integrating Context Knowledge: Combining Where and What

- Semantic segmentation requires integration of information from various spatial scales
 - Need to balance local and global information
- Local information crucial to achieve good pixel-level accuracy
- Global context of the image enables to resolve local ambiguities
- CNNs struggle with this balance
 - Different approaches to integrate local & global information

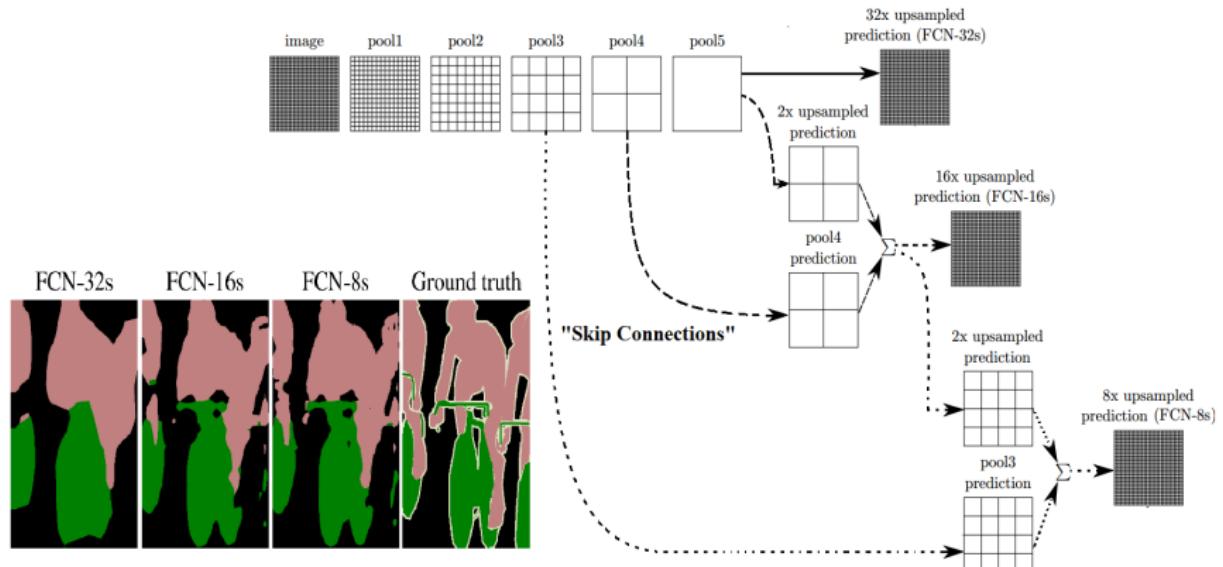
Integrating Context Knowledge (cont.)

Long et al.'s Fully Convolutional Networks [13]

- Upsampling using learnable transposed convolutions
- Idea: Add links combining the final prediction and previous (lower) layers with finer strides
- Additional 1×1 convolution after pooling layer, predictions are added up
 - Make local predictions with global structure
- Network topology is a directed acyclic graph (DAG), with “**skip connections**” from lower to higher layers
- Refinement of coarse segmentation

Integrating Context Knowledge (cont.)

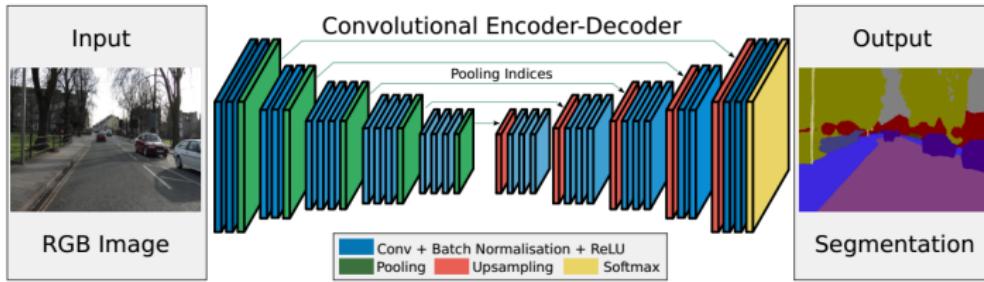
Long et al.'s Fully Convolutional Networks [13]



Source: Long et al. 2015

Integrating Context Knowledge (cont.)

SegNet [1]:



- Decoder: Upsampling & convolution layers followed by softmax
- Each upsampling layer corresponds to a **max-pooling layer** in encoder
- Upsampling reuses max-pooling indices from feature maps in encoder
- provides context information

Source: Badrinarayanan et al. 2015

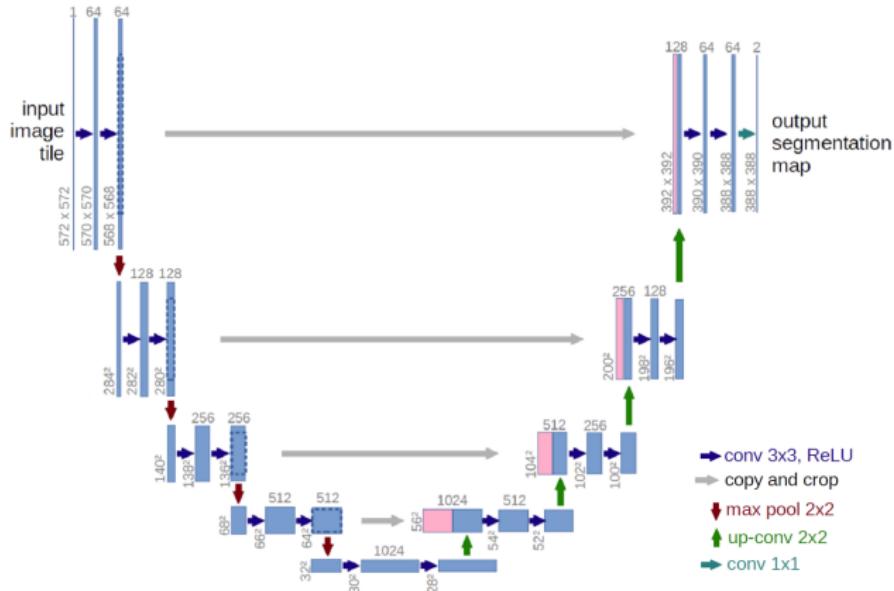
Integrating Context Knowledge (cont.)

U-Net [21]

- The network consists of:
 - Encoder: A contracting path to capture context
 - Decoder: A **symmetric** expansion path for localization (decoder)
- Encoder follows typical architecture of a CNN
- Decoder (expansion path) consists of
 - **Upsampling of feature maps** followed by a 2×2 convolution that halves the number of feature channels.
 - **Concatenation** with the corresponding cropped feature map from the contracting path
- The training strategy relies on use of data augmentation
 - Non-rigid deformation
 - Rotation & translation

Integrating Context Knowledge (cont.)

U-Net [21]



Source: Ronneberger et al. 2015

Additional Approaches

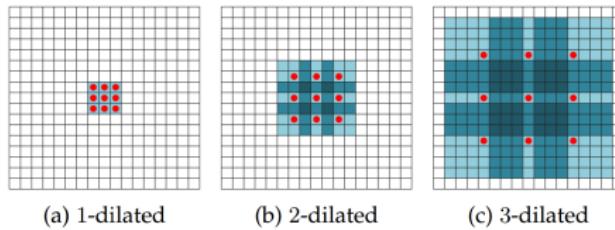
So far: Combination of feature maps from different levels + upsampling

Alternatives & extensions:

- Dilated convolutions
- Network stacks
- Multi-scale networks
- Defer context modeling to another network, e.g. an RNN
- Refinement as a post-processing step with Conditional Random Fields (CRFs)

Additional Approaches: Dilated convolutions

- Also named a-trous convolutions
- Support exponentially expanding receptive fields without losing resolution
- The dilation rate L controls the upsampling factor
- Stacking L -dilated convolutions makes the receptive fields grow exponentially while the number of parameters for the filters grows linear



Source: Fisher and Vladlen, 2015

Additional Approaches: Dilated convolutions (cont.)

Examples

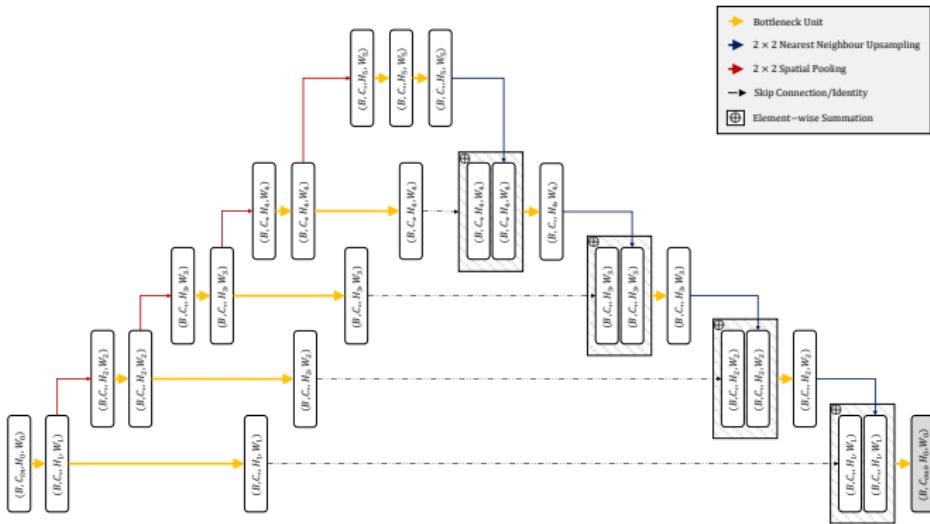
- DeepLab (improved version) [4]
- ENet [17]
- Multi scale context aggregation module [28]

Main issue: No efficient implementation available, benefit somewhat unclear

Additional Approaches: Stacked Hourglass Networks [30]

Hourglass Network vs U-Net:

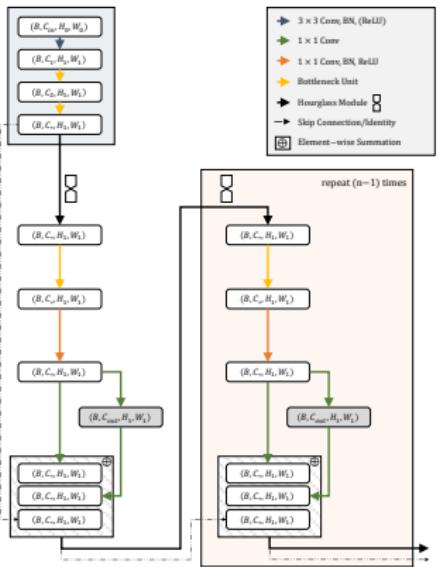
- Additional bottleneck layers
- Learned skip connections
- Constant feature dimension



Additional Approaches: Stacked Hourglass Networks [30] (cont.)

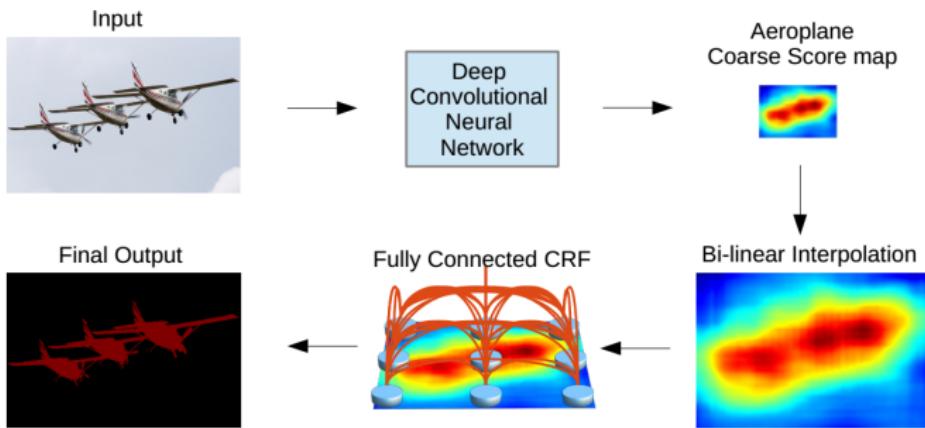
- Stacked Hourglass Network:

Refinement cascade of the prediction with losses after each hourglass



Additional Approaches: Conditional Random Fields

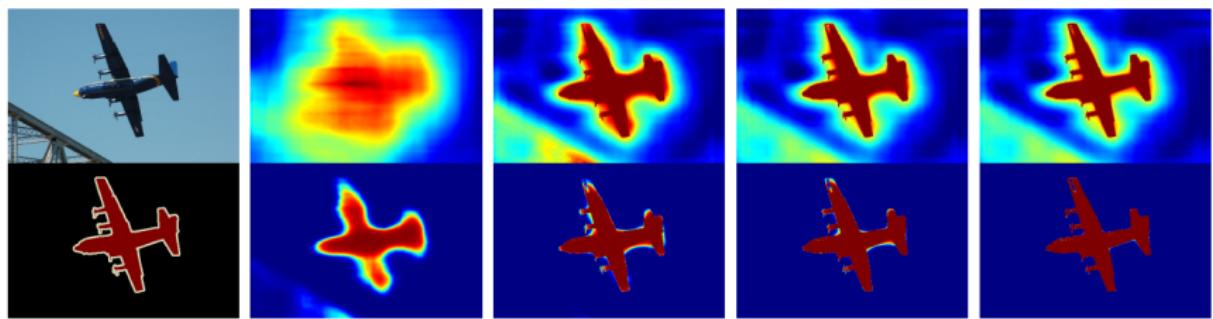
- Idea: Refine coarse CNN output using Conditional Random Field (CRF)
- Each pixel is modeled as a node in the random field, pairwise term between pixels
- Can capture long-range dependencies and fine local information



Source: Chen et al., 2014

Additional Approaches: Conditional Random Fields

- Example: DeepLab [3]
- Iterative CRF refinement of segmentation
- Improvement with extensions (e.g., atrous convolutions) in [4]
- Also possible: Modeling CRF with RNNs [29] → end-to-end training



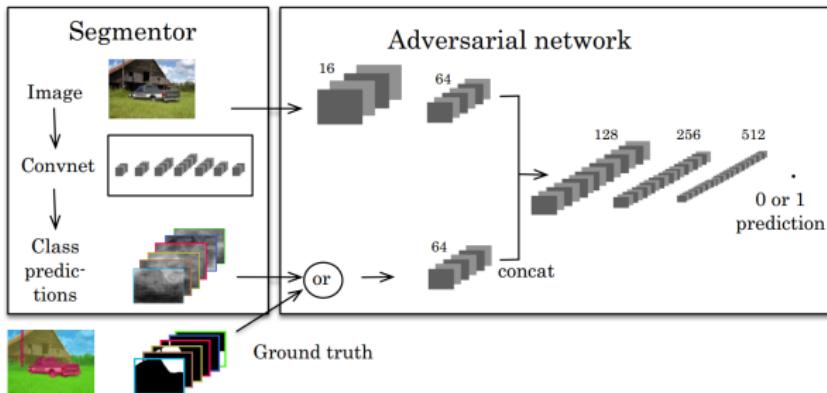
(a) GT (b) CNNout (c) CRFit1 (d) CRFit2 (e) CRFit10

First row: inputs before softmax function, second row: output after softmax function.

Source: Chen et al., 2014

Advanced Topics

Adversarial Networks for Segmentation [14]



Source: Luc et al., 2016

Optimize Segmentor with hybrid loss function with two terms:

- 1.) Usual pixel-wise multi-class cross-entropy for semantic segmentation
- 2.) Loss based on min-max game with Discriminator

Adversarial Networks for Segmentation [14] (cont.)

- Given a data set of N training images \mathbf{x}_n and a corresponding label maps \mathbf{y}_n the loss function defined as:

$$l(\theta_s, \theta_a) = \sum_{n=1}^N l_{mce}(s(\mathbf{x}_n), \mathbf{y}_n) - \lambda [l_{bce}(a(\mathbf{x}_n, \mathbf{y}_n), 1) + l_{bce}(a(\mathbf{x}_n, s(\mathbf{x}_n)), 0)]$$

- $(\mathbf{x}_n, \mathbf{y}_n)$, $n \in \{1, \dots, N\}$: Data set with ground truth labels
- θ_s : Parameters of the Segmentor $s(\mathbf{x}, \mathbf{y})$
- θ_a : Parameters of the Discriminator $a(\mathbf{x}, \mathbf{y})$
- Segmentor is trained both on the segmentation and on fooling the discriminator
- **Multi-task learning** with adversarial task

Adversarial Networks for Segmentation [14] (cont.)

- Given a data set of N training images \mathbf{x}_n and a corresponding label maps \mathbf{y}_n the loss function defined as:

$$I(\theta_s, \theta_a) = \sum_{n=1}^N \underbrace{l_{mce}(s(\mathbf{x}_n), \mathbf{y}_n)}_{\text{multi-class cross-entropy}} - \lambda \underbrace{[l_{bce}(a(\mathbf{x}_n, \mathbf{y}_n), 1) + l_{bce}(a(\mathbf{x}_n, s(\mathbf{x}_n)), 0)]}_{\text{adversarial loss} \rightarrow \text{binary classification}}$$

- $(\mathbf{x}_n, \mathbf{y}_n)$, $n \in \{1, \dots, N\}$: Data set with ground truth labels
- θ_s : Parameters of the Segmentor $s(\mathbf{x}, \mathbf{y})$
- θ_a : Parameters of the Discriminator $a(\mathbf{x}, \mathbf{y})$
- Segmentor is trained both on the segmentation and on fooling the discriminator
- **Multi-task learning** with adversarial task

NEXT TIME
ON DEEP LEARNING



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Segmentation and Object Detection - Part 3

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2020





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Object Detection



Motivation and Background

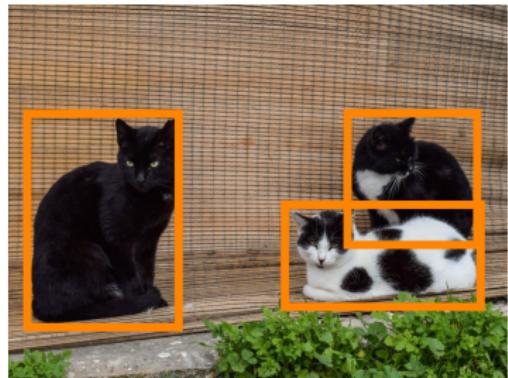
Object Detection – Goal

Two tasks:

- Object location
- Classification

Commonly solved by:

1. Hypothesize bounding boxes
2. Re-sample selected boxes
3. Apply classifier

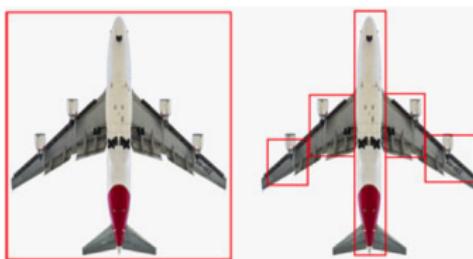


Main distinction: clever combination/replacement of these steps to achieve higher speed or accuracy

What are we looking for?

Bounding Box

- Smallest box by some measure that fully contains the object in question
- Typically defined by: top left corner (x, y) , width w , height h
 - + classifier confidence



Source: <https://github.com/rkerian/Launch-Academy/blob/master/bounding-box/bounding-box.md>

Early approaches

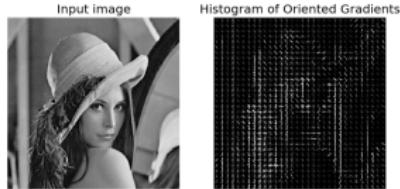
Viola & Jones '01

- Mainly used for face detection
- Haar Features + Adaboost + cascading classifier for fast rejection
- First competitive real-time object detection



Dalal & Triggs '05

- Histogram of Oriented Gradients (HOG)
- Support Vector Machines



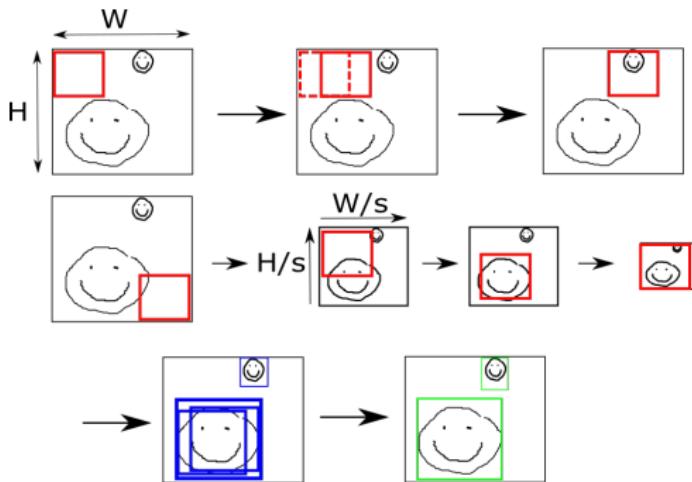
Source: <https://en.wikipedia.org> , www.sharky93.github.io/docs/gallery/

Modern Approaches

Based on neural networks

- **Sliding window:** classify each possible window by a CNN
- **Region proposal CNN (R-CNN):** find interesting image regions first, then classify by CNN
- **Single-Shot Detectors:** joint detection and classification
 - You Only Look Once (YOLO)
 - Single-Shot MultiBox Detector (SSD)

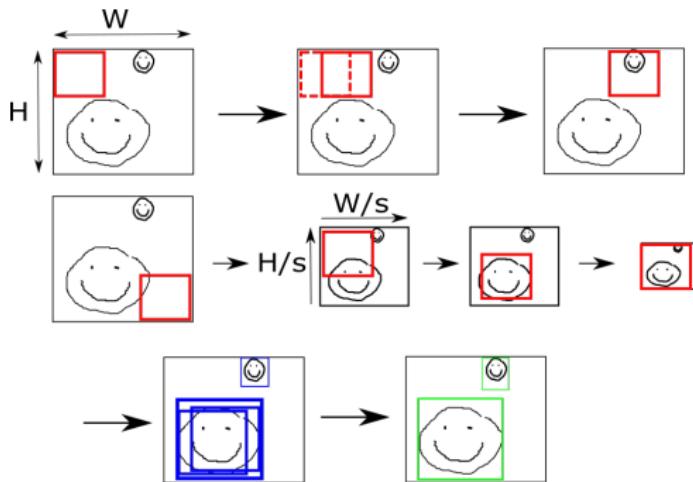
Sliding Window Approach



- Slide fixed size window over image + apply trained CNN to each window
- Use multiple image scales to detect objects of different sizes
 - Large number of predictions, but many with low confidence
 - Keep only high confidence areas

Source: <https://www.semanticscholar.org/>

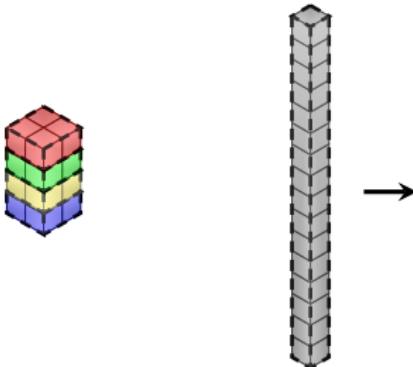
Sliding Window Approach



- + Trained **classification network** can be used for object detection directly
- Computationally inefficient: One pass through the network for every patch!
- Improve speed by using Fully Convolutional Networks

Source: <https://www.semanticscholar.org/>

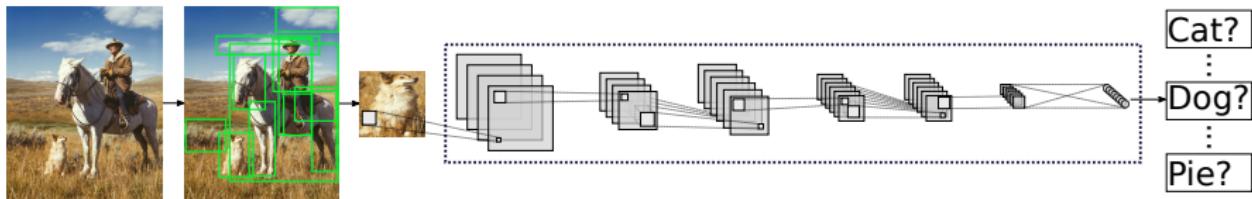
Reminder: Fully Convolutional Networks



- How can we apply a **fully connected layer** to an arbitrary shaped tensor?
- We can **flatten** the activations
- but we can also **reshape the weights** and **convolve**
- This produces the **exact same result**
- but can be calculated on **arbitrary spatial input sizes**
- Results can be **aggregated** using pooling

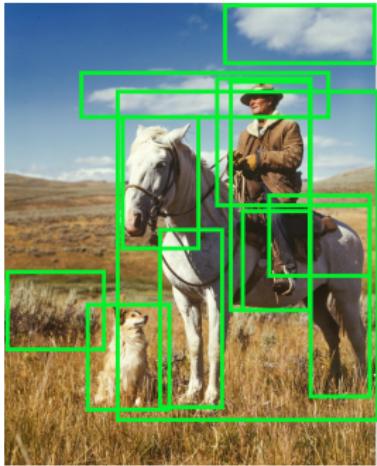
Region-based Detectors

Regional CNN (R-CNN) [20]



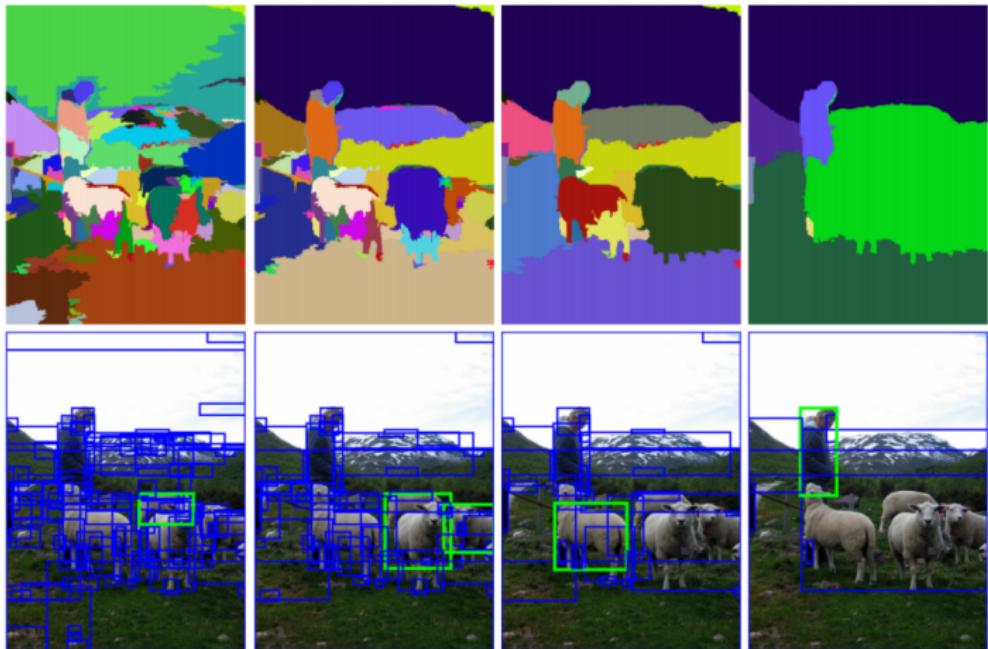
- CNN are powerful classifiers
- FCNs help to improve efficiency - but still **still brute-force**
- Also: Different scales and shapes
- Can we **improve efficiency** by only **considering interesting regions** (ROIs)?
- Multi-step approach in R-CNN [20]:
 - Generate **region proposals**: Selective Search
 - Classify content of region proposals + refine bounding box

Region Proposal: Selective Search [23]



- Candidate objects by grouping pixels of similar texture or color
- Apply for different sized windows
- Produce few thousand ($\approx 2k$) object proposals per image \ll number possible windows
- Essentially a form of segmentation

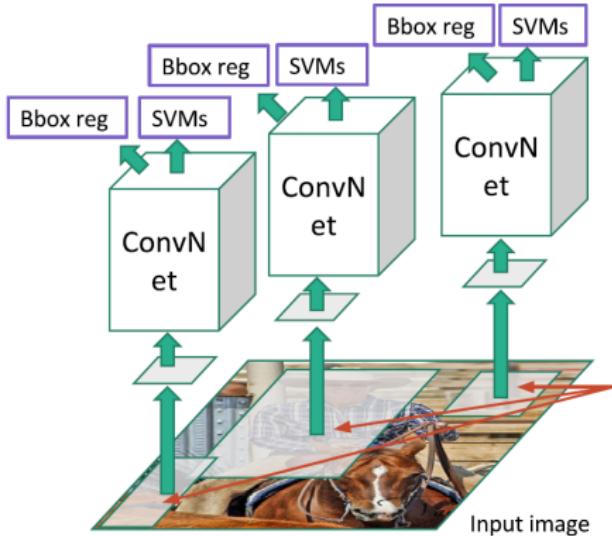
Region Proposal: Selective Search (cont.)



Source: [23]

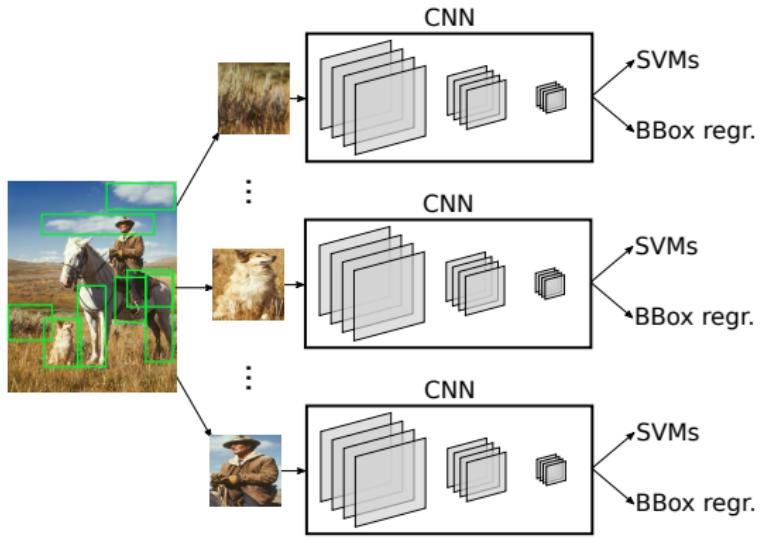
Regional CNN (R-CNN) [20]

- For each region proposal window
 - Warp to standard window size
 - Pre-trained CNN for feature extraction
 - Linear SVM for object classification
 - Linear regression for bounding box refinement
- + Improved retrieval rate at that time (2013) by more than 30%
- Much faster... but still slow
- Not end-to-end



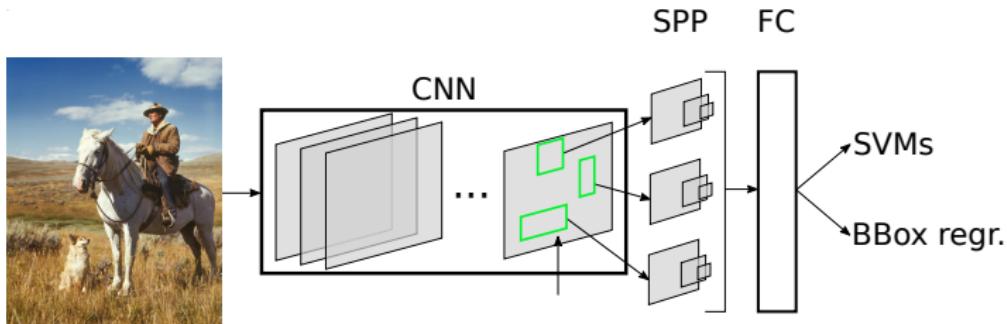
Source: Figure copyright Ross Girshick, 2015, <https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0>

Towards Fast R-CNN



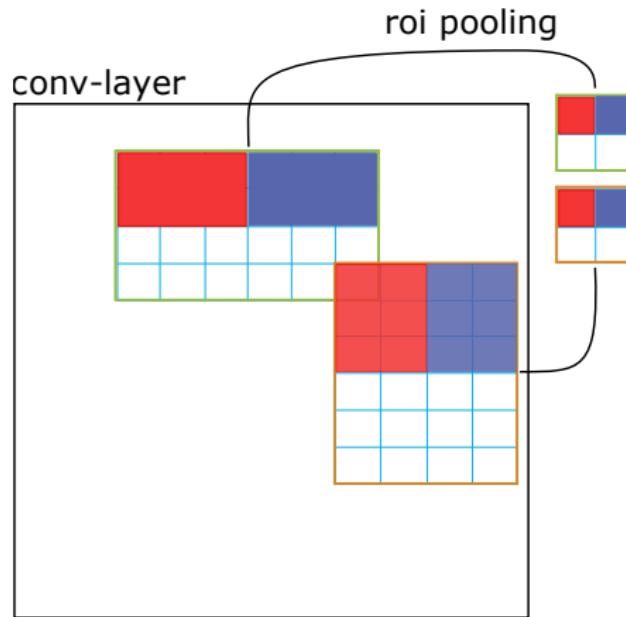
- R-CNN requires full forward pass for **each** region proposal
 - Training is slow
 - Inference is slow
- We can share computations when computing feature maps
- Why not use region proposals on feature maps?

Towards Fast R-CNN: Spatial Pyramid Pooling [22]

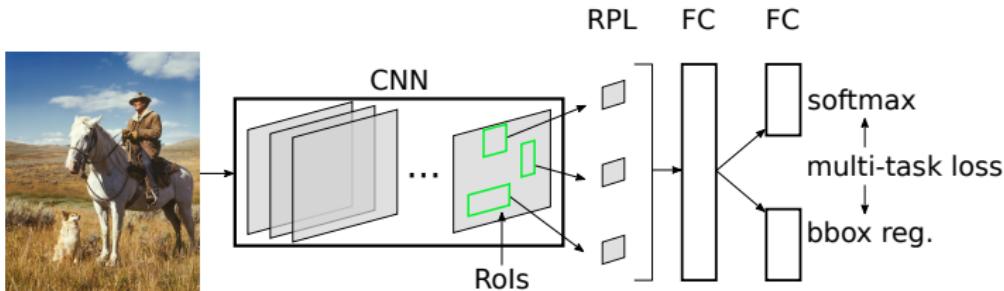


- Pass full image through the network: Feature maps
- Apply region proposals to **last conv layer**
- Classification CNN has **fixed input size**
- **Spatial pyramid pooling** layer pools to fixed size using max-pooling (orig. 3x w. different window size & stride)
- + Image-wise computation shared → Speed up by SPP during inference: $\approx 24\text{--}104 \times$
- R-CNN problems: **slow training** / not end-to-end

Towards Fast-RCNN: Region Proposal Pooling

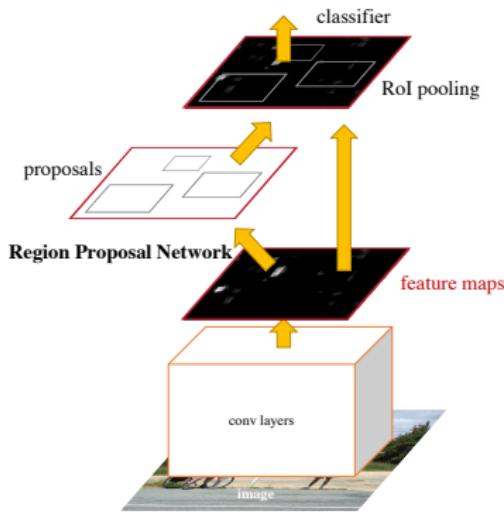


Fast R-CNN [6]



- Uses SPP layer with one output map
- Better sample selection for mini-batches
 - Sample 128 ROIs uniformly random → feature maps don't overlap
 - Instead: **Hierarchical sampling:**
Sample from **few** images, but many ROIs (64) → high overlap
- Replace SVM + regression by: softmax layer for classification + regression layer for bounding box finetuning → trained by multi-task loss
- + 9× faster than R-CNN
- Still not real-time
- / Almost end-to-end **apart** from ROI proposals on conv layer

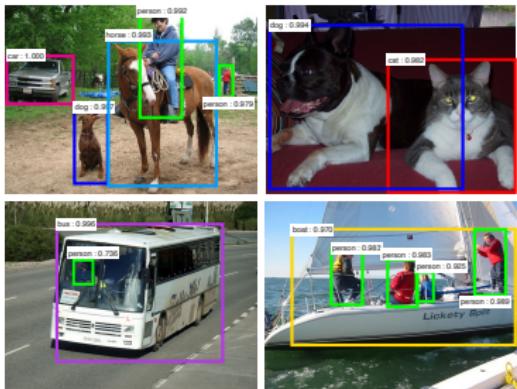
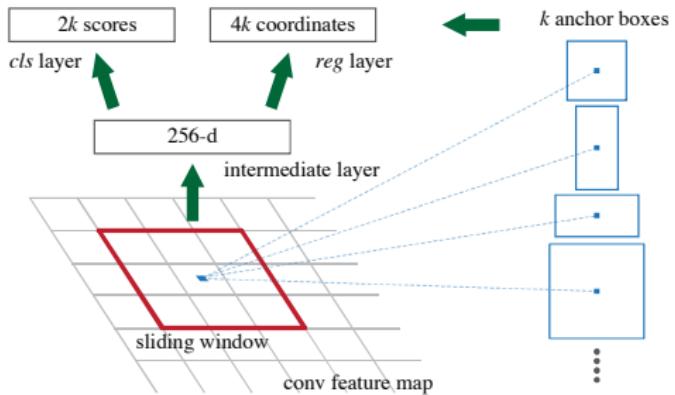
Faster R-CNN [5]



- Key idea: Add region proposal network
- Input: generated feature maps
- Output: region proposals
- Integrate into “Fast R-CNN”

Source: [5]

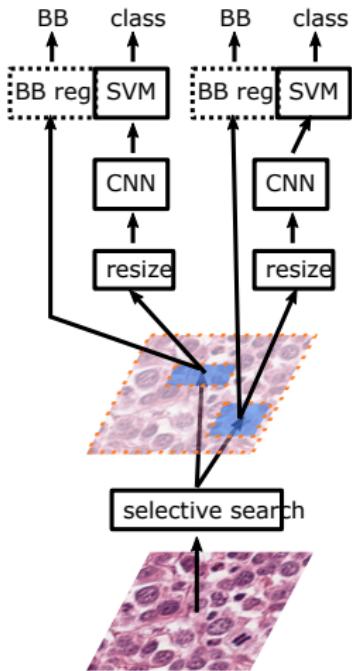
Faster R-CNN [5]



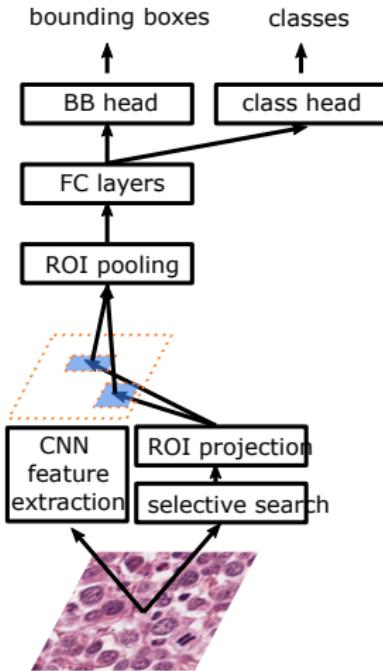
- Defines k anchor boxes associated with different scale and aspect ratios (typically: 3 scales / 3 aspect ratios $\rightarrow k = 9$)
- Efficient multi-scale ability (neither image nor filter sizes need to change)
- $4k$ box parameters + $2k$ scores (softmax for object/not object)
- + End-to-end system
- Not real-time (5-17 fps)

Source: [5]

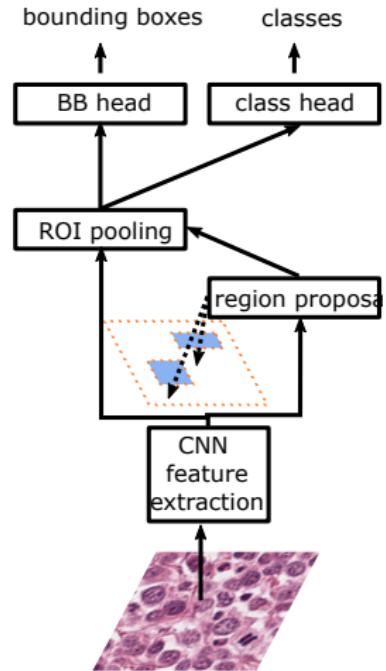
Overview: Architectures based on R-CNN



R-CNN



Fast R-CNN



Faster R-CNN

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Segmentation and Object Detection - Part 4

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2020

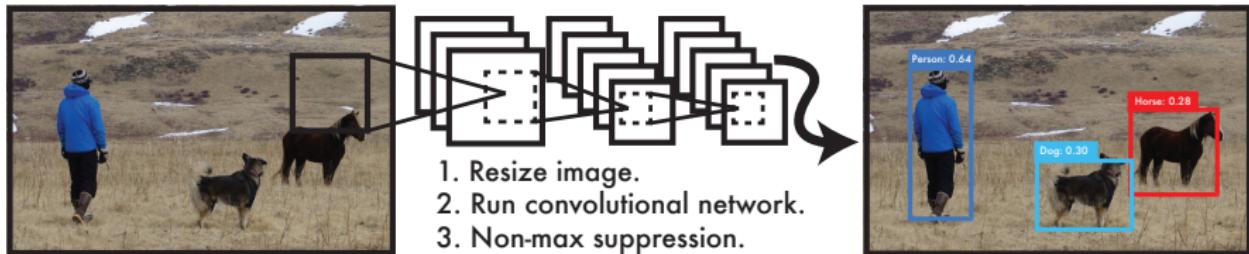


Single-Shot Detectors

Can't we just use the region proposal network as detector?

In a YOLO fashion?

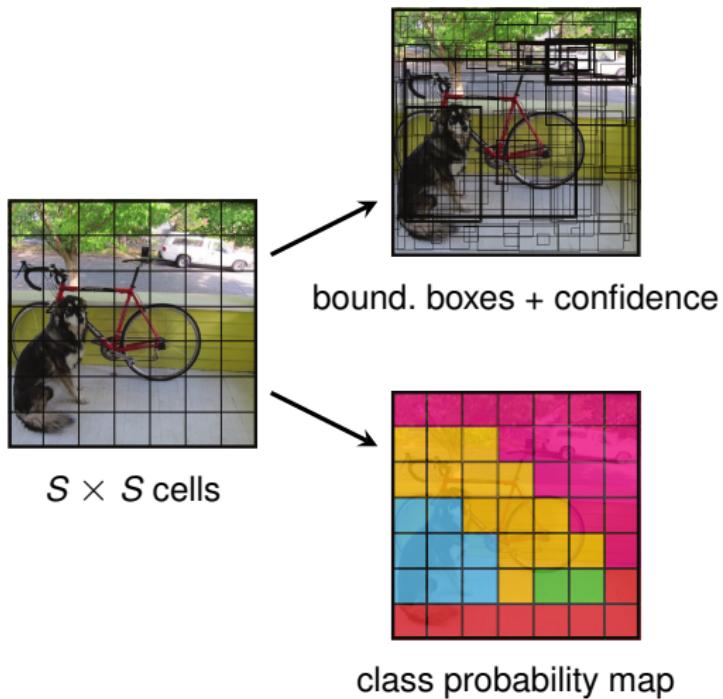
YOLO [26]



- R-CNN / Fast R-CNN produces many object candidate suggestions that are passed to the CNN
 - Slow and cumbersome
 - Single-shot detectors replace the many forward-passes by only one
- **You Only Look Once (YOLO)** algorithm
- Combined bounding box prediction and classification in one network

Source: [26]

YOLO [26]

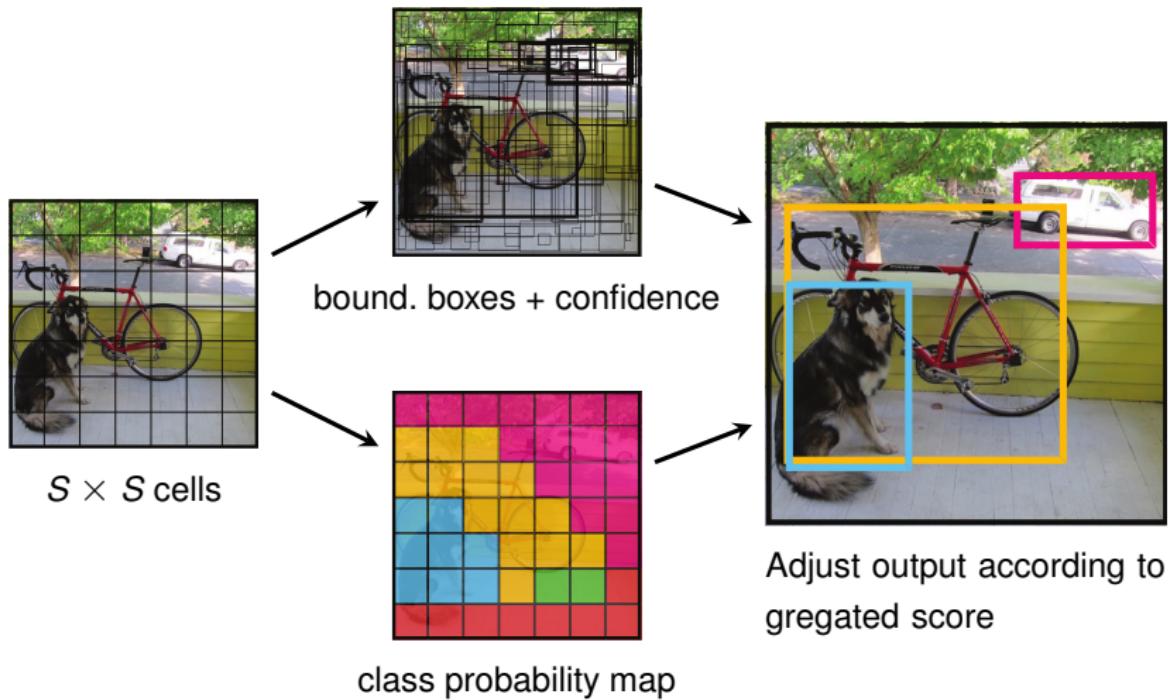


- Each cell predicts
 - B bounding boxes + confidence score
 - Class confidence
- CNN predicts:

$$S \times S \times (5B + C)$$
values for C classes

Source: [26]

YOLO [26]



Source: [26]

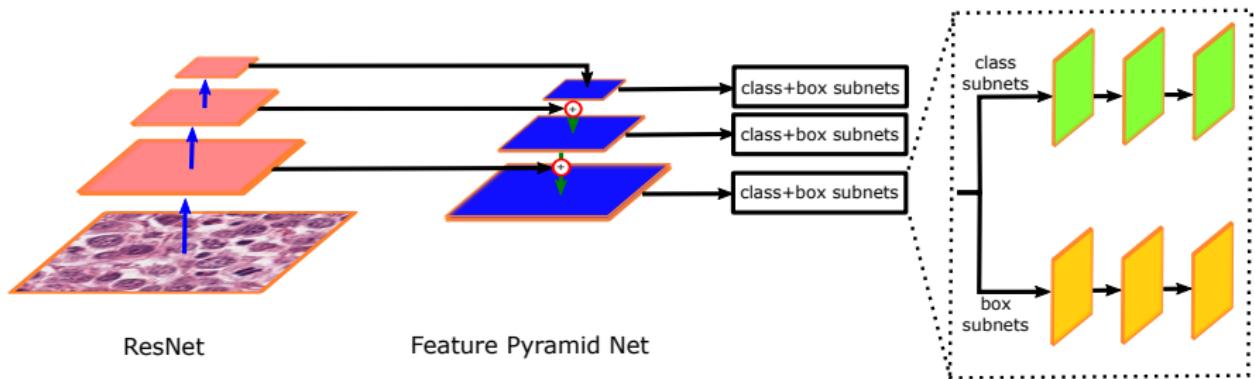
YOLO9000 [27]

- YOLO9000 is an improved version of YOLO advertised as Better, Faster, Stronger
- Better
 - Batch normalization and use of a high-res classifier improve mAP by up to 6%
 - Anchor boxes found by clustering over the training data improves recall by 7%
 - Training over multiple scales allows YOLO9000 to detect objects at different resolutions more easily
- Faster
 - Using a different CNN architecture speeds up the forward pass
- Stronger
 - Hierarchical prediction on a tree allows to combine different object detection datasets
- All this allows YOLO9000 to detect up to 9000 classes in real-time or faster

The Single-Shot Multi-Box Detector[24]

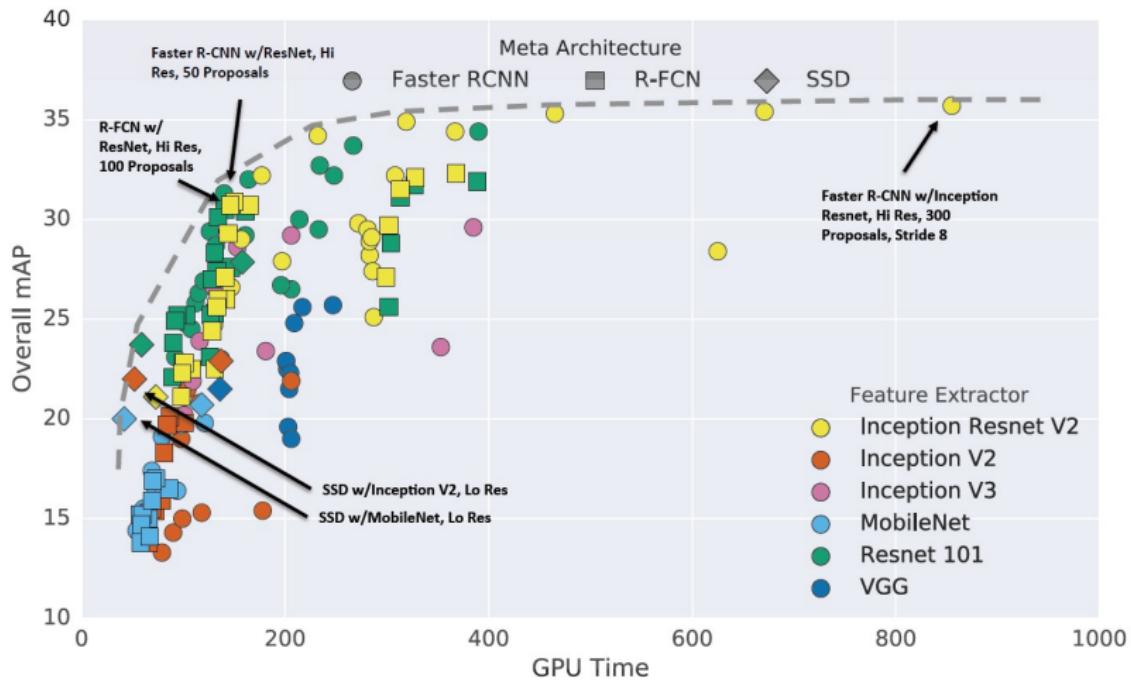
- A popular alternative to YOLO:
 - **Single-Shot:** Like YOLO, only one forward pass through the CNN
 - **MultiBox:** Name of the bounding box regression technique which SSD is based on [15]
 - **Detector:** It's (obviously) an object detector
- Differs from YOLO in several aspects but shares the same core idea

RetinaNet Detector [7]



- Stages of Feature Pyramid Network (FPN) represent different scalings and are used for BBox Regression + Classification
- Shows similarities to U-Net
 - Encoding - ResNet as feature extractor
 - Decoding - FPN trained multiscaling
 - Skip-Connections - Residual connections
- Uses Focal Loss [7]

Tradeoff: Speed-Accuracy

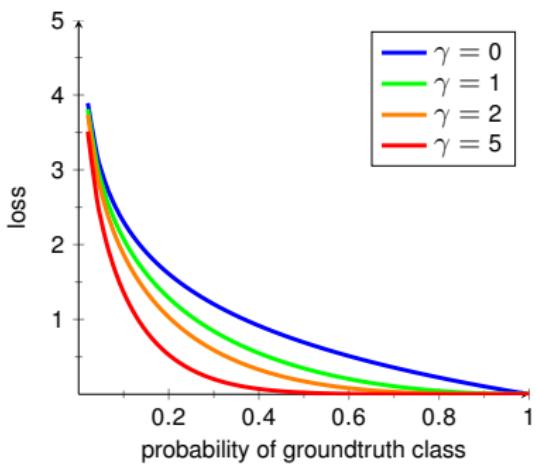


Source: Speed/accuracy trade-offs for modern convolutional object detectors [12]

Class Imbalance to Tackle the Speed-Accuracy Tradeoff

- All single shot detectors evaluate **many hypothesis locations**
 - but **most** of them are **easy negatives**
- This **imbalance** is **not addressed** by current training
- In classical methods this is dealt with by **hard-negative mining**
- Can we **change the loss functions** to pay **less attention to easy examples**?

Focal Loss[7]



- Objectness is binary
 - Model as Bernoulli distributed:
$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$
 - The usual loss function is cross-entropy: $\text{CE}(p_t) = -\log(p_t)$
 - We modify this to:
- $$\text{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \cdot \log(p_t)$$
- α_t are **imbalance weights** calculated as inverse class frequency
 - γ is a hyperparameter **decreasing** the **influence of easy examples**

Summary: Object detection

- Main tasks: Bounding boxes + classification
- Sliding window approach extremely inefficient
- Region proposal networks reduce number of candidates
- Single-shot detectors (YOLO, SSD) avoid additional step
- Object detector concepts can be combined with arbitrary feature extraction/classification network
- Speed/accuracy tradeoff!

**NEXT TIME
ON DEEP LEARNING**



FAU

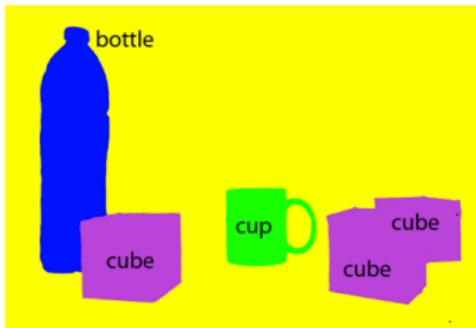
FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Segmentation and Object Detection - Part 5

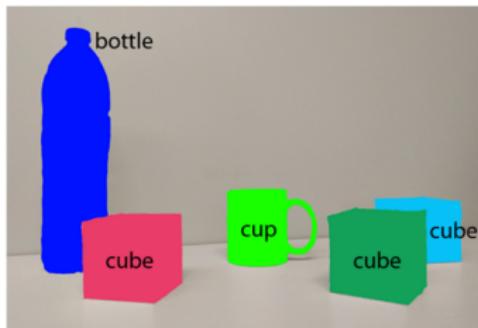
**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2020



Instance Segmentation



Semantic segmentation



Instance segmentation

Source: Garcia et al. 2017

- Next step after semantic segmentation
- Main goal: detect **different instances** of the same class
- Number of instances initially unknown, pixel-wise prediction as in semantic segmentation not sufficient
- Combination of **object detection** and **semantic segmentation**

Instance Segmentation

Examples for potential applications:

- Information about occlusion
- Counting number of elements belonging to the same class
- Detecting object boundaries, e.g., for gripping objects in robotics

Examples in literature:

- Simultaneous Detection and Segmentation (SDS) [9]
- DeepMask [18]
- SharpMask [19]
- Mask R-CNN [10]

Instance Segmentation Example: Mask R-CNN [10]

- Back to the “start”: Combine object detection (R-CNN) with segmentation
- Object detection solves instance separation, segmentation refines bounding boxes per instance

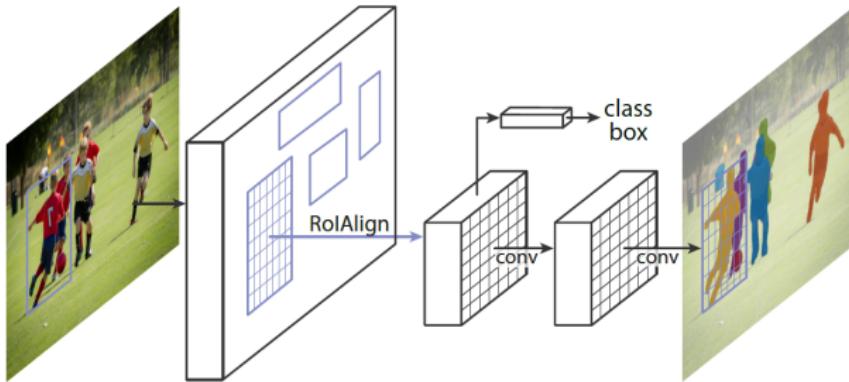


Source: He et al., 2017 [10]

Instance Segmentation Example: Mask R-CNN [10] (cont.)

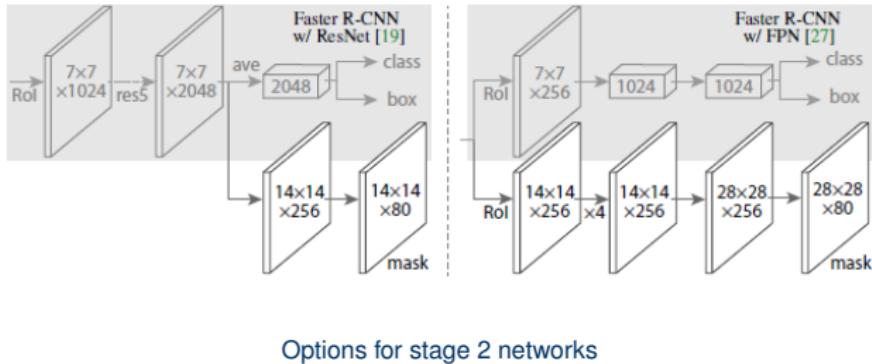
Workflow: Two-stage procedure

1. Region proposal: proposes candidate object bounding boxes
 2. Classification, bounding-box regression **and** segmentation in parallel
- **Multi-task loss:** $L = L_{cls} + L_{box} + L_{mask}$



Source: He et al., 2017 [10]

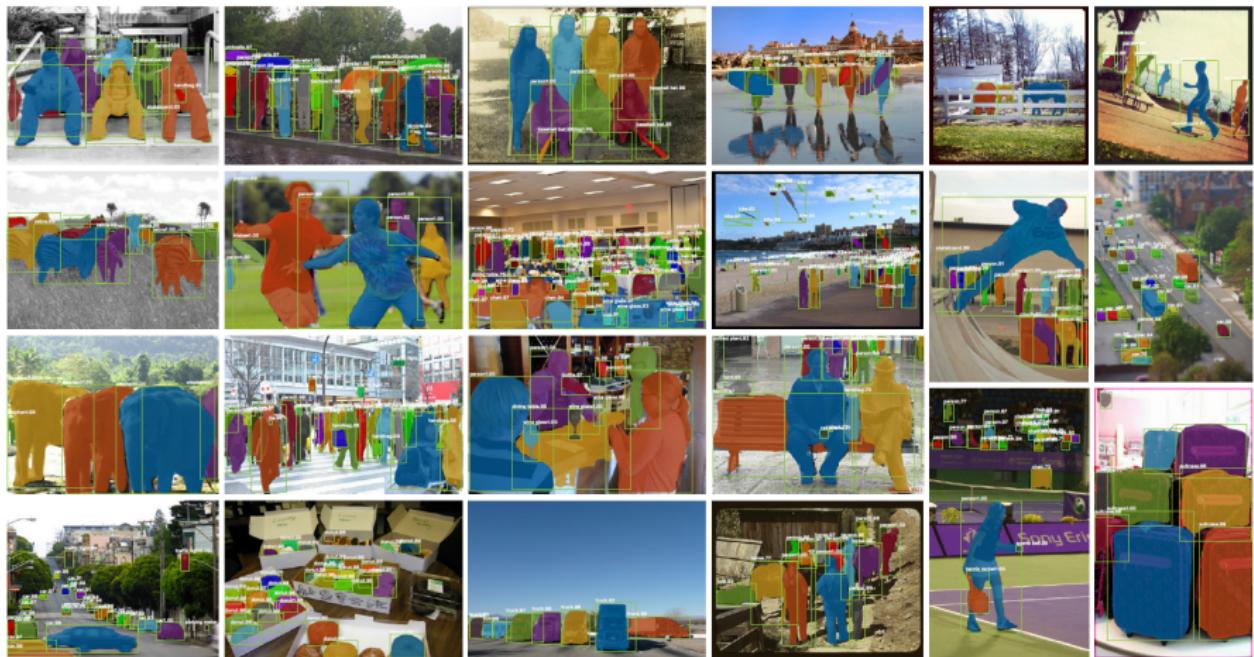
Instance Segmentation Example: Mask R-CNN [10] (cont.)



- Segmentation is independent of classification → only binary loss computed for correct class
- Second-stage networks can be arbitrarily coupled
- **Multi-task loss:** $L = L_{cls} + L_{box} + L_{mask}$

Source: He et al., 2017 [10]

Instance Segmentation Example: Mask R-CNN [10] (cont.)



Examples for instance segmentation

Summary

- Segmentation is commonly solved by architectures analyzing the image and subsequently refining coarse results
- Fully convolutional networks preserve spatial layout and enable arbitrary input sizes combined with pooling
- Object detectors can be implemented as a sequence of region proposals and classification. Examples are the R-CNN family of networks.
- Alternatively one can go all YOLO and perform single-shot object detection with the region proposals networks
- Object detection and segmentation are closely related and combinations are common

**NEXT TIME
ON DEEP LEARNING**

Coming Up

- Methods to relieve the burden of labeling
- Integration of known operators

Comprehensive Questions

- What is the difference between semantic and instance segmentation and what is the connection to object detection?
- How can we construct a network which accepts arbitrary input sizes?
- What is ROI pooling?
- How can we perform backpropagation through a ROI pooling layer?
- What are typical measures for evaluation of segmentations?
- What similarities do typical autoencoders share with typical segmentation networks?
- Explain a method for instance segmentation.

Further Reading

- [Link](#) - Joseph Redmons awesome website including the DarkNet library and material about YOLO
- [Link](#) - Joseph Redmons CV - have a look at it - will definitely jumpstart your career!



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

References



References I

- [1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: [arXiv preprint arXiv:1511.00561](#) (2015). arXiv: 1311 . 2524.
- [2] Xiao Bian, Ser Nam Lim, and Ning Zhou. "Multiscale fully convolutional network with application to industrial inspection". In: [Applications of Computer Vision \(WACV\), 2016 IEEE Winter Conference on](#). IEEE. 2016, pp. 1–8.
- [3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs". In: [CoRR abs/1412.7062](#) (2014). arXiv: 1412 . 7062.

References II

- [4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: [arXiv preprint arXiv:1606.00915](#) (2016).
- [5] S. Ren, K. He, R. Girshick, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: vol. 39. 6. June 2017, pp. 1137–1149.
- [6] R. Girshick. "Fast R-CNN". In: [2015 IEEE International Conference on Computer Vision \(ICCV\)](#). Dec. 2015, pp. 1440–1448.
- [7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, et al. "Focal loss for dense object detection". In: [arXiv preprint arXiv:1708.02002](#) (2017).

References III

- [8] Alberto Garcia-Garcia, Sergio Orts-Escalano, Sergiu Oprea, et al. "A Review on Deep Learning Techniques Applied to Semantic Segmentation". In: [arXiv preprint arXiv:1704.06857](#) (2017).
- [9] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, et al. "Simultaneous detection and segmentation". In: [European Conference on Computer Vision](#). Springer. 2014, pp. 297–312.
- [10] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al. "Mask R-CNN". In: [CoRR abs/1703.06870](#) (2017). arXiv: 1703.06870.
- [11] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: [2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition](#). Vol. 1. June 2005, 886–893 vol. 1.

References IV

- [12] Jonathan Huang, Vivek Rathod, Chen Sun, et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: [CoRR abs/1611.10012](#) (2016). arXiv: 1611.10012.
- [13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: [Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition](#) 2015, pp. 3431–3440.
- [14] Pauline Luc, Camille Couprie, Soumith Chintala, et al. "Semantic segmentation using adversarial networks". In: [arXiv preprint arXiv:1611.08408](#) (2016).
- [15] Christian Szegedy, Scott E. Reed, Dumitru Erhan, et al. "Scalable, High-Quality Object Detection". In: [CoRR abs/1412.1441](#) (2014). arXiv: 1412.1441.

References V

- [16] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation". In: Proceedings of the IEEE International Conference on Computer Vision. 2015, pp. 1520–1528.
- [17] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, et al. "Enet: A deep neural network architecture for real-time semantic segmentation". In: arXiv preprint arXiv:1606.02147 (2016).
- [18] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. "Learning to segment object candidates". In: Advances in Neural Information Processing Systems. 2015, pp. 1990–1998.
- [19] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, et al. "Learning to refine object segments". In: European Conference on Computer Vision. Springer. 2016, pp. 75–91.

References VI

- [20] Ross B. Girshick, Jeff Donahue, Trevor Darrell, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: [CoRR abs/1311.2524](#) (2013). arXiv: 1311.2524.
- [21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: [MICCAI](#). Springer. 2015, pp. 234–241.
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: [Computer Vision – ECCV 2014](#). Cham: Springer International Publishing, 2014, pp. 346–361.
- [23] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, et al. "Selective Search for Object Recognition". In: [International Journal of Computer Vision](#) 104.2 (Sept. 2013), pp. 154–171.

References VII

- [24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. "SSD: Single Shot MultiBox Detector". In: Computer Vision – ECCV 2016. Cham: Springer International Publishing, 2016, pp. 21–37.
- [25] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In:
Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 1. 2001, pp. 511–518.
- [26] J. Redmon, S. Divvala, R. Girshick, et al. "You Only Look Once: Unified, Real-Time Object Detection". In:
2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2016, pp. 779–788.
- [27] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In:
CoRR abs/1612.08242 (2016). arXiv: 1612.08242.

References VIII

- [28] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: [arXiv preprint arXiv:1511.07122](#) (2015).
- [29] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, et al. "Conditional Random Fields as Recurrent Neural Networks". In: [CoRR abs/1502.03240](#) (2015). arXiv: 1502.03240.
- [30] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation". In: [European conference on computer vision](#). Springer. 2016, pp. 483–499.



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Weakly and Self-Supervised Learning

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
July 7, 2020



Outline

Learning with Limited Annotations

Definition

Image-based SSL for Representation Learning

Generative

Spatial Context

Context Similarity

Contrastive SSL

Supervised Contrastive Learning

Bootstrap SSL – A paradigm change



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Learning with Limited Annotations



Supervised Learning

So far, we have seen impressive results, achieved with ...

- ... large amounts of training data and
- ... consistent, high-quality annotations



Mask R-CNN [7], image source [2]

The Cost of Annotation

Image-level class labels: ~27 sec [11]



Instance spotting: + 14 sec [11]



Instance Segmentation: + 80 sec [11]

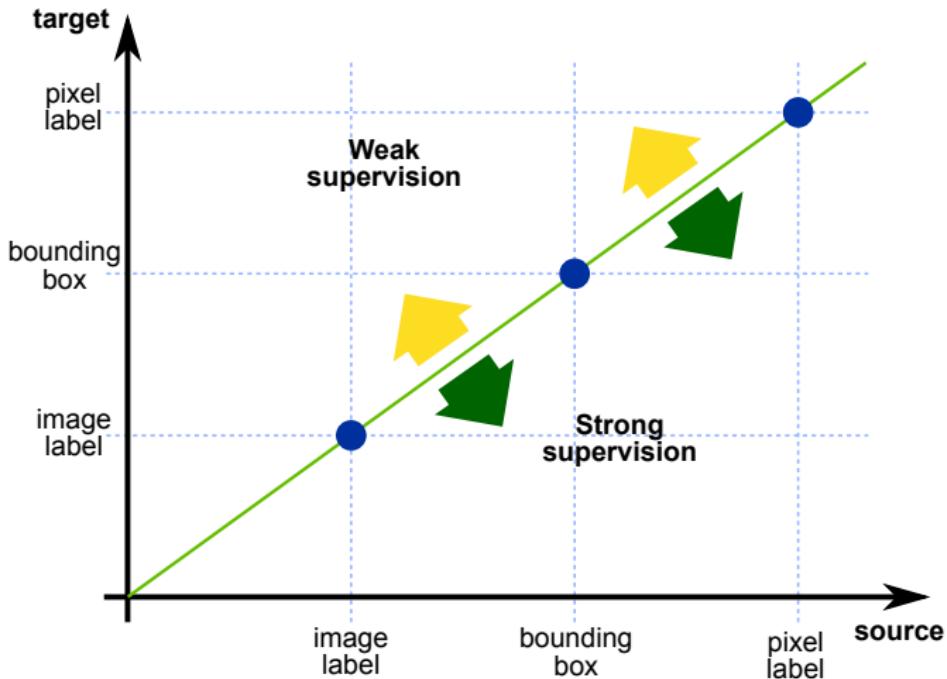


Dense pixel-level annotations: 1.5h [4]



Source: [4], [11]

Strongly vs. Weakly Supervised Learning



Source: Reproduced from CVPR18 Tutorial: Weakly Supervised Learning for Computer Vision

Key Ingredients for Weakly Supervised Learning

Priors: Explicit and Implicit

- Shape + size
- Contrast
- Motion
- Class distribution
- Similarity across images

Hints

- Image labels
- Bounding boxes
- Image caption
- Sparse temporal labels
- Scribbles
- Clicks inside objects

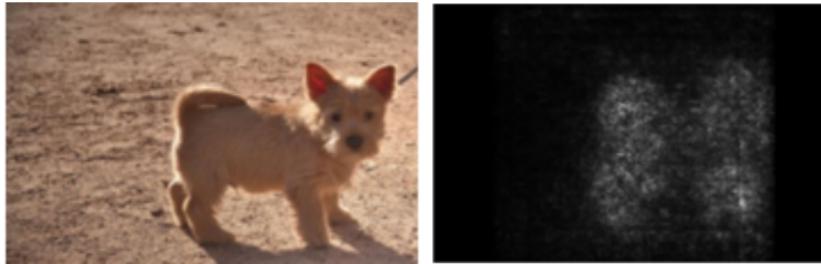


Source: Bearman et al. [3]

From Labels to Localization

Approach 1: Use a **pretrained** classification network [13]

- How does a change in the input affect the classification?
→ Lecture on Visualization
- Qualitative segmentation map
- **Problem 1:** Classifier was never trained for localized decisions
- **Problem 2:** Good classifiers don't automatically yield good maps



Source: Simonyan et al. [13]

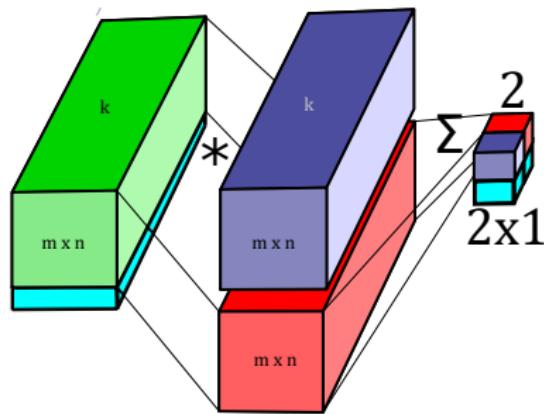
From Labels to Localization

Approach 2: Use a classification network, but smarter [14]

- Core idea: Use **global average pooling**
→ Fully convolutional networks revisited

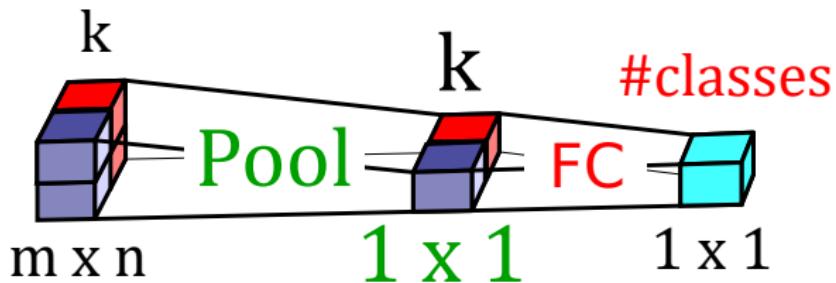
Fully Convolutional Networks: Revisited

- Fully connected layers fix the input size
→ replace by $m \times n$ convolution
- Alternatively, we can also **pool** to the correct size first



Fully Convolutional Networks: Revisited

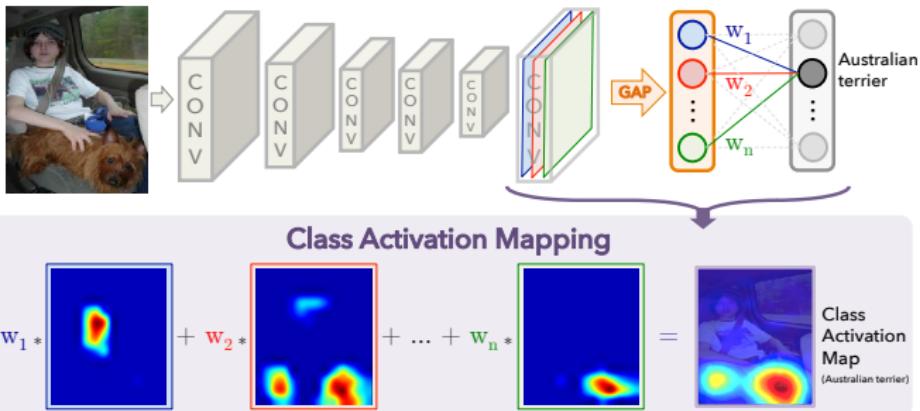
- Fully connected layers fix the input size
→ replace by $m \times n$ convolution
- Alternatively, we can also **pool** to the correct size first



From Labels to Localization

Approach 2: Use a classification network, but smarter [14]

- Core idea: Use **global average pooling**
- Then look at penultimate layer
- **Class Activation Maps (CAMs)**
- Generalization: Grad-CAM [12]



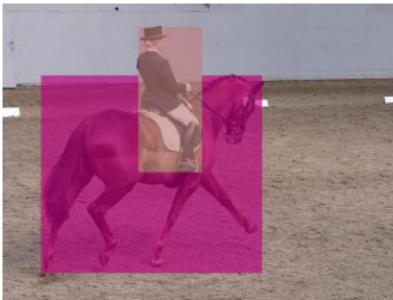
Source: Zhou et al. [14]

From Bounding Boxes to Segmentation

Expensively annotated
Fully supervised



Cheaply annotated



Cheaply annotated
Weakly supervised

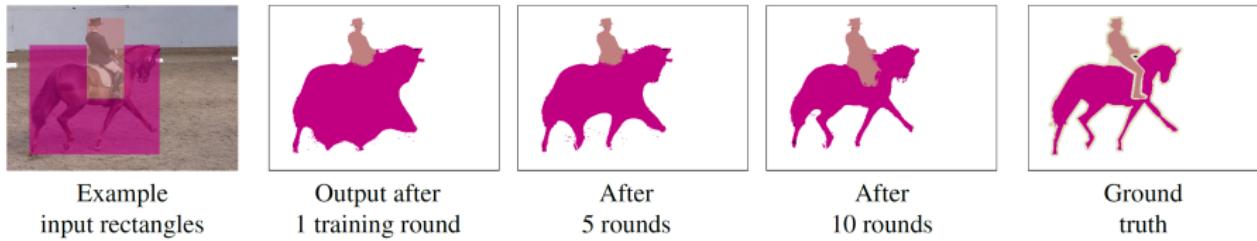


- Manual segmentation is tedious
- Bounding boxes are less tedious
- Can we learn segmentation from boxes?

Source: Khoreva et al. [6]

From Bounding Boxes to Segmentation

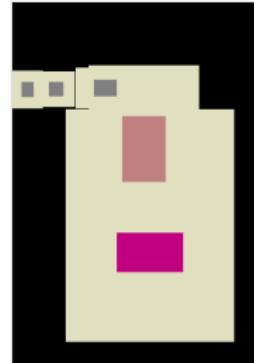
- Observation: Convolutional NNs are somewhat robust to (label-)noise
- Let's use that: Use bounding boxes as target and recursively estimate better targets [6]



- Problem: Training quickly degrades
- **Postprocess** intermediate predictions

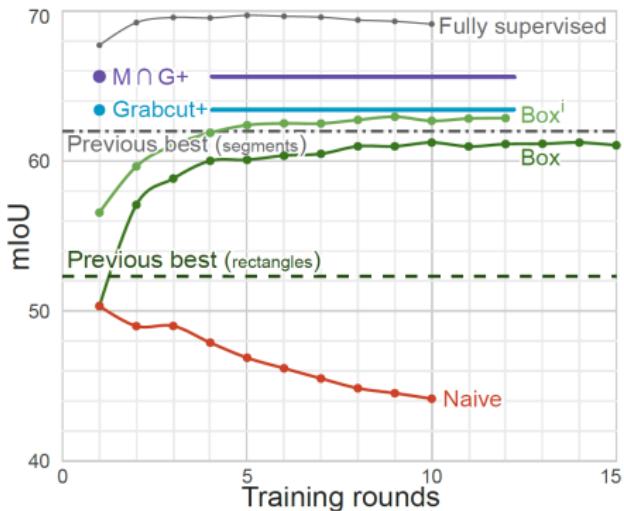
From Bounding Boxes to Segmentation

- **SUPPRESS** detections
 - ...of wrong class
 - ...outside the box
 - ... $<\%$ box area
 - ...outside of conditional random field boundaries
- Additional improvement: smaller boxes
 - Objects are “on average” roundish
 - Corners and edges contain “on average” the least true positives
 - Define “ignore” regions with unknown labels



Source: Khoreva et al. [6]

Improved Recursive Training



- Shrinking boxes beats state of the art
- Combine Grabcut and MCG for initial label
 - No need for recursion

Source: [6]

NEXT TIME
ON DEEP LEARNING



FAU

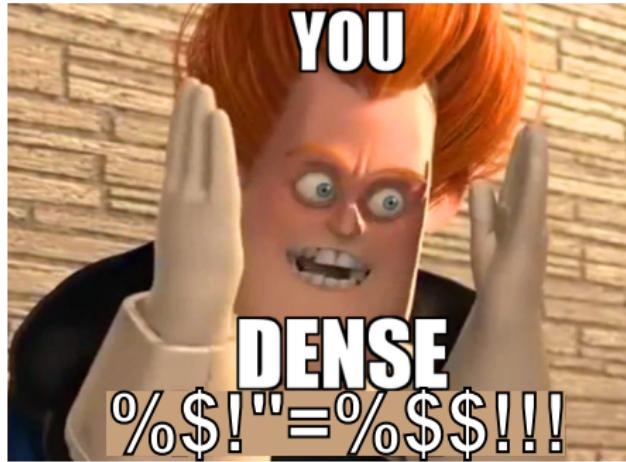
FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Weakly and Self-Supervised Learning - Part 2

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
July 7, 2020



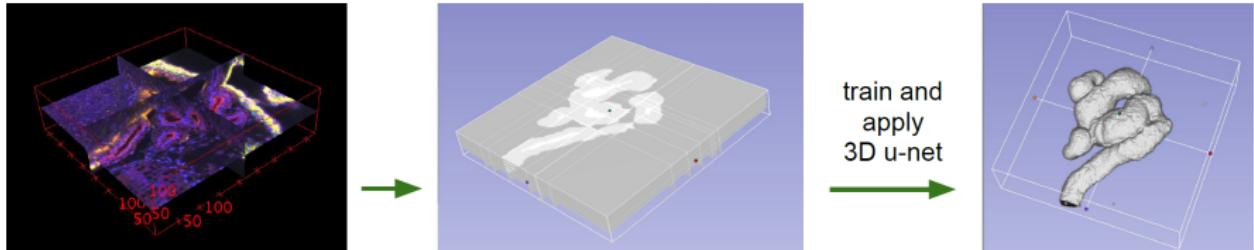
From Sparse Annotations to Dense Segmentations



... not quite

Source: Adapted from <https://knowyourmeme.com/memes/>.

From Sparse Annotations to Dense Segmentations



- 3D segmentation is extremely tedious
- Obtain only a few labelled 2D slides
- Compute automatic segmentation in 3D
- Allows for interactive correction

Source: Çiçek et al. [1]

From Sparse Annotations to Dense Segmentations

Training with sparse labels

- Problem: “One hot” labels $y_{n,i}$ with element i being 1
 → either **true** or **false**

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_n -\log \hat{y}_{n,i} \Big|_{y_{n,i}=1}$$

- Obtain sparse loss by weighted cross entropy [1]

$$L'(\mathbf{y}, \hat{\mathbf{y}}) = L(\mathbf{y}, \hat{\mathbf{y}}) \cdot w(\mathbf{y})$$

where $w(y_{n,i}) = \begin{cases} 0 & \text{if } y_n \text{ is not labelled} \\ w_{n,i} > 0 & \text{otherwise} \end{cases}$

- Can be easily extended to interactive segmentation by updating \mathbf{y} and $w(\mathbf{y})$

Take Away: Weakly Supervised Learning

- Fine grained labels are expensive - can we get away with something cheaper?
- Core definition: **Label** has less detail than **target**
- Methods depend on **prior knowledge** and **weak labels** ("hints")
- Typically inferior to **fully supervised** training
 - but highly relevant in practice
- Don't forget transfer learning (!)
- Related:
 - **Semi-supervised** Learning
 - **Self-supervised** Learning

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Weakly and Self-Supervised Learning - Part 3

**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
July 7, 2020



Outline

Learning with Limited Annotations

Definition

Image-based SSL for Representation Learning

Generative

Spatial Context

Context Similarity

Contrastive SSL

Supervised Contrastive Learning

Bootstrap SSL – A paradigm change



FAU

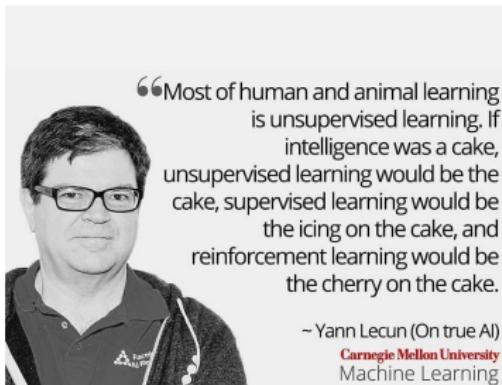
FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Definition



Motivation

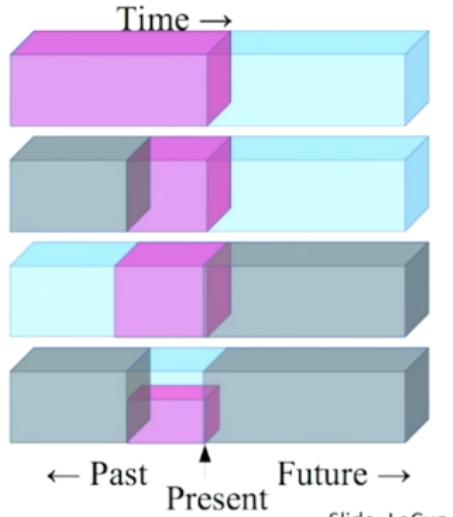
- Jitendra Malik: “Supervision is the opium of the AI researcher”
- Alyosha Efros: “The AI revolution will not be supervised”
- Yann LeCun:



Source: <https://www.facebook.com/722677142/posts/10156036317282143/>

Idea

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the **occluded** from the **visible**
- ▶ **Pretend there is a part of the input you don't know and predict that.**



Slide: LeCun

Source: <https://www.youtube.com/watch?v=7I0Qt7GALVk>

Self-supervised Learning: Definition



Yann LeCun

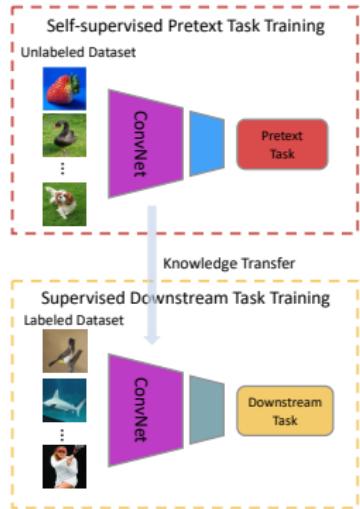
April 30, 2019 · ②

I now call it "self-supervised learning", because
 "unsupervised" is both a loaded and confusing term.

- Subcategory of unsupervised learning
- Use pretext/surrogate/pseudo tasks in a supervised fashion
 - Automatically generated labels
 - Measurement of correctness
- Downstream task: retrieval, supervised or semi-supervised classification, etc.

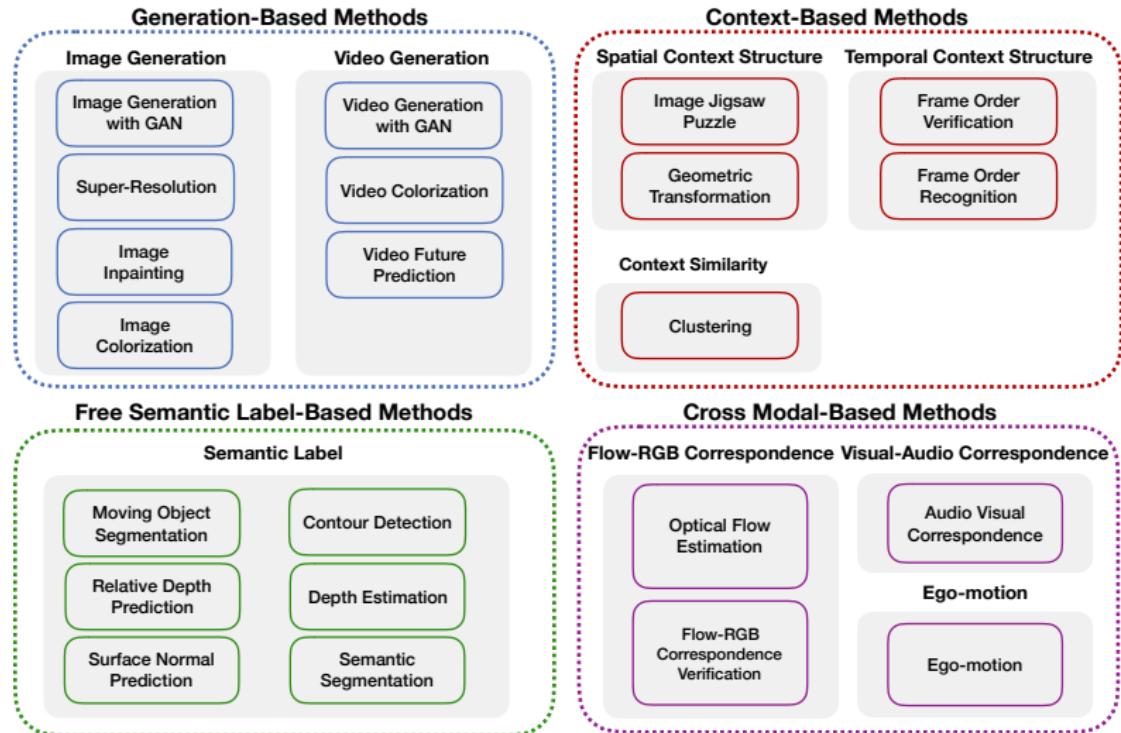
Note: Generative models (e.g. GANs) are also SSL methods

Source: <https://www.facebook.com/722677142/posts/10155934004262143/>



Source: [15]

Pretext Tasks Overview



Source: [15]



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

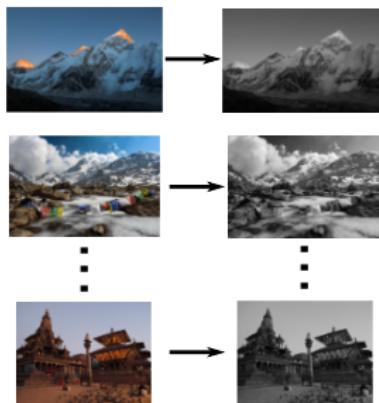
Image-based SSL for Representation Learning



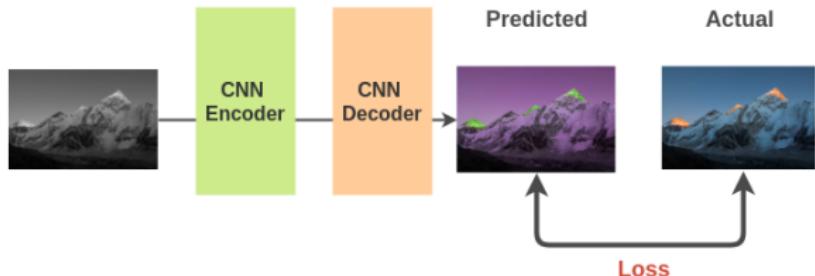
Generative

Image Colorization

Data generation:



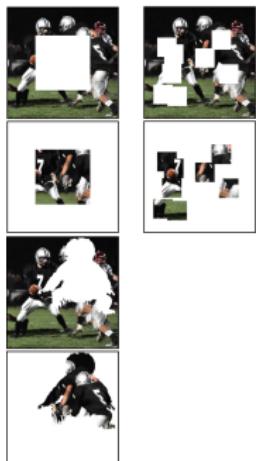
Pretext task: ℓ_2 loss between gray and color version



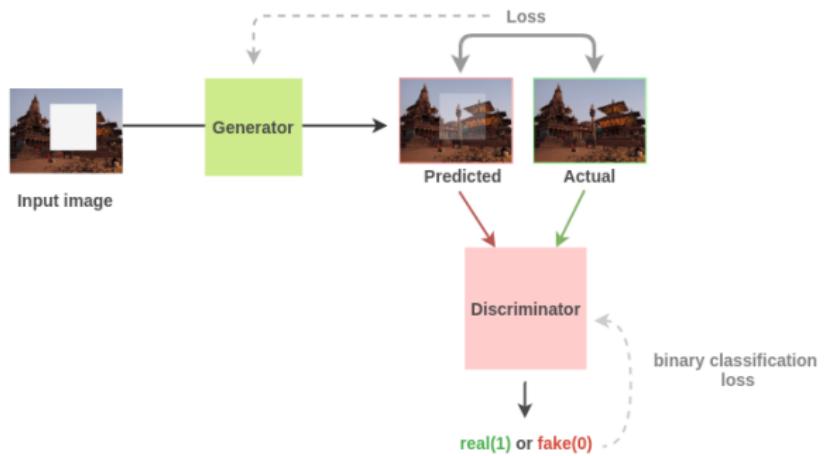
Source: <https://amitness.com/2020/02/illustrated-self-supervised-learning/>

Image Inpainting

Data generation:



Pretext task:

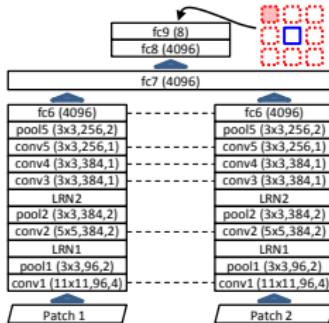
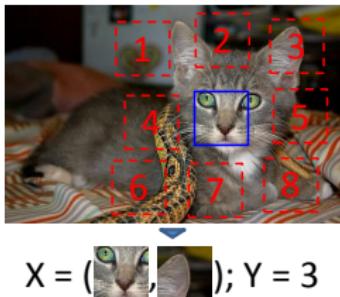


Source: [16]

Source: <https://amitness.com/2020/02/illustrated-self-supervised-learning/>

Spatial Context

Solve Jigsaw Puzzle [17]

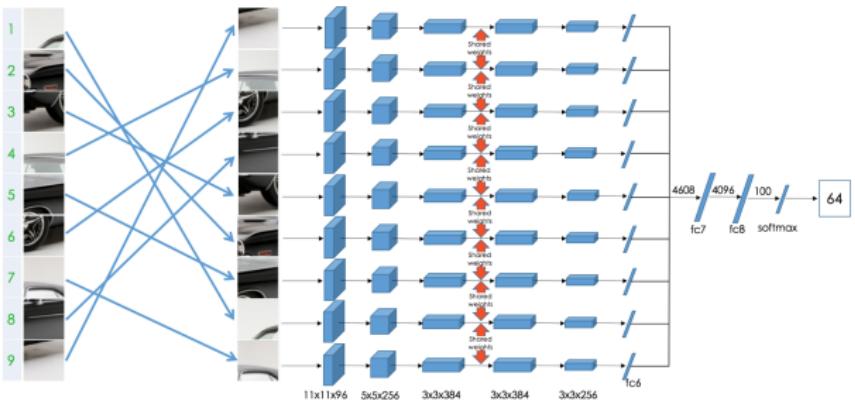
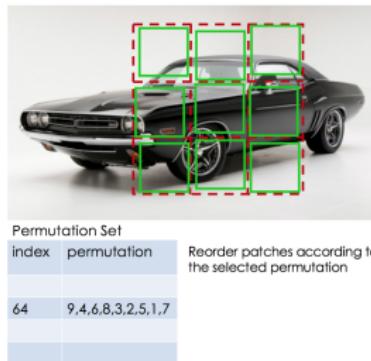


Attention: Trivial solution possible

- boundary patterns, continuing textures → use large enough gaps
- chromatic aberration
 - Pre-process images by shifting green and magenta toward gray
 - randomly drop 2 color channels

Source: [17]

Solve Jigsaw Puzzle++ [18]



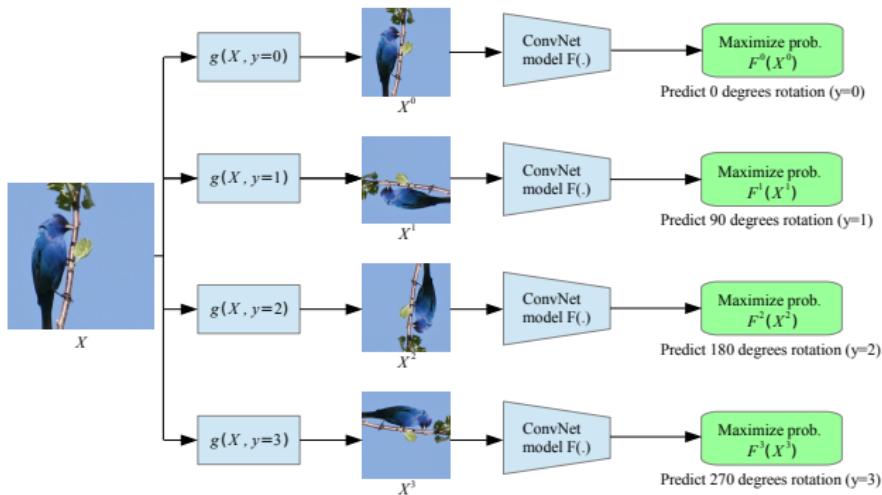
9 tiles $\rightarrow 9! = 362\,880$ possible permutations

Source: [18]

Solve Jigsaw Puzzle++ [18] (cont.)

Number of permutations	Average hamming distance	Minimum hamming distance	Jigsaw task accuracy	Detection performance
1000	8.00	2	71	53.2
1000	6.35	2	62	51.3
1000	3.99	2	54	50.2
100	8.08	2	88	52.6
95	8.08	3	90	52.4
85	8.07	4	91	52.7
71	8.07	5	92	52.8
35	8.13	6	94	52.6
10	8.57	7	97	49.2
7	8.95	8	98	49.6
6	9	9	99	49.7

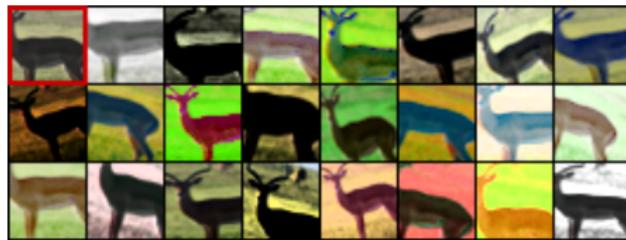
Rotation [19]



Source: [19]

Context Similarity

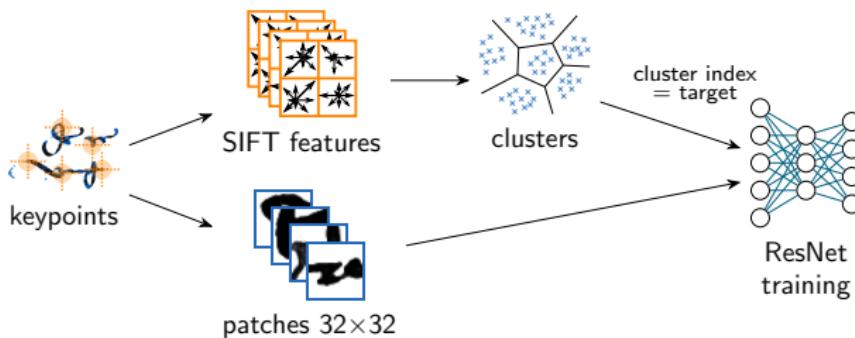
Distortions [21] (Exemplar-CNN)



- For each input patch, create N (e.g. $N = 100$) distorted images
 - All these distorted images form one class
- Discriminate between a set of surrogate classes (e.g. 8000 pseudo-classes)

Source: [21]

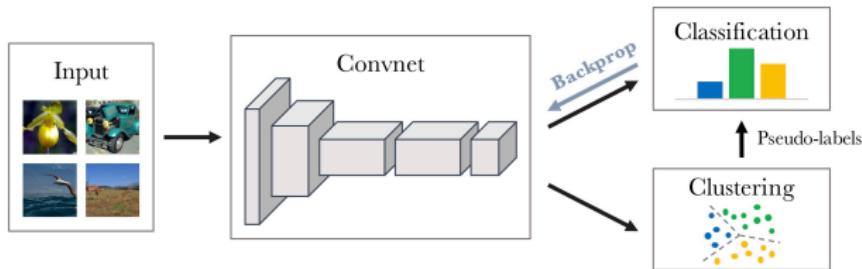
Clustering [22]



- From keypoints in an image extract patches and compute descriptors
- Cluster features of patches using k -means into N clusters ($N = 5000$)
- Use cluster indices as targets for input patches
- Remove features (+patches) in between of two clusters

Source: [22]

Clustering [20] (DeepCluster)

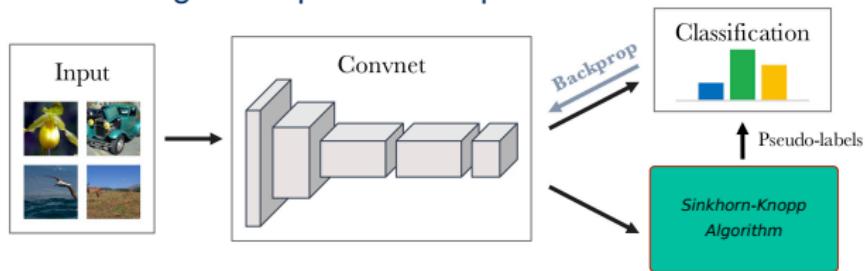


- Alternate between
 - k -means clustering (each epoch) of PCA-whitened ($D = 256$) & ℓ_2 -normalized activation features
 - CNN training
- Avoid trivial solutions
 - Re-assign empty clusters
 - Weight contribution of an input by inverse of the size of its assigned cluster

Source: [20]

Clustering [24]

Self-labelling with Optimal Transport



Optimal transport

Supply		Need	
25 laptops	Warehouse A	Shop 1	25 laptops
25 laptops	Warehouse B	Shop 2	25 laptops

		Distance(cost) matrix		Optimal Allocation	
		Shop 1	Shop 2	Shop 1	Shop 2
Warehouse A	Warehouse A	2km	3km	25	0
Warehouse B	Warehouse B	2km	1km	0	25

Source: <https://amitness.com/2020/04/illustrated-self-labelling/>

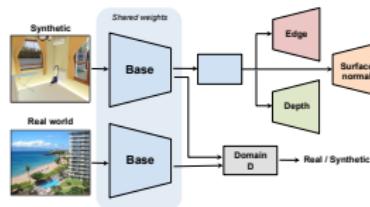
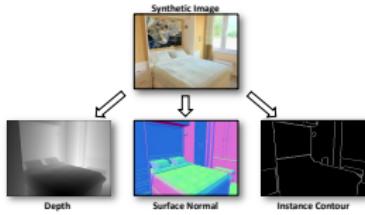
Clustering [24] (cont.)

Self-labelling with Optimal Transport

Comparison to DeepCluster

- no separate clustering loss → can lead to degenerate solutions
- clustering approach that minimizes the same cross-entropy loss that the network also optimize.

Multi-task SSL using Synthetic Imagery [23]



- Given: input synthetic RGB image
- Network simultaneously predicts: surface normal, depth, instance contour
- Additionally: minimize feature space domain differences between real and synthetic data

Source: [23]

**NEXT TIME
ON DEEP LEARNING**



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Weakly and Self-Supervised Learning - Part 4

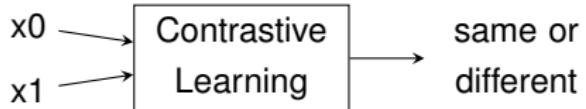
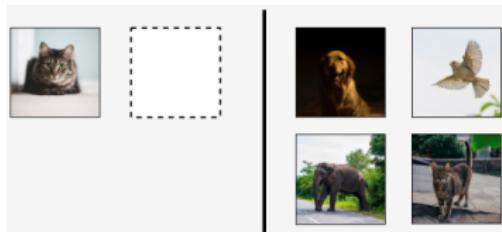
**A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul,
M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang**
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
July 7, 2020



Contrastive SSL

Contrastive Learning

Match the correct animal



Advantages over generative/context models:

- Pixel-level losses could overly focus on pixel-based details, rather than more abstract latent factors
- Pixel-based objectives often assume pixel independence → reduce ability to model correlations or complex structure

Source: <https://amitness.com/2020/03/illustrated-simclr/>

Contrastive Loss

Given: $\mathcal{X} = \{\mathbf{x}, \underbrace{\mathbf{x}^+}_{\text{positive sample}}, \underbrace{\mathbf{x}_1^-, \dots, \mathbf{x}_{N-1}^-}_{\text{negative samples}}\}$; similarity function $s(\cdot)$ (e.g. cosine similarity)

Goal: $s(f(\mathbf{x}), f(\mathbf{x}^+)) >> s(f(\mathbf{x}), f(\mathbf{x}^-))$

Contrastive/InfoNCE Loss (aka ‘n-pair loss’/‘consistency loss’/‘ranking-based NCE’):

Cross-entropy loss for (N)-way softmax classifier

$$\begin{aligned}\mathcal{L}_N &= -\mathbb{E}_{\mathcal{X}} \left[\log \frac{\exp(s(f(\mathbf{x}), f(\mathbf{x}^+)))}{\exp(s(f(\mathbf{x}), f(\mathbf{x}^+))) + \sum_{j=1}^{N-1} \exp(s(f(\mathbf{x}), f(\mathbf{x}_j^-)))} \right] \\ &= -\mathbb{E}_{\mathcal{X}} \left[\log \frac{\exp(s(f(\mathbf{x}), f(\mathbf{x}^+)))}{\sum_{j=1}^N \exp(s(f(\mathbf{x}), f(\mathbf{x}_j)))} \right]\end{aligned}$$

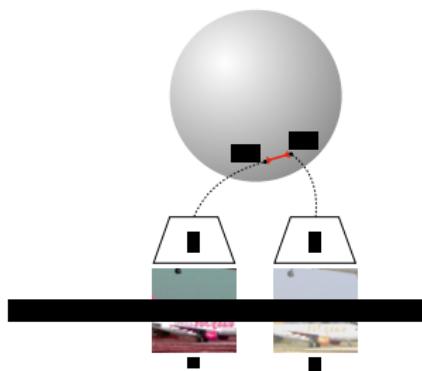
Contrastive Loss (cont.)

Minimizing Contrastive Loss maximizes a lower bound on the mutual information between $f(\mathbf{x})$ and $f(\mathbf{x}^+)$ [25], [27].

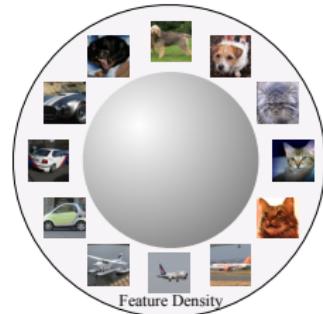
Common Variation: temperature hyperparameter τ

$$\mathcal{L}_N = -\mathbb{E}_{\mathcal{X}} \left[\log \frac{\exp(s(f(\mathbf{x}), f(\mathbf{x}^+))/\tau)}{\sum_{j=1}^{N+1} \exp(s(f(\mathbf{x}), f(\mathbf{x}_j))/\tau)} \right]$$

Effectivity of Contrastive Loss



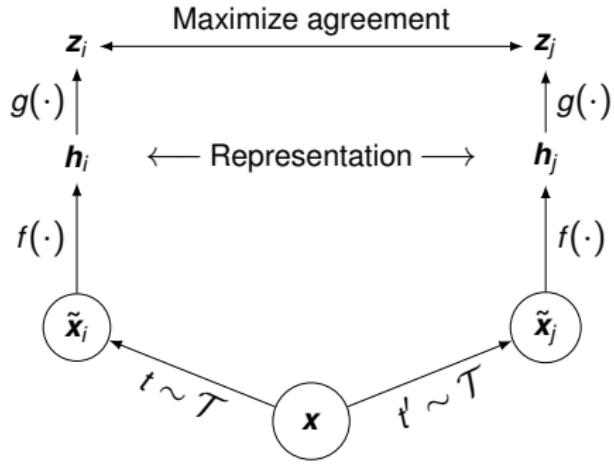
Alignment: Similar samples have similar features



Uniformity: Preserve maximal information.

Source: [35]

Examples: SimCLR [31]



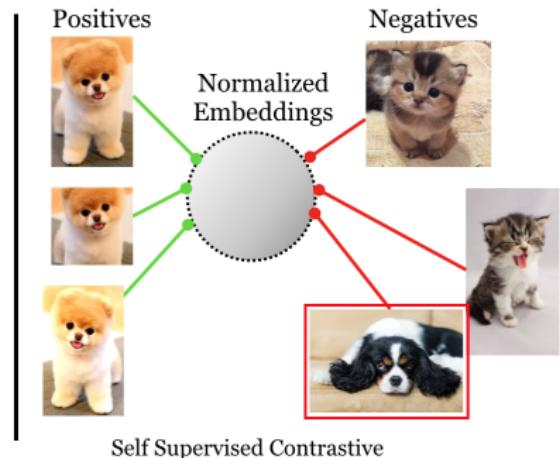
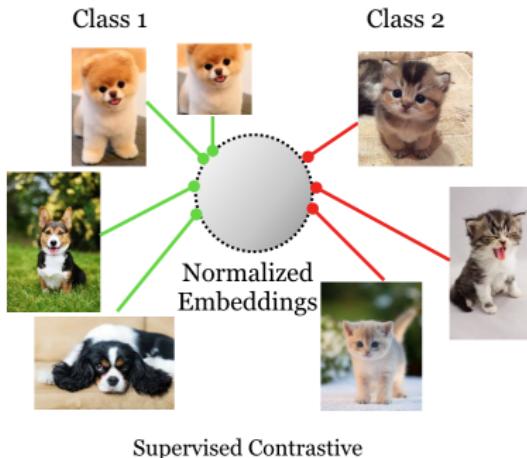
1. Mini-batch of n samples. Each sample is applied with two different data augmentation operations $\rightarrow 2n$ augmented samples: $\tilde{\mathbf{x}}_i = f(\mathbf{x}), \quad \tilde{\mathbf{x}}_j = f'(\mathbf{x}), \quad t, t' \sim \mathcal{T}$
2. One positive pair, $2(n - 1)$ negatives. Representation through base encoder f :

$$\mathbf{h}_i = f(\tilde{\mathbf{x}}_i), \quad \mathbf{h}_j = f(\tilde{\mathbf{x}}_j)$$
3. Contrastive loss on top of representation head g :

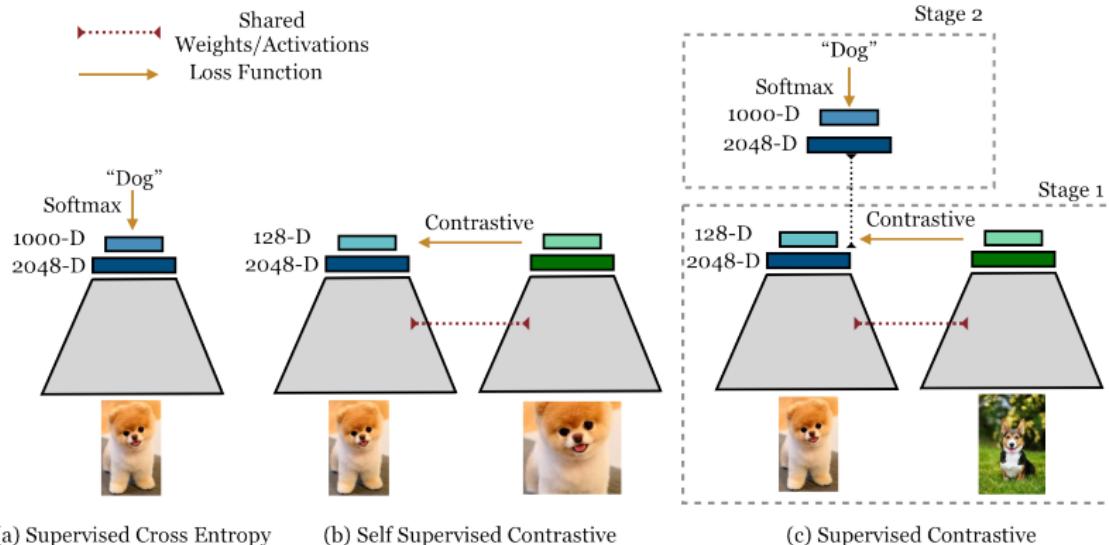
$$\mathcal{L}_{i,j} = -\log \frac{\exp(s(\mathbf{z}_i, \mathbf{z}_j) / \tau)}{\sum_{k=1}^{2n} \mathbf{1}_{[k \neq i]} \exp(s(\mathbf{z}_i, \mathbf{z}_k) / \tau)}$$

Supervised Contrastive Learning

Supervised Contrastive Learning [33]



Supervised Contrastive Learning [33]



Source: [33]

Supervised Contrastive Loss

- Self-supervised **has no** knowledge about class labels → only one positive example
- Supervised **has** knowledge about class labels → many positives per example
- Compute loss between any sample \mathbf{z}_j having the same class as anchor \mathbf{z}_i ($\mathbf{y}_i = \mathbf{y}_j$)

$$L_{\text{sup}} = \sum_{i=1}^{2N} - \dots \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\mathbf{y}_i = \mathbf{y}_j} \cdot \log \frac{\exp(\mathbf{z}_i^\top \mathbf{z}_j / \tau)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp(\mathbf{z}_i^\top \mathbf{z}_k / \tau)}$$

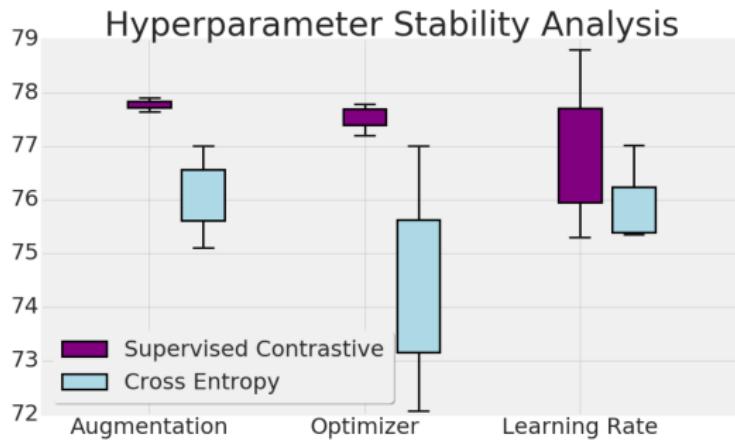
Supervised Contrastive Loss (cont.)

- Vectors \mathbf{z} need to be normalized, i.e. $\mathbf{z} = \mathbf{w}/\|\mathbf{w}\|$, where \mathbf{w} is the output of the projection network
- Gradient w.r.t. to \mathbf{w} is high for hard positives and negatives and small otherwise → “built-in” focal loss
- For one positive and one negative it turns out that

$$L_{\text{sup}} \propto \|\mathbf{z}_a - \mathbf{z}_p\|^2 - \|\mathbf{z}_a - \mathbf{z}_n\|^2 + 2\tau$$

→ Common contrastive loss in siamese networks

Hyperparameter stability



- Increased stability w.r.t. to non-optimal hyperparameters

Source: [33]

What else?

- Training about 50% slower than CE
- Improves over training with SOTA data augmentation (CutMix)
- Enables unsupervised clustering in latent space → correction of label-noise, new possibilities for semi-supervised, ...

Bootstrap SSL – A paradigm change

Bootstrap Your Own Latent (BYOL) [34] Overview

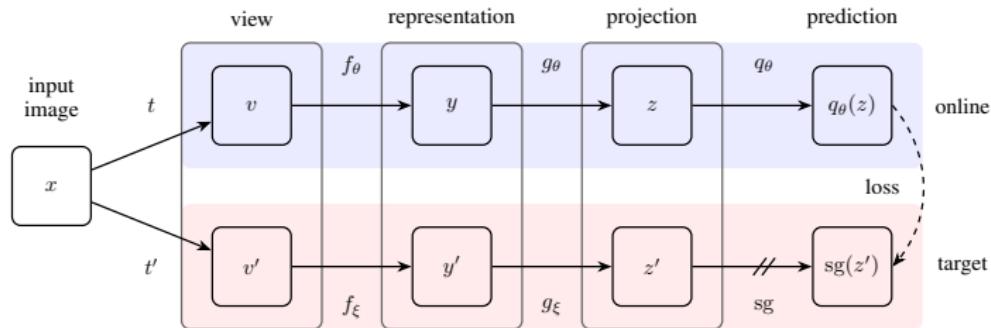
So far: Contrastive loss between exemplar, positive and negatives

- Negative pairs critical (often: large batchsizes, memory banks, custom mining strategies)
- Right augmentation strategy critical

BYOL:

- No negative pair
- No contrastive loss
- More resilient to changes in batch size and set of image augmentations compared to its contrastive counterparts

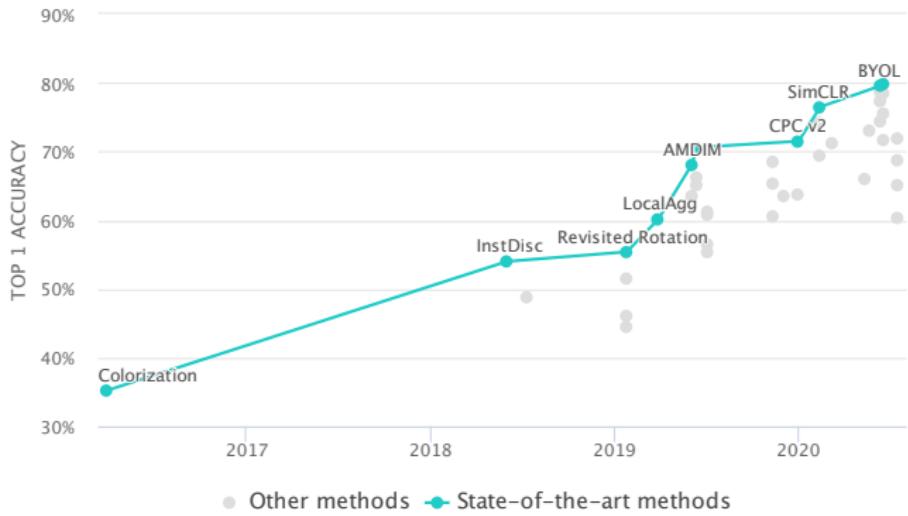
BYOL [34] Method



- Two networks: **online** and **target** network → interact and learn from each other
- In theory: trivial solution possible (e.g. zero for all images)
→ use slow-moving average of the online network as target network
- Loss: MSE of ℓ^2 -normalized predictions (proportional to cosine distance)

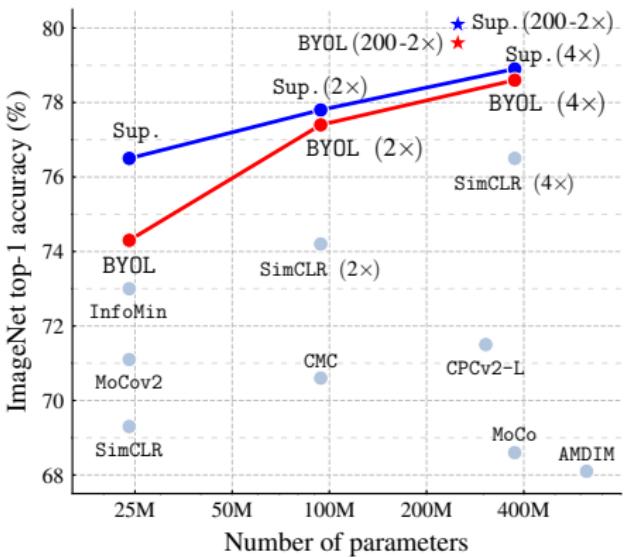
Source: [34]

SSL State of the Art



Source: <https://paperswithcode.com/sota/self-supervised-image-classification-on>

SSL State of the Art (cont.)



Source: [34]

Further Reading

Blogs:

- [https://lilianweng.github.io/lil-log/2019/11/10/
self-supervised-learning.html](https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html)
- <https://amitness.com/2020/02/illustrated-self-supervised-learning/>
- [https://ankeshanand.com/blog/2020/01/26/
contrative-self-supervised-learning.html](https://ankeshanand.com/blog/2020/01/26/contrative-self-supervised-learning.html)

Others:

- <https://github.com/jason718/awesome-self-supervised-learning>
- <https://www.youtube.com/watch?v=7I0Qt7GALVk>

**NEXT TIME
ON DEEP LEARNING**

Next Time: Emerging Methods

- Can we process graphs using deep networks?
- Do we really have to learn everything from scratch?
- Let's see whether we can re-use prior knowledge in deep learning...



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

References



References I

- [1] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, et al. "3d u-net: learning dense volumetric segmentation from sparse annotation". In: [MICCAI](#). Springer. 2016, pp. 424–432.
- [2] Waleed Abdulla. [Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow](#). Accessed: 27.01.2020. 2017.
- [3] Olga Russakovsky, Amy L. Bergman, Vittorio Ferrari, et al. "What's the point: Semantic segmentation with point supervision". In: [CoRR](#) abs/1506.02106 (2015). arXiv: 1506.02106.
- [4] Marius Cordts, Mohamed Omran, Sebastian Ramos, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: [CoRR](#) abs/1604.01685 (2016). arXiv: 1604.01685.

References II

- [5] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern classification. 2nd ed. New York: Wiley-Interscience, Nov. 2000.
- [6] Anna Khoreva, Rodrigo Benenson, Jan Hosang, et al. "Simple Does It: Weakly Supervised Instance and Semantic Segmentation". In: arXiv preprint arXiv:1603.07485 (2016).
- [7] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al. "Mask R-CNN". In: CoRR abs/1703.06870 (2017). arXiv: 1703.06870.
- [8] Sangheum Hwang and Hyo-Eun Kim. "Self-Transfer Learning for Weakly Supervised Lesion Localization". In: MICCAI. Springer. 2016, pp. 239–246.
- [9] Maxime Oquab, Léon Bottou, Ivan Laptev, et al. "Is object localization for free? weakly-supervised learning with convolutional neural networks". In: Proc. CVPR. 2015, pp. 685–694.

References III

- [10] Alexander Kolesnikov and Christoph H. Lampert. "Seed, Expand and Constrain: Three Principles for Weakly-Supervised Image Segmentation". In: [CoRR abs/1603.06098](#) (2016). arXiv: [1603.06098](#).
- [11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, et al. "Microsoft COCO: Common Objects in Context". In: [CoRR abs/1405.0312](#) (2014). arXiv: [1405.0312](#).
- [12] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, et al. "Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization". In: [CoRR abs/1610.02391](#) (2016). arXiv: [1610.02391](#).
- [13] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: [Proc. ICLR \(workshop track\)](#). 2014.

References IV

- [14] Bolei Zhou, Aditya Khosla, Agata Lapedriza, et al. "Learning deep features for discriminative localization". In: Proc. CVPR. 2016, pp. 2921–2929.
- [15] Longlong Jing and Yingli Tian. "Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey". In: arXiv e-prints, arXiv:1902.06162 (Feb. 2019). arXiv: 1902.06162 [cs.CV].
- [16] D. Pathak, P. Krähenbühl, J. Donahue, et al. "Context Encoders: Feature Learning by Inpainting". In:
2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016, pp. 2536–2544.
- [17] C. Doersch, A. Gupta, and A. A. Efros. "Unsupervised Visual Representation Learning by Context Prediction". In:
2015 IEEE International Conference on Computer Vision (ICCV). Dec. 2015, pp. 1422–1430.

References V

- [18] Mehdi Noroozi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: Computer Vision – ECCV 2016. Cham: Springer International Publishing, 2016, pp. 69–84.
- [19] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: International Conference on Learning Representations. 2018.
- [20] Mathilde Caron, Piotr Bojanowski, Armand Joulin, et al. "Deep Clustering for Unsupervised Learning of Visual Features". In: Computer Vision – ECCV 2018. Cham: Springer International Publishing, 2018, pp. 139–156.

References VI

- [21] A. Dosovitskiy, P. Fischer, J. T. Springenberg, et al. "Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 38.9 (Sept. 2016), pp. 1734–1747.
- [22] V. Christlein, M. Gropp, S. Fiel, et al. "Unsupervised Feature Learning for Writer Identification and Writer Retrieval". In: 2017 14th IAPR International Conference on Document Analysis and Recognition Vol. 01. Nov. 2017, pp. 991–997.
- [23] Z. Ren and Y. J. Lee. "Cross-Domain Self-Supervised Multi-task Feature Learning Using Synthetic Imagery". In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. June 2018, pp. 762–771.

References VII

- [24] Asano YM., Rupprecht C., and Vedaldi A. "Self-labelling via simultaneous clustering and representation learning". In: [International Conference on Learning Representations. 2020.](#)
- [25] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, et al. "On Variational Bounds of Mutual Information". In: [Proceedings of the 36th International Conference on Machine Learning.](#) Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 5171–5180.
- [26] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, et al. "Learning deep representations by mutual information estimation and maximization". In: [International Conference on Learning Representations. 2019.](#)

References VIII

- [27] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: [arXiv e-prints](#), arXiv:1807.03748 (July 2018). arXiv: 1807.03748 [cs.LG].
- [28] Philip Bachman, R Devon Hjelm, and William Buchwalter. “Learning Representations by Maximizing Mutual Information Across Views”. In: [Advances in Neural Information Processing Systems 32](#). Curran Associates, Inc., 2019, pp. 15535–15545.
- [29] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive Multiview Coding”. In: [arXiv e-prints](#), arXiv:1906.05849 (June 2019), arXiv:1906.05849. arXiv: 1906.05849 [cs.CV].
- [30] Kaiming He, Haoqi Fan, Yuxin Wu, et al. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: [arXiv e-prints](#), arXiv:1911.05722 (Nov. 2019). arXiv: 1911.05722 [cs.CV].

References IX

- [31] Ting Chen, Simon Kornblith, Mohammad Norouzi, et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: [arXiv e-prints](#), arXiv:2002.05709 (Feb. 2020), arXiv:2002.05709. arXiv: 2002.05709 [cs.LG].
- [32] Ishan Misra and Laurens van der Maaten. "Self-Supervised Learning of Pretext-Invariant Representations". In: [arXiv e-prints](#), arXiv:1912.01991 (Dec. 2019). arXiv: 1912.01991 [cs.CV].
- [33] Prannay Khosla, Piotr Teterwak, Chen Wang, et al. "Supervised Contrastive Learning". In: [arXiv e-prints](#), arXiv:2004.11362 (Apr. 2020). arXiv: 2004.11362 [cs.LG].

References X

- [34] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning”. In: [arXiv e-prints](#), arXiv:2006.07733 (June 2020), arXiv:2006.07733. arXiv: 2006 . 07733 [cs.LG].
- [35] Tongzhou Wang and Phillip Isola. “Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere”. In: [arXiv e-prints](#), arXiv:2005.10242 (May 2020), arXiv:2005.10242. arXiv: 2005 . 10242 [cs.LG].
- [36] Junnan Li, Pan Zhou, Caiming Xiong, et al. “Prototypical Contrastive Learning of Unsupervised Representations”. In: [arXiv e-prints](#), arXiv:2005.04966 (May 2020), arXiv:2005.04966. arXiv: 2005 . 04966 [cs.CV].



Graph Deep Learning

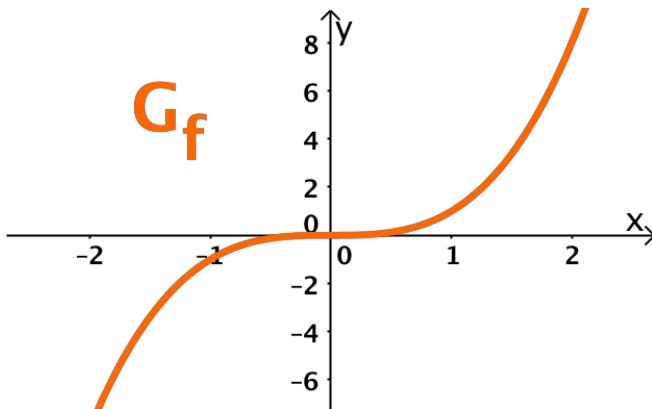
Part 1

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, C. Bergler, F. Denzinger, B. Geissler, Z. Yang, A. Popp, M. Nau

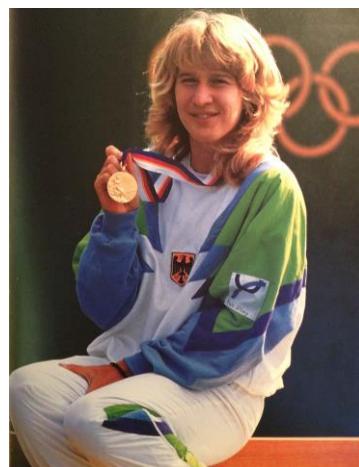
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg



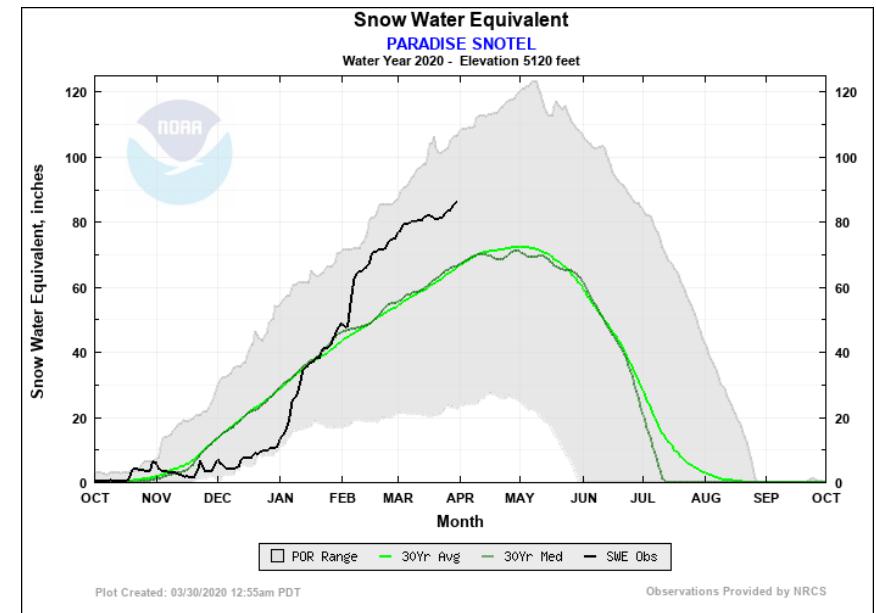
Graph Deep Learning



[a]



Steffi Graf (1999)



[b]



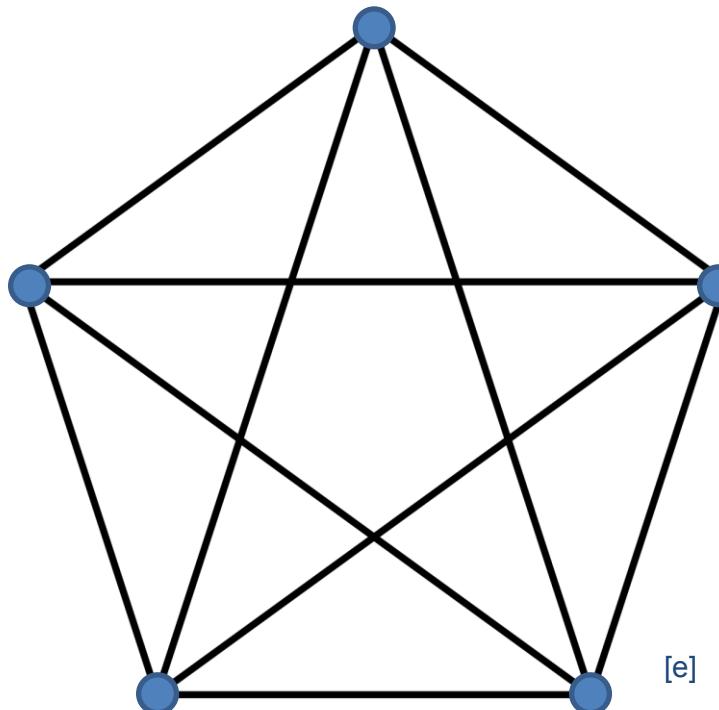
[c]

[d]

Graph Deep Learning

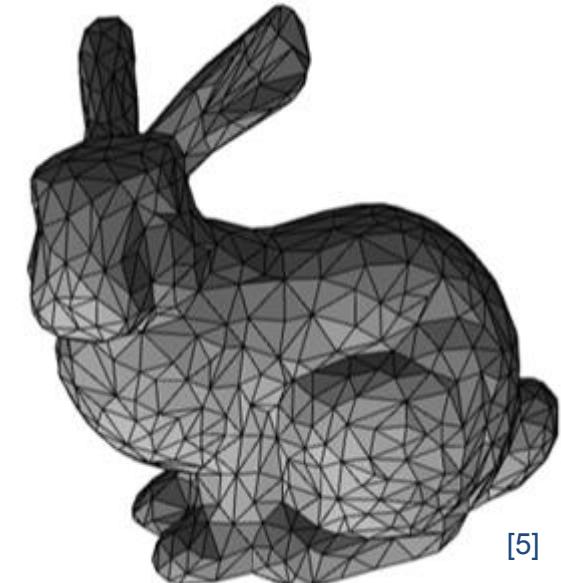
Computer Scientist:

A **Graph** is
a set of nodes
connected through edges



Mathematician:

A **Graph** is
a manifold
but a discrete one

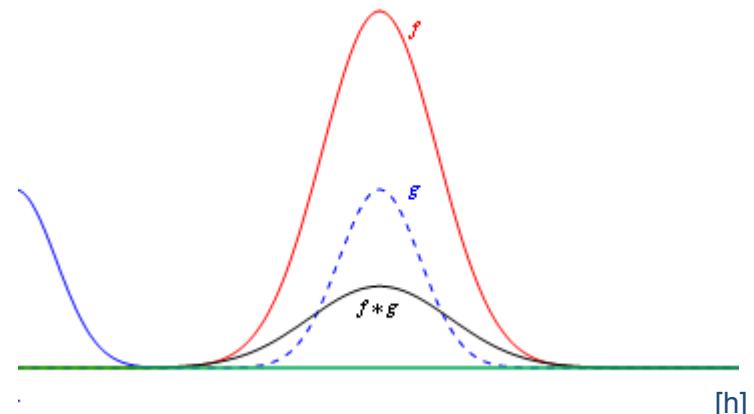
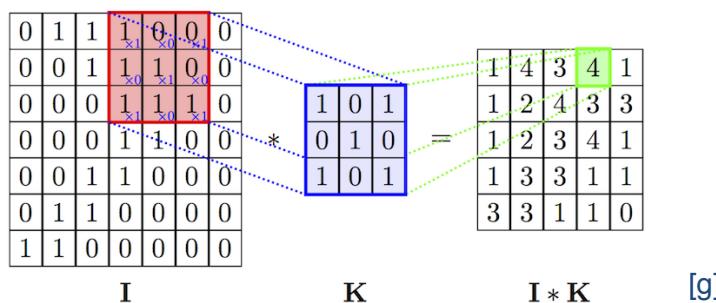
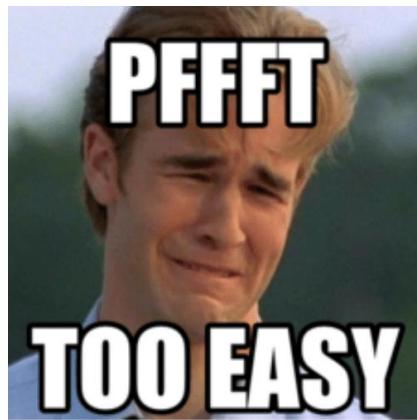


How would you define a convolution on Euclidean space?

Computer Scientist + Mathematician:

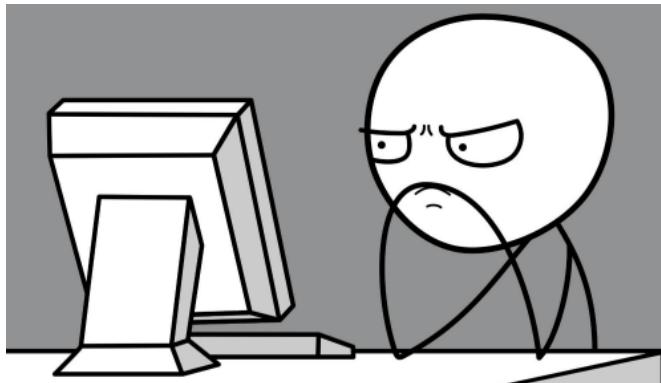
$$(f \star g)(n) = \sum_{k \in D} f(k)g(n - k)$$

$$(f \star g)(n) = \int_{\mathbb{R}^n} f(\tau)g(x - \tau)d\tau$$



How would you define a convolution on graphs?

Computer Scientist:



Mathematician:

$$\Delta f = -\text{div}(\nabla f)$$

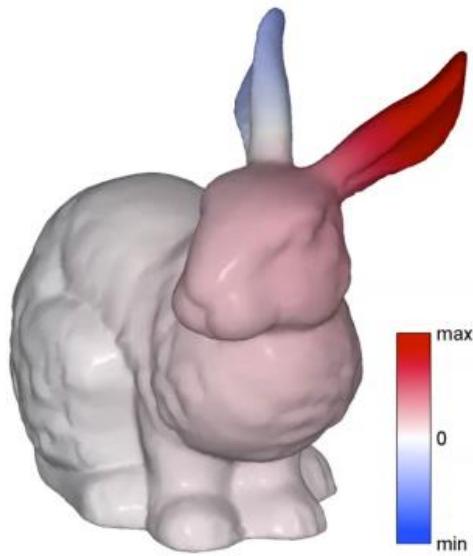


How would you define a convolution on graphs?

Manifold Idea:

1. We know to convolve manifolds
2. We can discretize convolutions
3. Thus, we know how to convolve graphs

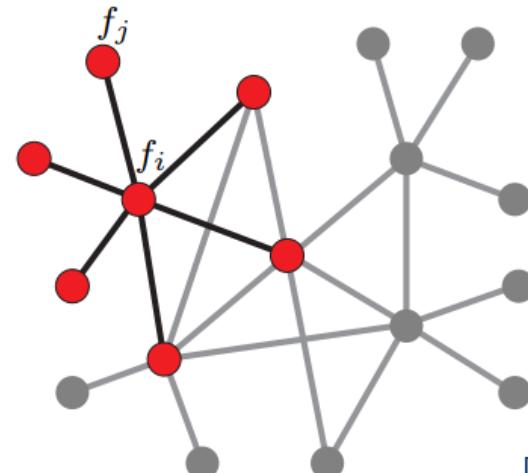
Convolution on manifolds



[5]

discretize

Convolution on graphs



[5]

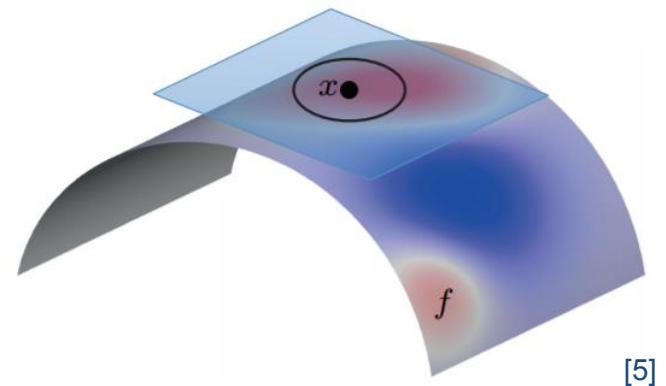
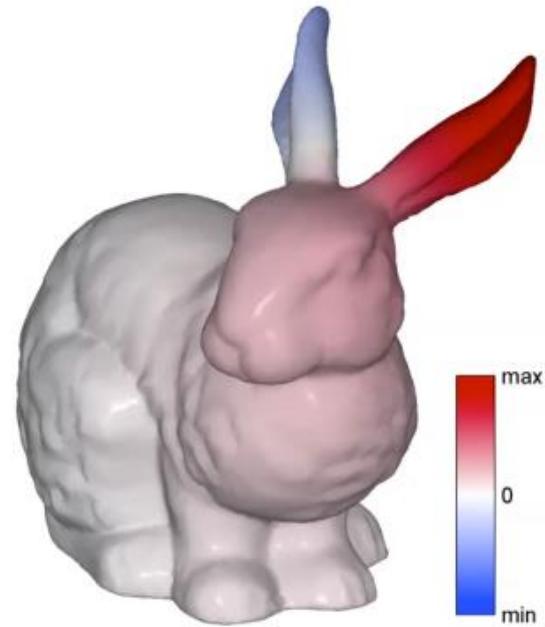
Graph Deep Learning

Lets diffuse some heat with Newton's Law of Cooling [6]:

$$f_t(x, t) = -\Delta f(x, t)$$

$$f(x, 0) = f_0(x)$$

- $f(x, t)$ amount of heat at point x at time t
- $f_0(x)$ initial heat distribution
- $\Delta f(x) = - \operatorname{div}(\nabla f)$ (Laplacian)
difference between $f(x)$ and the average of f on an infinitesimal sphere around x

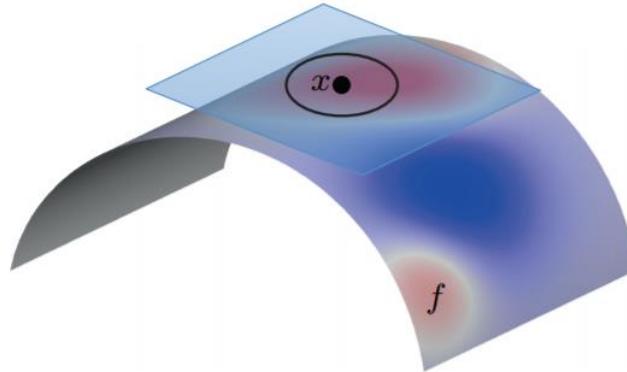


Graph Deep Learning

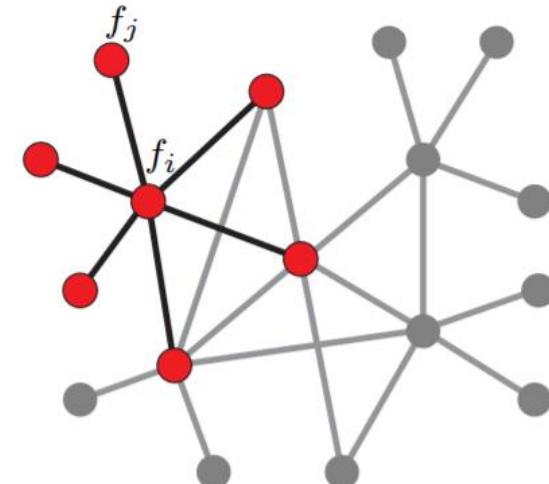
How do we express the Laplacian in a discrete form?

- $\Delta f(x) = -\operatorname{div}(\nabla f)$

“difference between $f(x)$ and the average of f on an infinitesimal sphere around x “



$$(\Delta f)_i = \frac{1}{d_i} \sum_{j:(i,j) \in E} a_{ij}(f_i - f_j)$$



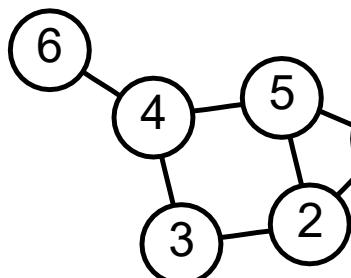
[5]

Graph Deep Learning

Is there another way of expressing this? (Below without the normalization d_i)

$$(\Delta f)_i = \sum_{j:(i,j) \in E} a_{ij}(f_i - f_j)$$

Yes.



$$\begin{aligned} \mathbf{D} &= \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ \mathbf{A} &= \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \\ \Delta &= \mathbf{D} - \mathbf{A} \end{aligned}$$

Graph Deep Learning

- $\Delta \in \mathbb{R}^{N \times N}$ is known as the **Laplacian Matrix** of a (sub-)graph consisting of N nodes
- $D \in \mathbb{R}^{N \times N}$ is the **Degree Matrix** and describes the number of edges connected to each node
- $A \in \mathbb{R}^{N \times N}$ is the **Adjacency Matrix** and describes the connectivity of the graph
- For a directed graph Δ is not s.p.d. thus we normalize Δ and get Δ_{sym} s.t.

$$\begin{aligned}\Delta &= D - A \\ \Delta_{sym} &= D^{-\frac{1}{2}} \Delta D^{-\frac{1}{2}} \\ \Delta_{sym} &= D^{-\frac{1}{2}} (D - A) D^{-\frac{1}{2}} \\ \Delta_{sym} &= D^{-\frac{1}{2}} D D^{-\frac{1}{2}} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \\ \Delta_{sym} &= I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\end{aligned}$$

Graph Deep Learning

Let's do some magic!

- Δ_{sym} is now s.p.d.

$$\Delta_{sym} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T$$

$$\mathbf{U} = [\mathbf{u}_0, \dots, \mathbf{u}_{N-1}] \in \mathbb{R}^{N \times N}$$

$$\boldsymbol{\Lambda} = \text{diag}([\lambda_0, \dots, \lambda_{N-1}]) \in \mathbb{R}^{N \times N}$$

- $\mathbf{u}_0, \dots, \mathbf{u}_{N-1} \in \mathbb{R}^N$: Eigenvectors are known as the graph Fourier modes
- $\lambda_0, \dots, \lambda_{N-1} \in \mathbb{R}$: Eigenvalues are known as the spectral frequencies
- **That means:** We can use \mathbf{U} and \mathbf{U}^T in order to Fourier transform a graph whereas $\boldsymbol{\Lambda}$ are the spectral filter coefficients!

Graph Deep Learning

Let's do some magic!

- Let $x \in \mathbb{R}^N$ be some signal (a scalar for every node)
- and using the Laplacians eigenvectors we can define its Fourier transform using U^T

$$\hat{x} = U^T x$$

and inverse

$$x = U\hat{x}$$

- We can therefore describe as convolution with a filter g in spectral domain

$$g * x = U((U^T g) \cdot (U^T x))$$

- Lets construct a filter \widehat{G} composed by a k -th order polynomial of Laplacians with $\theta_i \in \mathbb{R}$

$$\widehat{G} = \sum_i^k \theta_i \Lambda^i = \theta_k \Lambda^k + \dots + \theta_1 \Lambda^1 + \theta_0$$

Graph Deep Learning

Let's do some magic!

- Lets construct a filter $\widehat{\mathbf{G}}$ composed by a k -th order polynomial of Laplacians

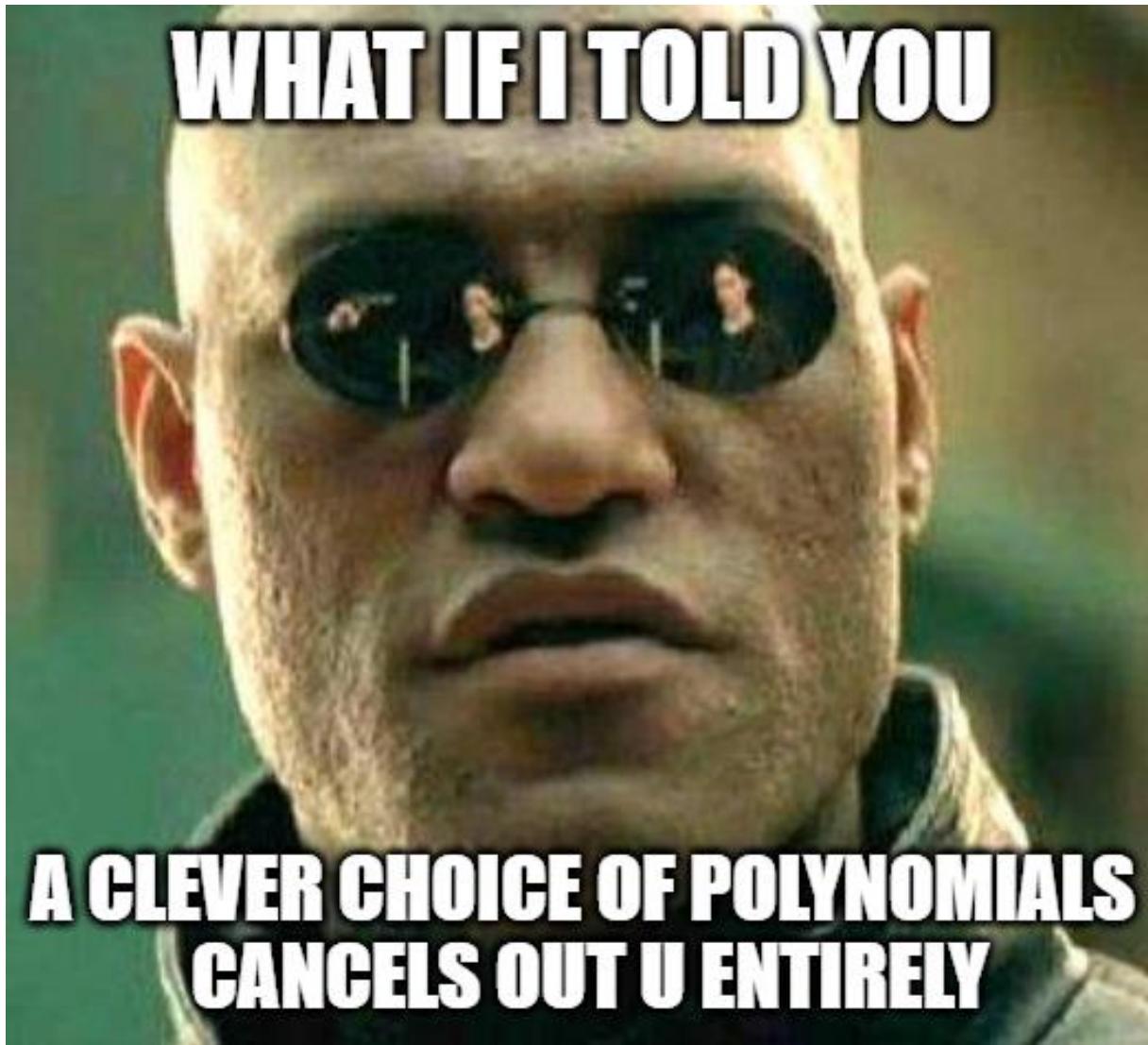
$$\widehat{\mathbf{G}} = \sum_i^k \theta_i \Lambda^i$$

- and filter some signal

$$\mathbf{U} \widehat{\mathbf{G}} \mathbf{U}^T \mathbf{x} = \mathbf{U} \left(\sum_i^k \theta_i \Lambda^i \right) \mathbf{U}^T \mathbf{x}$$

- **And now what?!**
 - We can convolve now \mathbf{x} using Laplacian as we adapt θ_i
 - ... but \mathbf{U} is heavy to compute for every (sub-)graph we want to convolve!

Graph Deep Learning



Graph Deep Learning

Part 2

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, C. Bergler, F. Denzinger, B. Geissler, Z. Yang, A. Popp, M. Nau

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



Graph Deep Learning

$$\mathbf{U}\widehat{\mathbf{G}}\mathbf{U}^T \mathbf{x} = \mathbf{U} \left(\sum_i^k \theta_i \boldsymbol{\Lambda}^i \right) \mathbf{U}^T \mathbf{x}$$

- **Remedy:** We choose k and θ such that we get rid of \mathbf{U}
- Let $k = 1$, $\theta_0 = 2\theta$ and $\theta_1 = -\theta$ we get the following polynomial

$$\begin{aligned}
 \mathbf{U}\widehat{\mathbf{G}}\mathbf{U}^T \mathbf{x} &= \mathbf{U}(2\theta\boldsymbol{\Lambda}^0 - \theta\boldsymbol{\Lambda}^1)\mathbf{U}^T \mathbf{x} \\
 &= (\mathbf{U}2\theta\boldsymbol{\Lambda}^0\mathbf{U}^T - \mathbf{U}\theta\boldsymbol{\Lambda}\mathbf{U}^T)\mathbf{x} \\
 &= (2\theta\mathbf{U}\mathbf{U}^T - \theta\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T)\mathbf{x} \quad \Delta_{sym} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T \\
 &= (2\theta\mathbf{I} - \theta\Delta_{sym})\mathbf{x} \\
 &= \theta(2\mathbf{I} - \Delta_{sym})\mathbf{x} \quad \Delta_{sym} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \\
 &= \theta(2\mathbf{I} - \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x} \\
 &= \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}
 \end{aligned}$$

Graph Deep Learning

We can convolve x in spectral domain

Polynomial is $k = 1$,
 $\theta_0 = 2\theta$ and $\theta_1 = -\theta$
now only depends on θ

$$U \widehat{G} U^T x = \theta(I + D^{-1/2} A D^{-1/2})x$$

We construct \widehat{G} as a polynomial of Laplacian filters

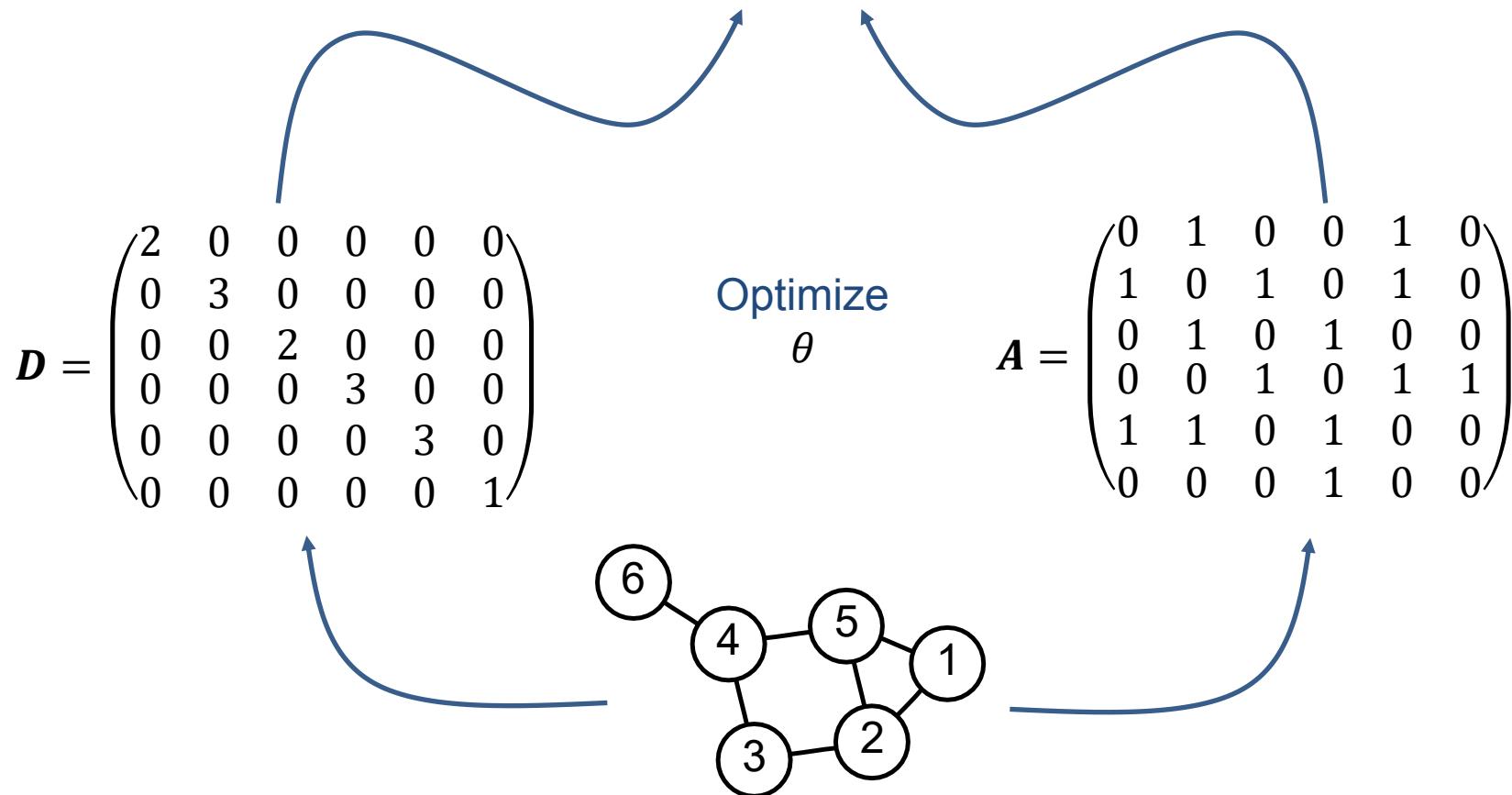
$$\widehat{G} = \sum_i^k \theta_i \Lambda^i$$

With all restrictions we got rid of the Fourier transform U^T

Graph Deep Learning

Basic GCN Operation [1]:

$$\mathbf{U}\widehat{\mathbf{G}}\mathbf{U}^T \mathbf{x} = \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$$

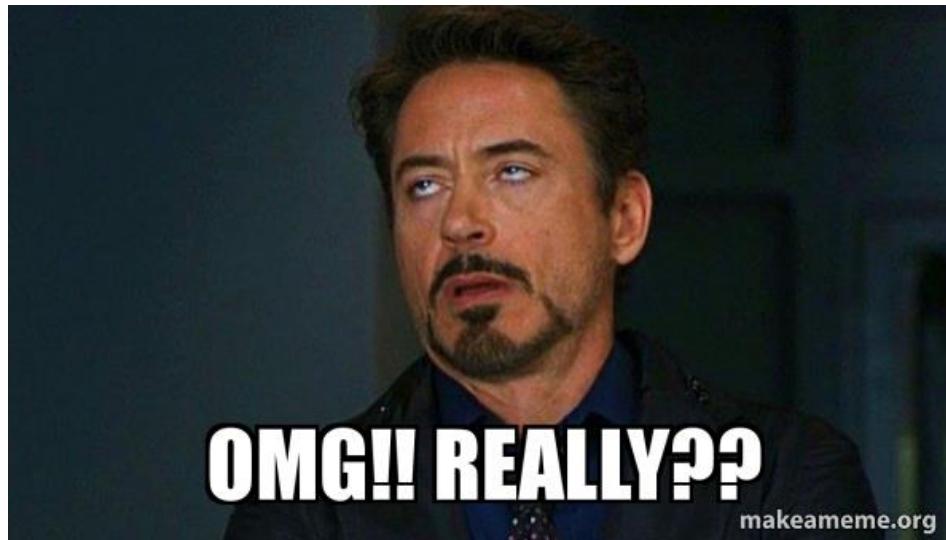


[1]: Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

Graph Deep Learning

Question:

Is it *really* necessary to motivate the Graph Convolution from Spectral Domain?



No.

We can motivate spatially as well

Graph Deep Learning

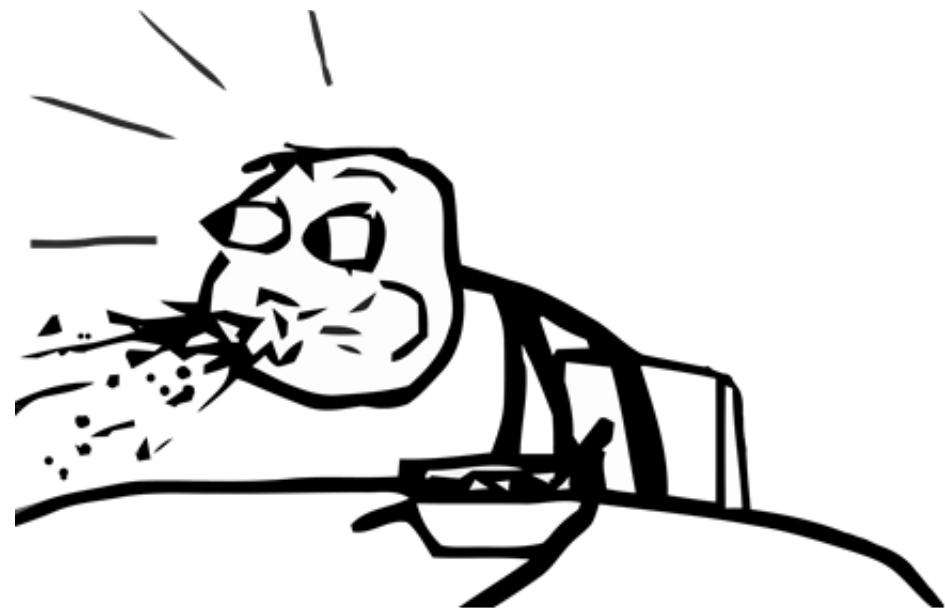
Computer Scientist:

A **Graph** is
a set of nodes (vertices)
connected through edges

We define how to **aggregate**
the information of one Vertex
through its **Neighbors**

Spatial Graph Convolution

Mathematician:



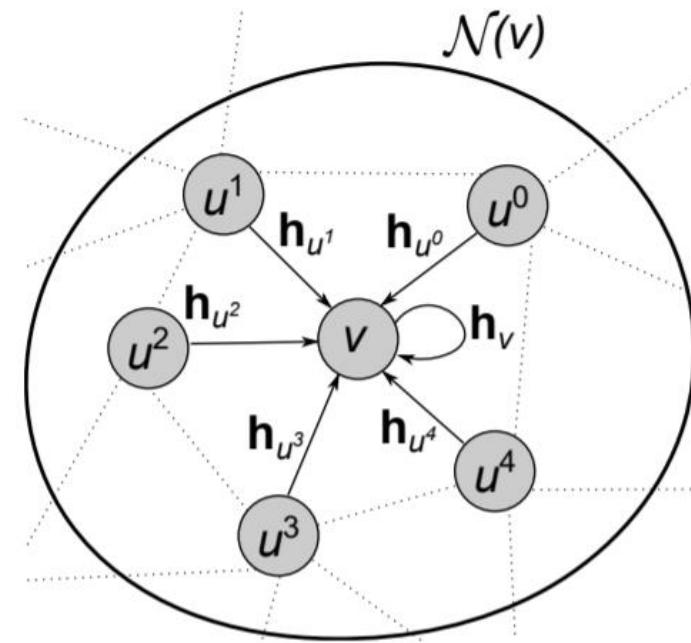
Spectral Graph Convolution

Graph Deep Learning

GraphSAGE [2]

Practically:

- We define a vertex of interest
- We define how neighbors contribute to vertex of interest



[3]

Technically:

- Feature vector at node v in k -th layer: \mathbf{h}_v^k
e.g. the 0-th layer may contain the input: $\mathbf{h}_v^0 = \mathbf{x}_v$
- We aggregate \mathbf{h}_v^k over \mathbf{h}_v^{k-1} with its neighbors $\mathbf{h}_u^{k-1} \forall u \in N(v)$

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

[3]: Wolterink, Jelmer M., Tim Leiner, and Ivana Išgum. "Graph convolutional networks for coronary artery segmentation in cardiac CT angiography." *International Workshop on Graph Learning in Medical Imaging*. Springer, Cham, 2019.

Graph Deep Learning

GraphSAGE [2] - The Algorithm

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K ; weight matrices $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity σ ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output: Vector representations \mathbf{z}_v for all $v \in \mathcal{V}$

```

1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$  ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma \left( \mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k) \right)$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 

```

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

Graph Deep Learning

GraphSAGE [2] - Aggregators

- Mean Aggregator:

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}))$$

- GCN Aggregator
- Pooling Aggregator

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma(\mathbf{W}_{\text{pool}} \mathbf{h}_{u_i}^k + \mathbf{b}), \forall u_i \in \mathcal{N}(v)\}).$$

- LSTM Aggregator

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

Graph Deep Learning

^[4]

GNN

MPNN

MoNet

GraphSAGE

PATCHY-SAN

GAT

DGCNN

FastGCN

SumPooling

AvgPooling

GGNN

DGI

CGMM

GraphESN

SortPooling

SSE

DiffPool

GAAN

Huang et al.

PGC-DGCNN

SpectralCNN

NN4G

AGCN

GeniePath

ChebNet

DualGCN

MaxPooling

Henaff et al.

ClusterGCN

FastGCN

CayleyNet

GlobalAttentionPooling

LGCN

GCN

StoGCN

DCNN

Set2Set

SortPooling

[4]: Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *arXiv preprint arXiv:1901.00596* (2019).

References

- [1]: Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
- [2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.
- [3]: Wolterink, Jelmer M., Tim Leiner, and Ivana Išgum. "Graph convolutional networks for coronary artery segmentation in cardiac CT angiography." *International Workshop on Graph Learning in Medical Imaging*. Springer, Cham, 2019.
- [4]: Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *arXiv preprint arXiv:1901.00596* (2019).
- [5]: Bronstein, Michael et al. Lecture “Geometric deep learning on graphs and manifolds” held at SIAM Tutorial Portland (2018)

Image References

- [a] <https://de.serlo.org/mathe/funktionen/funktionsbegriff/funktionen-graphen/graph-funktion>
- [b] https://www.nwrfc.noaa.gov/snow/plot_SWE.php?id=AFSW1
- [c] <https://tennisbeiolympia.wordpress.com/meilensteine/steffi-graf/>
- [d] <https://www.pinterest.de/pin/624381935818627852/>
- [e] <https://www.uhere.com/free-cliparts/the-pentagon-pentagram-symbol-regular-polygon-golden-five-pointed-star-2282605>
- [f] <http://geometricdeeplearning.com/> (Geometric Deep Learning on Graphs and Manifolds)
- [g] <https://i.stack.imgur.com/NU7y2.png>
- [h] [https://de.wikipedia.org/wiki/Datei:Convolution_Animation_\(Gaussian\).gif](https://de.wikipedia.org/wiki/Datei:Convolution_Animation_(Gaussian).gif)
- [i] <https://www.researchgate.net/publication/306293638/figure/fig1/AS:396934507450372@1471647969381/Example-of-centerline- extracted-left-and-coronary-artery-tree-mesh-reconstruction.png>
- [j] https://www.eurorad.org/sites/default/files/styles/figure_image_teaser_large/public/figure_image/2018-08/0000015888/000006.jpg?itok=hwX1sbCO



Known Operator Learning - Towards Integration of Prior Knowledge into Machine Learning

Andreas Maier
Katharina Breininger

Lehrstuhl für Mustererkennung (Informatik 5),
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany



European
Research
Council

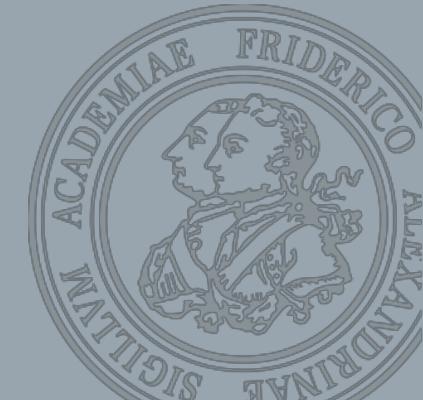


Deutsche
Forschungsgemeinschaft



European
Commission

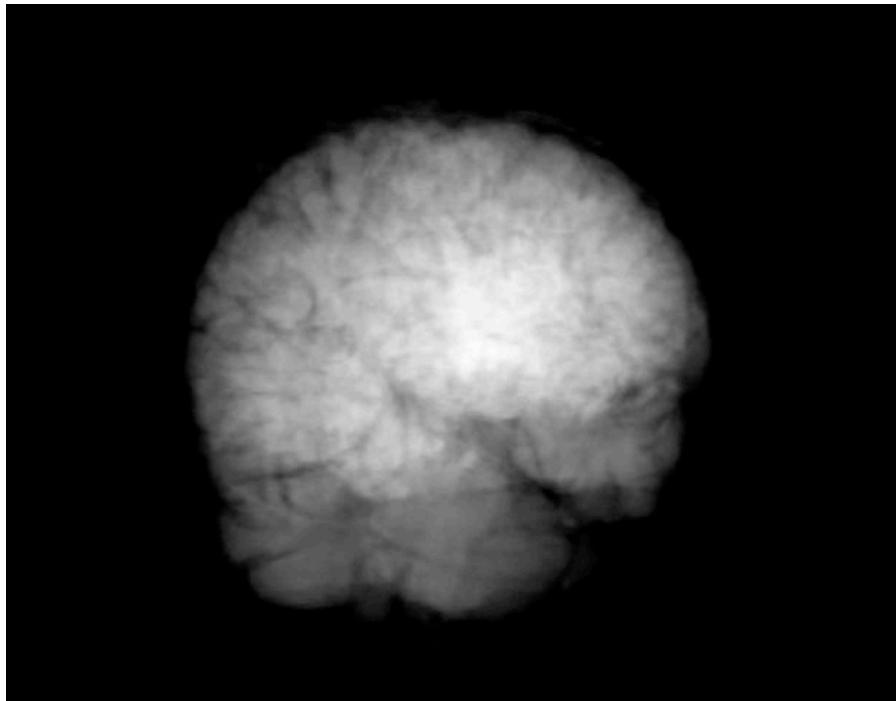
Horizon 2020
European Union funding
for Research & Innovation



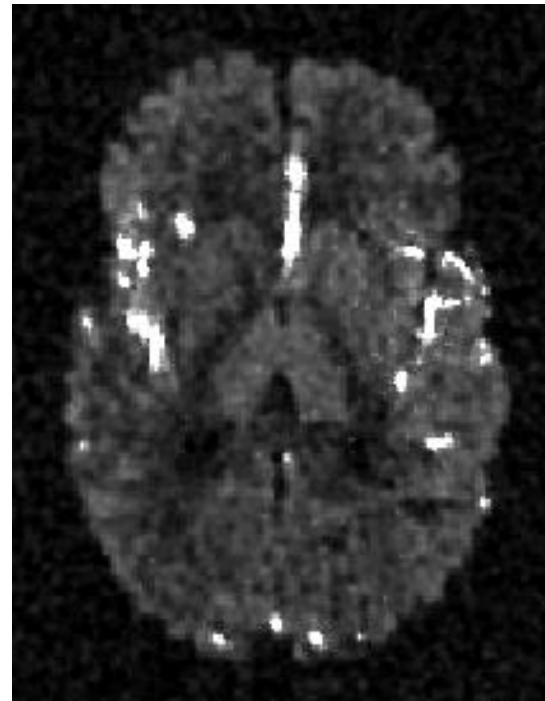
Known Operator Learning

- Motivation
- Known Operators in Deep Networks
- Examples in Computed Tomography
- Future Work

Image Reconstruction

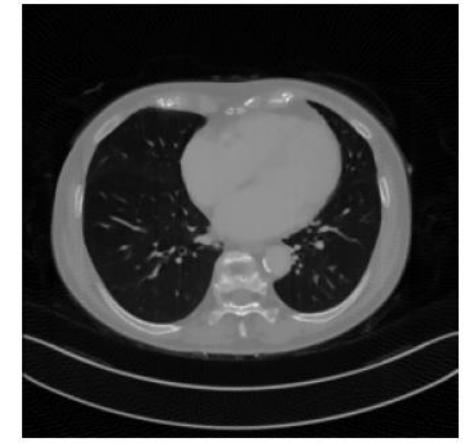
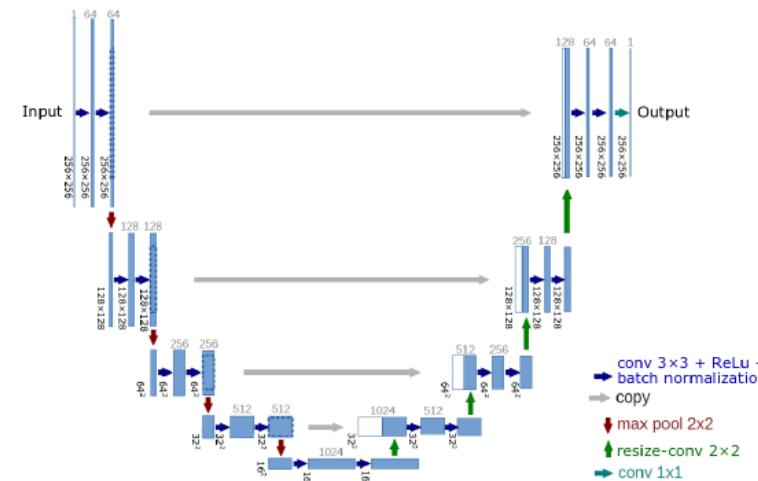
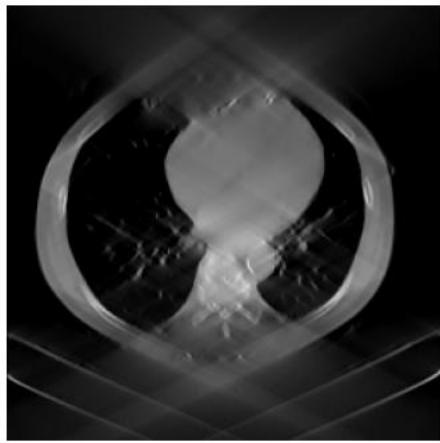


Projection Data



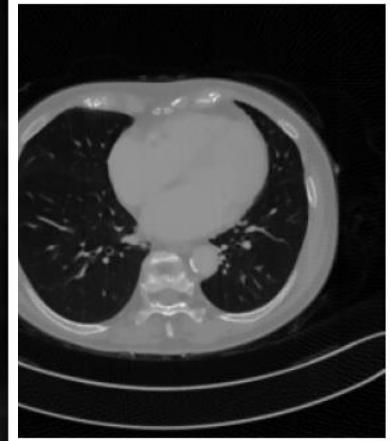
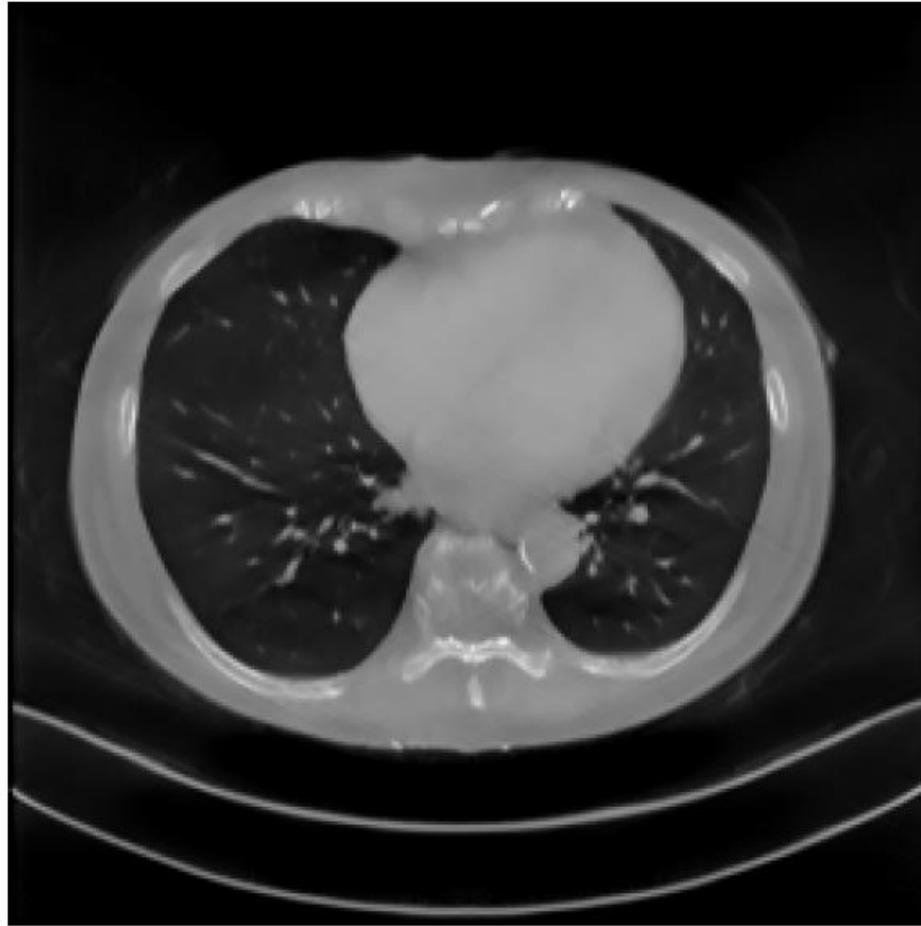
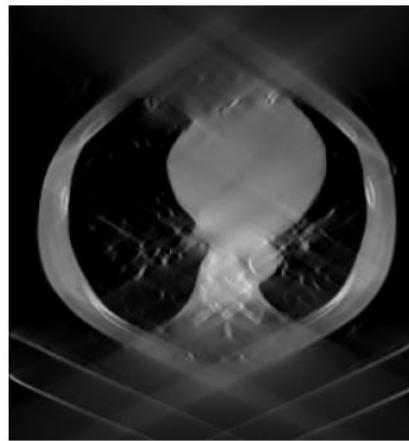
Sliced Volume

Deep Learning in Image Reconstruction?



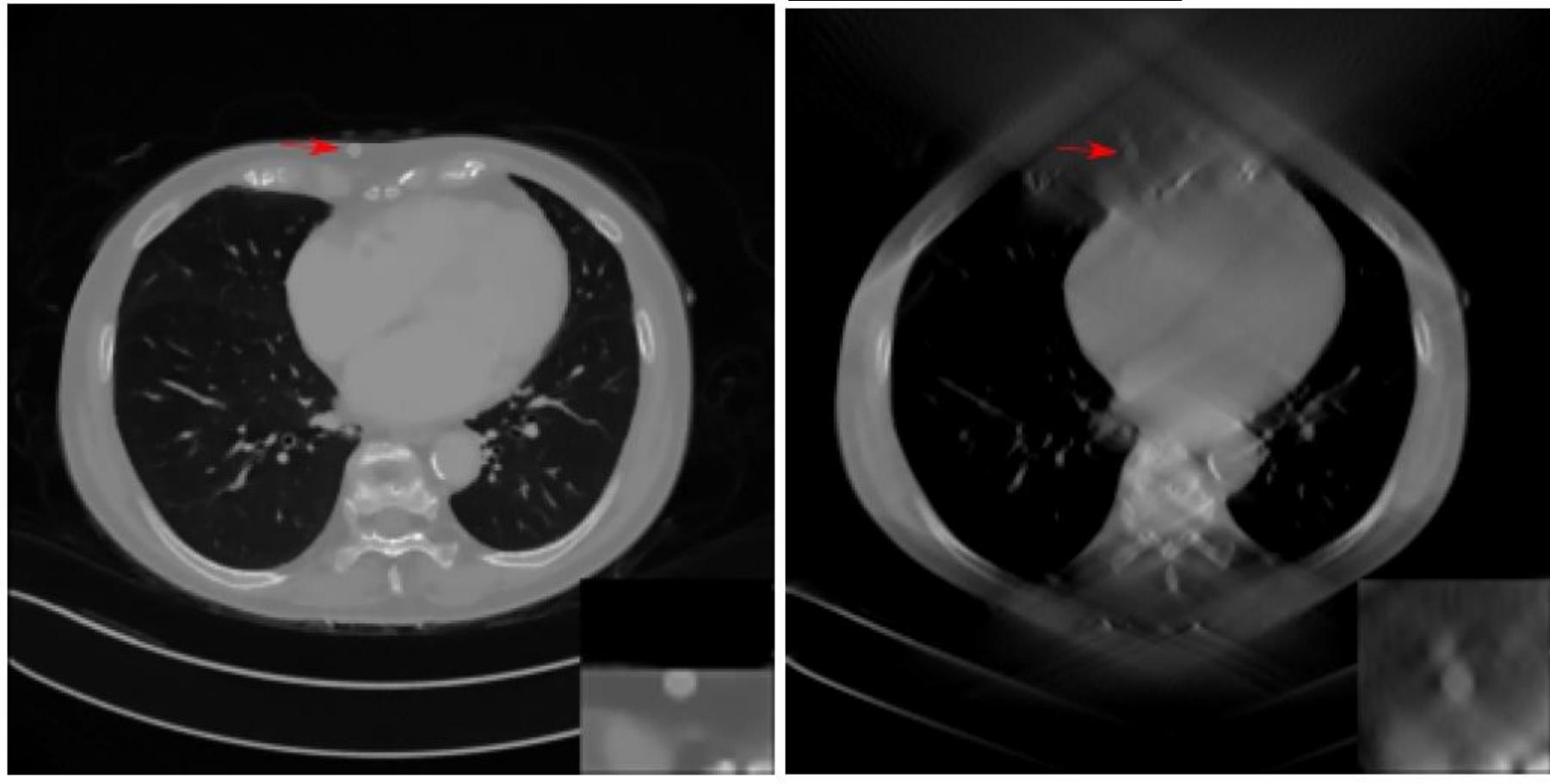
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



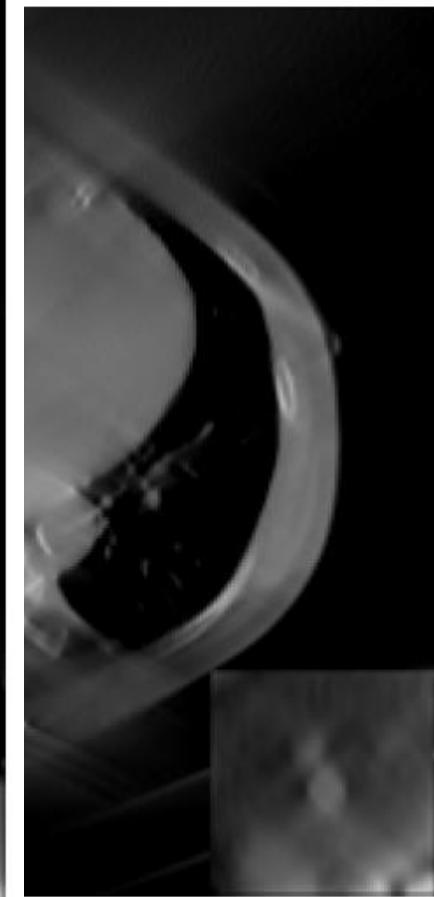
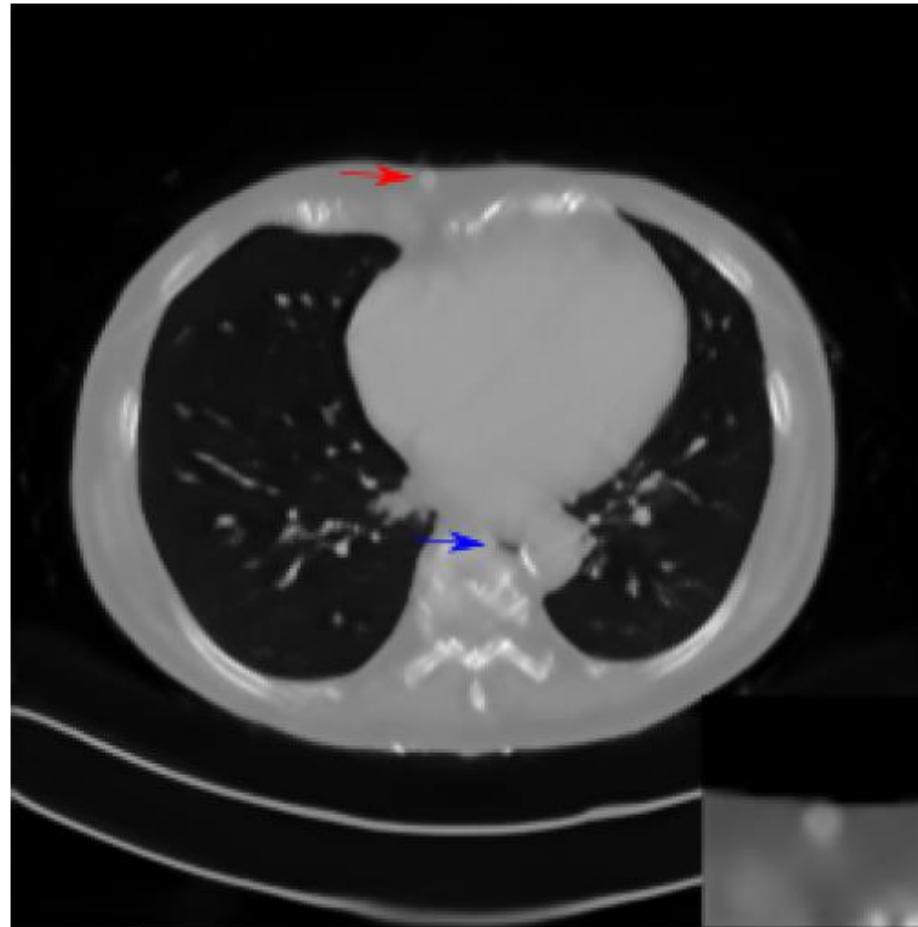
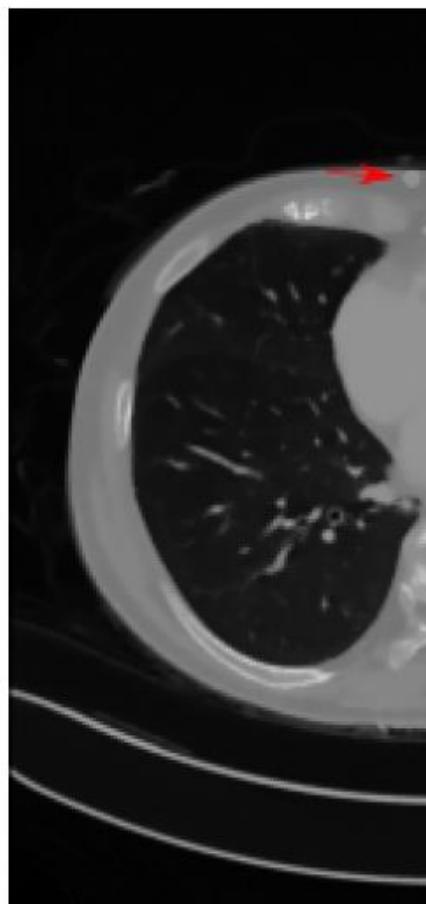
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



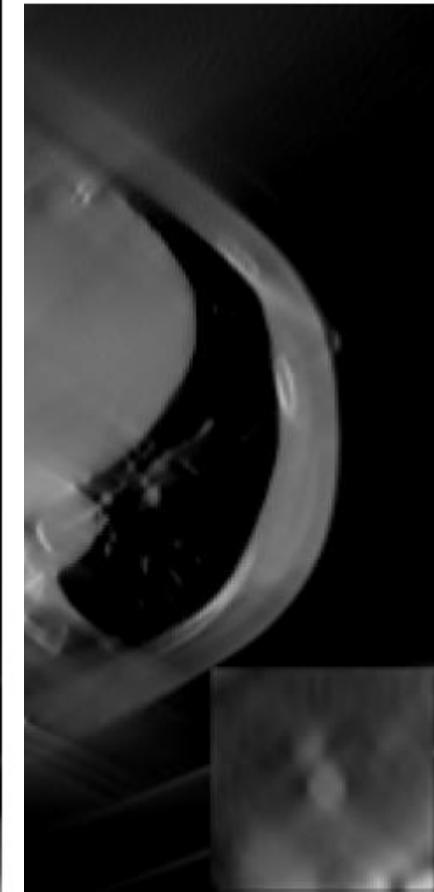
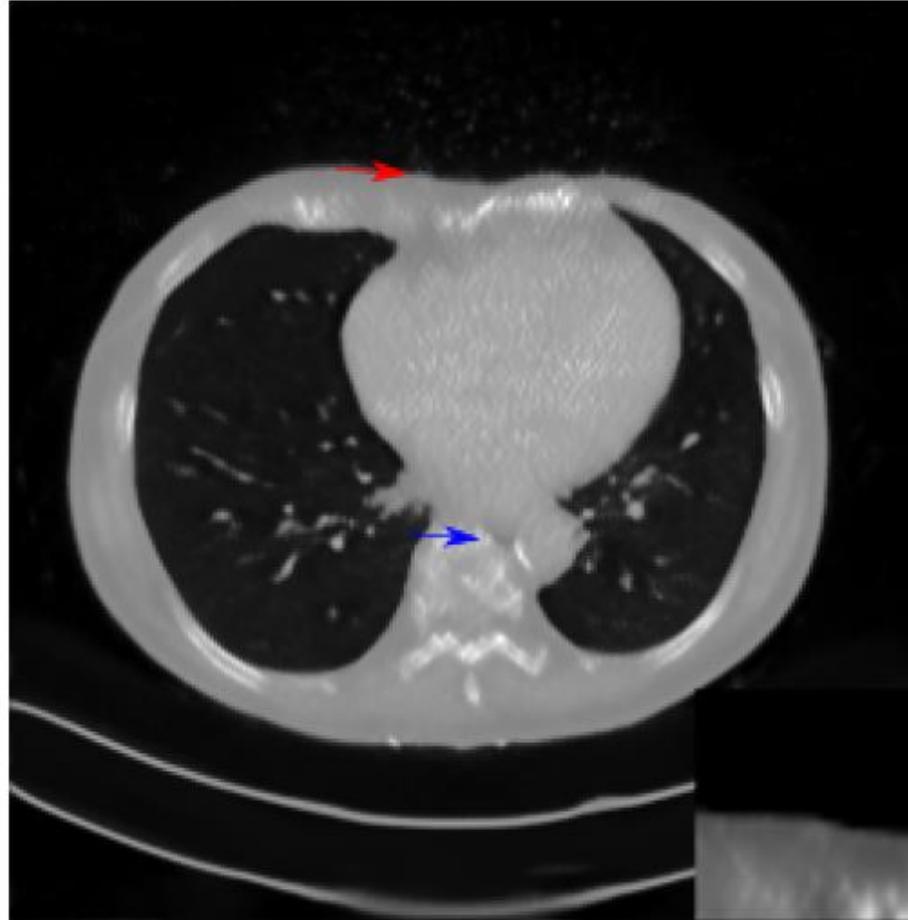
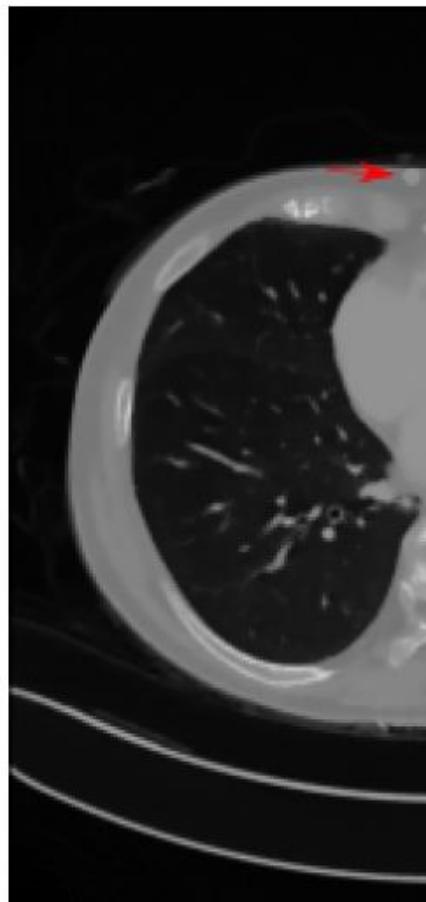
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



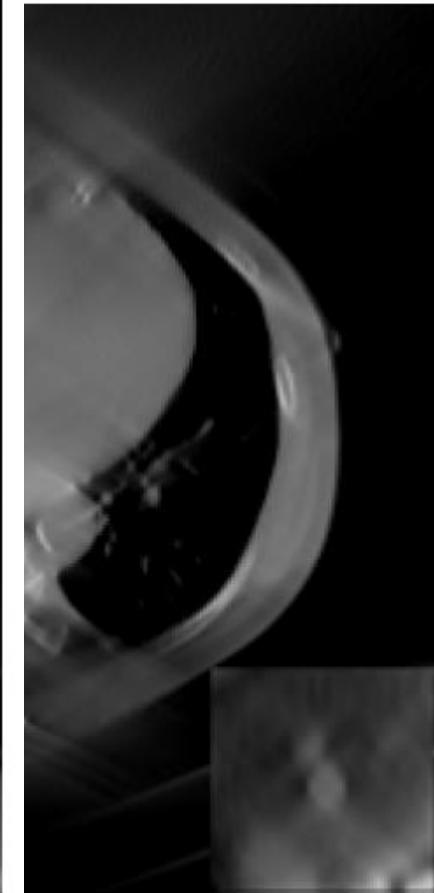
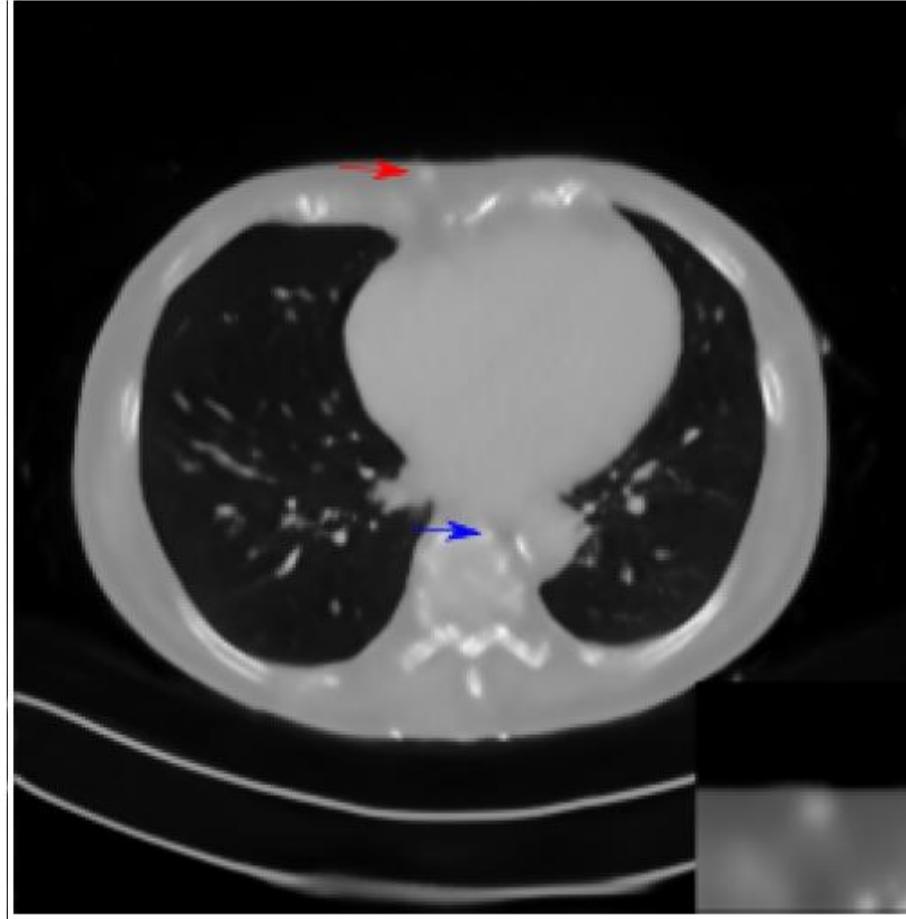
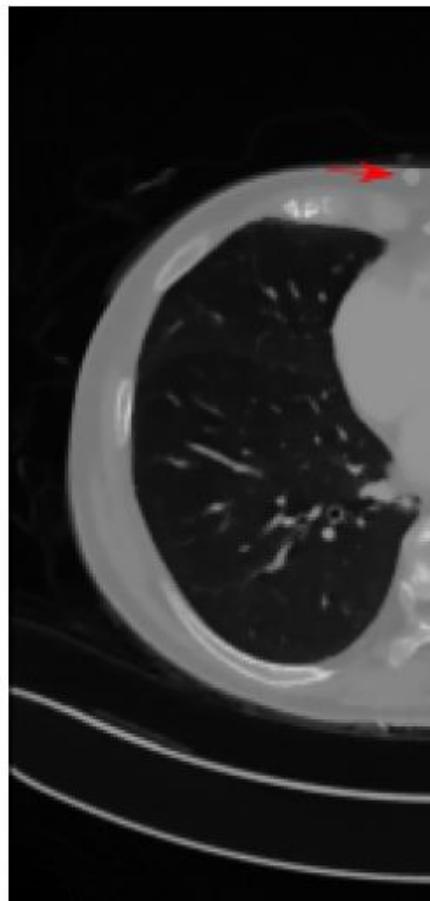
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



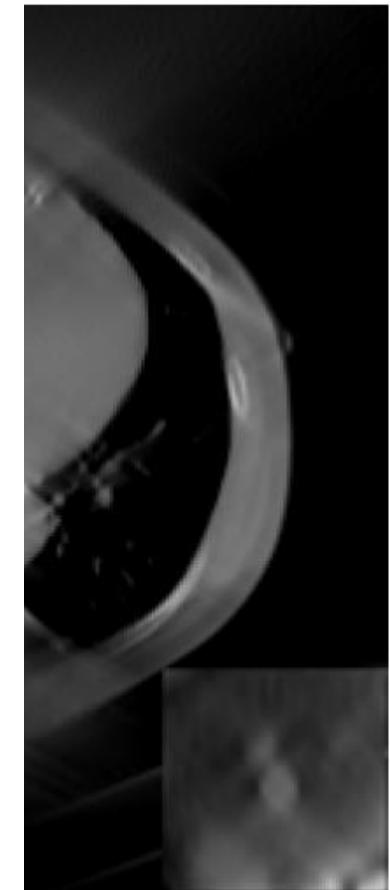
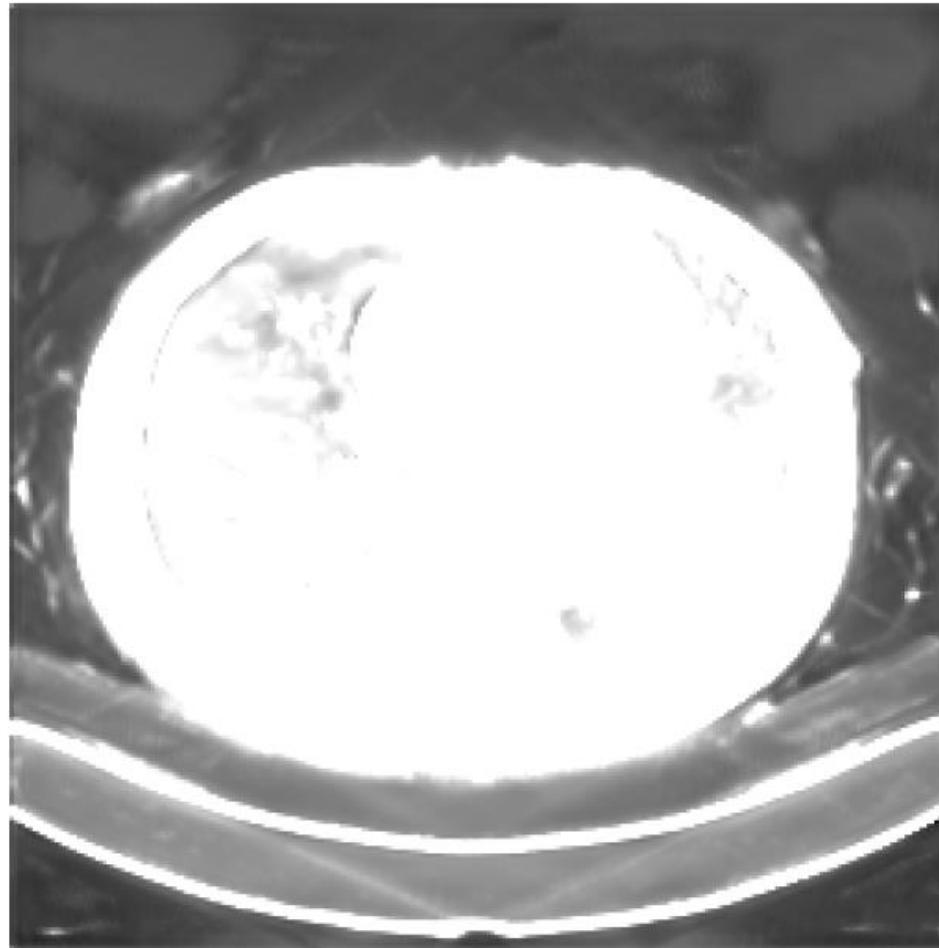
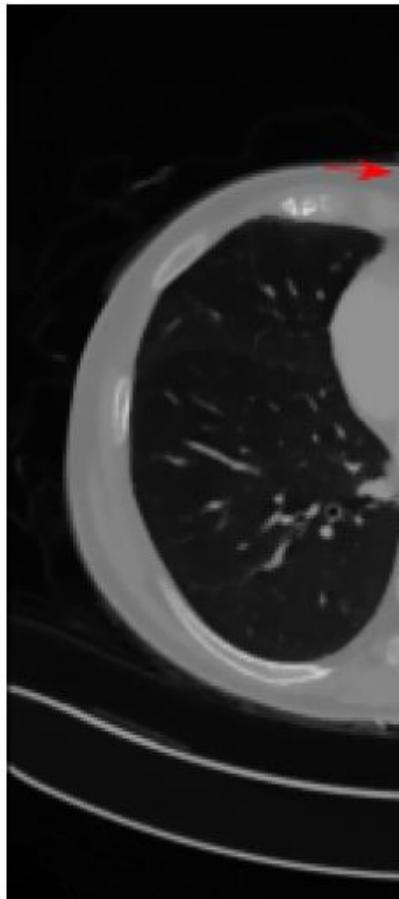
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Known Operators in Neural Networks

"Let's not reinvent the wheel..."

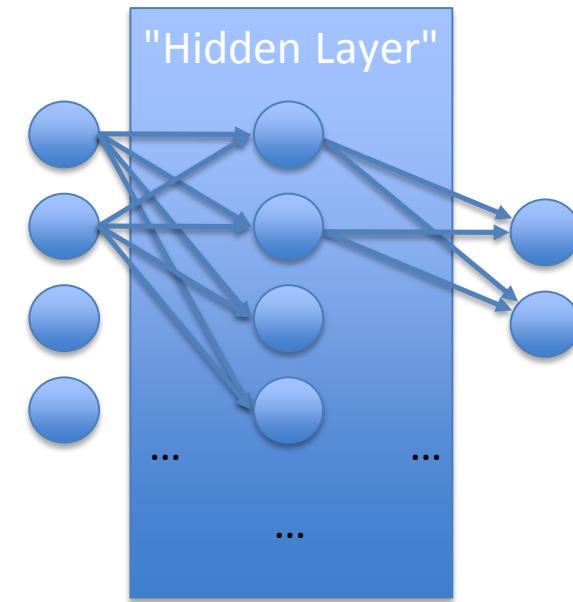
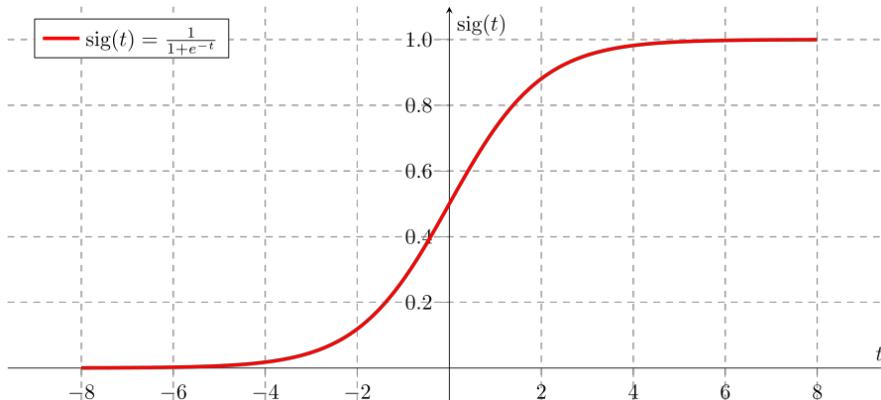
Universal Approximation Theorem

- Any continuous function can be approximated by Neural Net

$$u(\mathbf{x}) \approx U(\mathbf{x}) = \sum_i u_i s(\mathbf{w}_i^\top \mathbf{x} + w_{j,0})$$

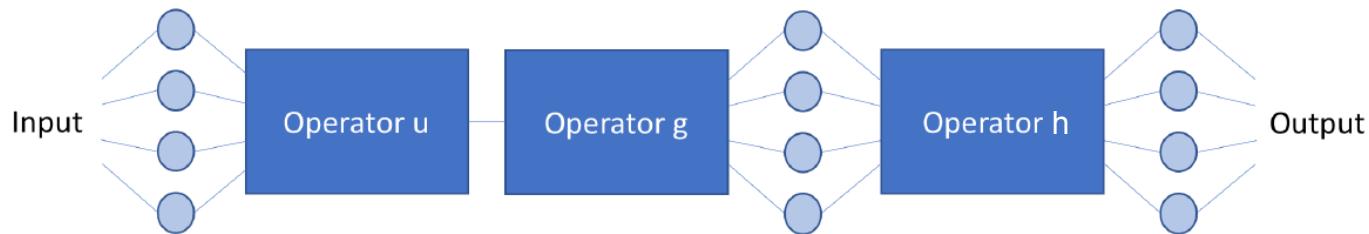
- The error is bound by

$$|U(\mathbf{x}) - u(\mathbf{x})| \leq \epsilon_u$$



Known Operators – Precision Learning

- Consider the case of using known operators in the net



- Specifically consider the use of two operators in sequence

$$f(\mathbf{x}) = g(\mathbf{u}(\mathbf{x}))$$

[5] Andreas Maier et al. Precision Learning: Towards use of known operators in neural networks. ICPR 2018.

Approximation Sequences

- Sequential operations

$$f(\mathbf{x}) = g(\mathbf{u}(\mathbf{x}))$$

- Can be approximated:

$$F_u(\mathbf{x}) = g(\mathbf{U}(\mathbf{x})) = f(\mathbf{x}) - e_u$$

$$F_g(\mathbf{x}) = G(\mathbf{u}(\mathbf{x})) = f(\mathbf{x}) - e_g$$

$$F(\mathbf{x}) = G(\mathbf{U}(\mathbf{x})) = f(\mathbf{x}) - e_f$$

Error of Approximation Sequences

- **Approximation introduces error**

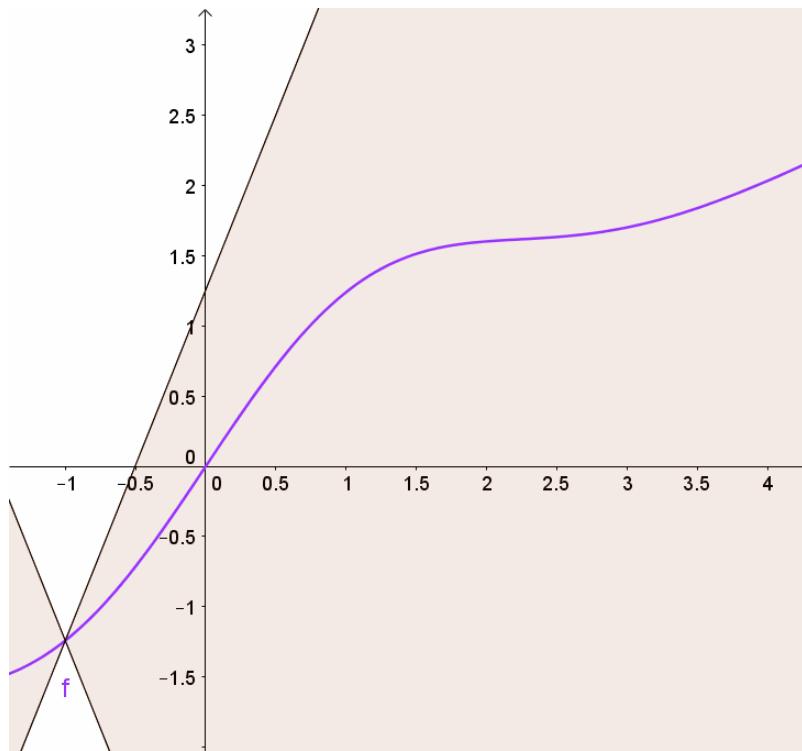
$$\begin{aligned}f(\mathbf{x}) &= g(\mathbf{u}(\mathbf{x})) = G(\mathbf{u}(\mathbf{x})) + e_g \\&= \sum_j g_j s(u_j(\mathbf{x})) + g_0 + e_g \\&= \sum_j g_j s(U_j(\mathbf{x}) + e_{u_j}) + g_0 + e_g\end{aligned}$$

- **Can we find bounds on this error?**

Bounds for Sigmoid Functions

- Sigmoid function satisfies the following upper bound:

$$s(x + e) \leq s(x) + l_s \cdot |e|$$



https://en.wikipedia.org/wiki/Lipschitz_continuity

Bounds for Sigmoid Functions

- Sigmoid function satisfies the following upper bound:

$$s(x + e) \leq s(x) + l_s \cdot |e|$$

- However this does not hold for linear combinations
- Alternate formulation:

$$g_j s(x + e) \leq g_j s(x) + |g_j| \cdot l_s \cdot |e|$$

Error of Approximation Sequences (2)

- **Recall**

$$\begin{aligned}
 f(\mathbf{x}) &= \sum_j g_j s(U_j(\mathbf{x}) + e_{u_j}) + g_0 + e_g \leq \underbrace{\sum_j g_j s(U_j(\mathbf{x})) + g_0}_{F(\mathbf{x})} + \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g \\
 &\leq F(\mathbf{x}) + \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g.
 \end{aligned}$$

- **Subtraction of $F(\mathbf{x})$ yields**

$$\begin{aligned}
 \underbrace{f(\mathbf{x}) - F(\mathbf{x})}_{e_f} &\leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g \\
 e_f &\leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g
 \end{aligned}$$

- **Which is bounded by**

$$e_f \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g$$

Error of Approximation Sequences (3)

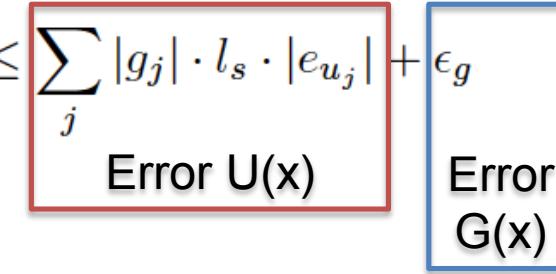
- Same idea can also be used for lower bound

$$e_f \geq - \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| - \epsilon_g$$

- Thus a general bound is

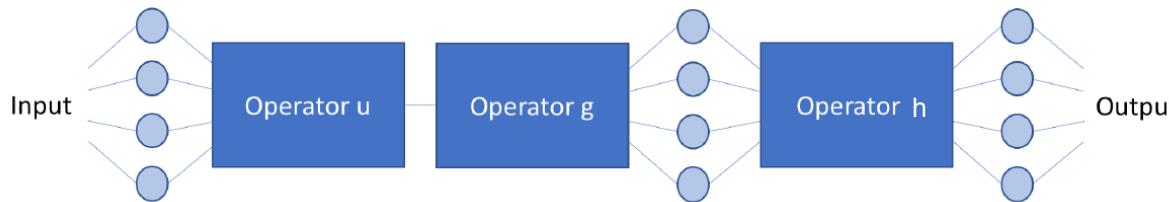
$$|e_f| \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + \epsilon_g$$

Observations on Bounds

- **Bound on Error:** $|e_f| \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + \epsilon_g$

- **Observations**
 - **Error in $U(x)$ and $G(x)$ additive**
 - **Error of $U(x)$ amplified by $g(x)$**
 - **Interpretation as Feature Extractor => Importance of Features**
 - **Requires Lipschitz continuity**

Observations on Bounds

- Extension to Deep Networks:



- Proof by Recursion:



Learning with known operators reduces maximum error bounds

Andreas K. Maier^{1*}, Christopher Syben¹, Bernhard Stimpel¹, Tobias Würfl¹, Mathis Hoffmann¹, Frank Schebesch¹, Weilin Fu¹, Leonid Mill¹, Lasse Kling^{1,2} and Silke Christiansen^{1,2,3}

We describe an approach for incorporating prior knowledge into machine learning algorithms. We aim at applications in physics and signal processing in which we know that certain operations must be embedded into the algorithm. Any operation that allows computation of a gradient or sub-gradient towards its inputs is suited for our framework. We derive a maximal error bound for deep nets that demonstrates that inclusion of prior knowledge results in its reduction. Furthermore, we show experimentally that known operators reduce the number of free parameters. We apply this approach to various tasks ranging from computed tomography image reconstruction over vessel segmentation to the derivation of previously unknown imaging algorithms. As such, the concept is widely applicable for many researchers in physics, imaging and signal processing. We assume that our analysis will support further investigation of known operators in other fields of physics, imaging and signal processing.

Computed Tomography

- Efficient solution via filtered back-projection:

$$f(x, y) = \int_0^{\pi} p(s, \theta) * h(s) \Big|_{s=x \cos \theta + y \sin \theta} d\theta$$

- Three steps:
 - Convolution along s
 - Back-projection along θ
 - Suppress negative values

Computed Tomography

- Efficient solution via filtered back-projection (FBP):

$$f(x, y) = \int_0^{\pi} p(s, \theta) * h(s) \Big|_{s=x \cos \theta + y \sin \theta} d\theta$$

In the continuous case: $h(s)$ in frequency domain = „ramp“

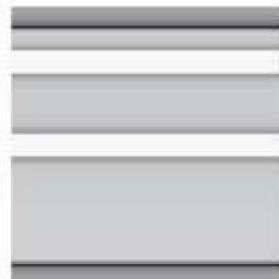
- Can also be derived in matrix notation:

$$\mathbf{Ax} = \mathbf{p}$$
$$\mathbf{x} = \mathbf{A}^+ \mathbf{p} = \mathbf{A}^T \underbrace{(\mathbf{A}\mathbf{A}^T)^+}_{\text{Filter}} \mathbf{p}$$

Computed Tomography

- Efficient solution via filtered back-projection (FBP):

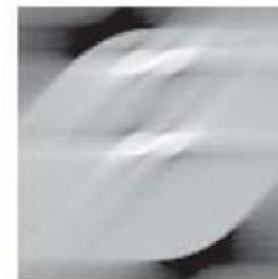
$$f(x, y) = \int_0^{\pi} p(s, \theta) * h(s) \Big|_{s=x \cos \theta + y \sin \theta} d\theta$$



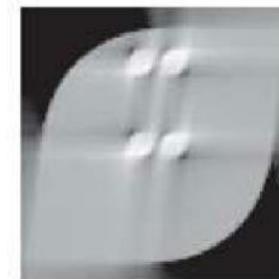
(A)



(B)



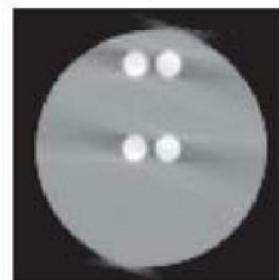
(C)



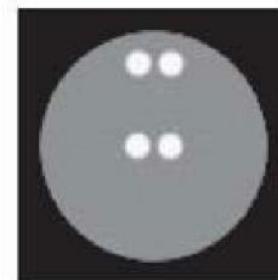
(D)



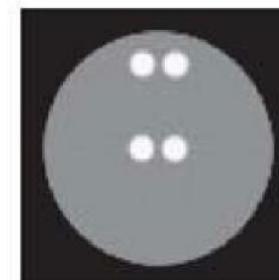
(E)



(F)



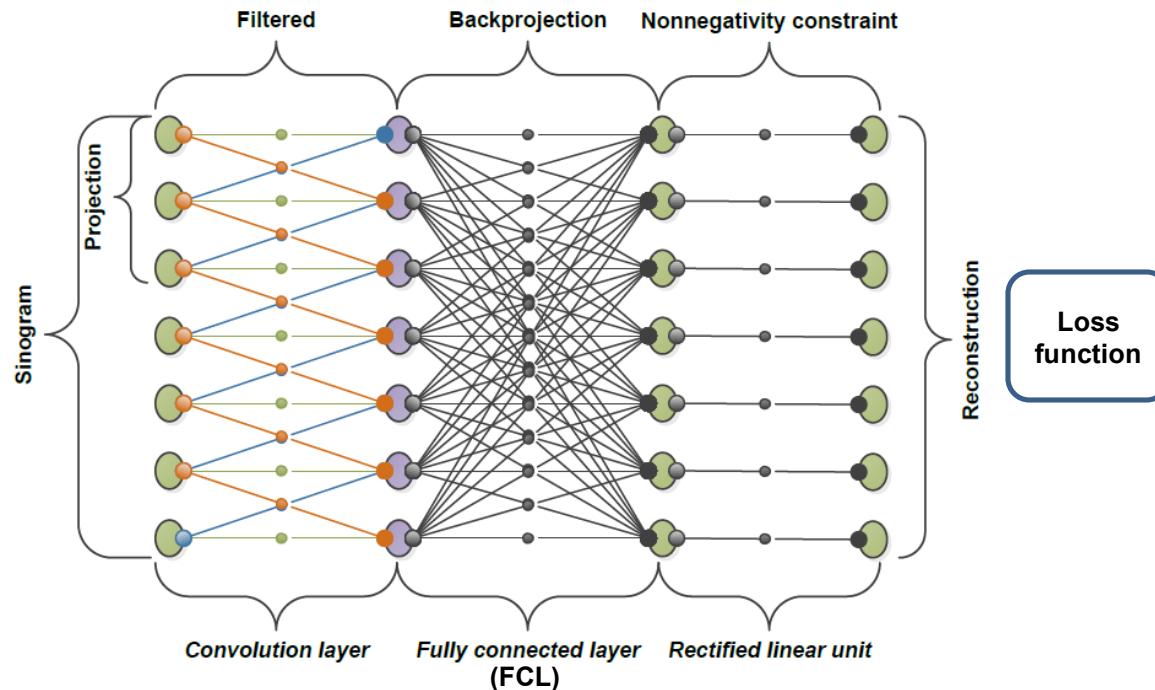
(G)



Original Image $f(x, y)$

Computed Tomography using Neural Networks

- All three steps can be modeled as neural network:



- Interesting: All weights are known from FBP

Discretization (2)

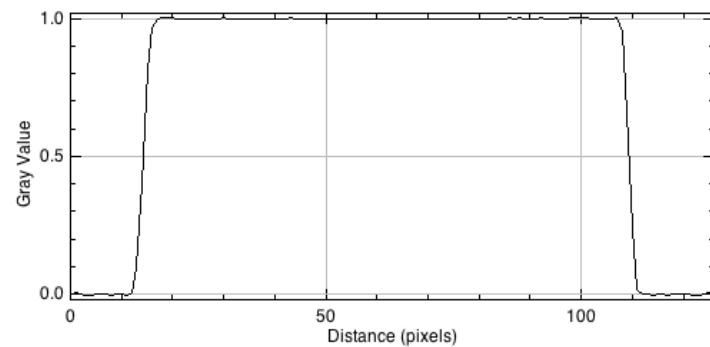
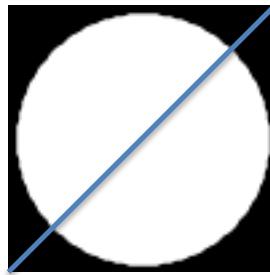
- Implementation by the book:



[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (2)

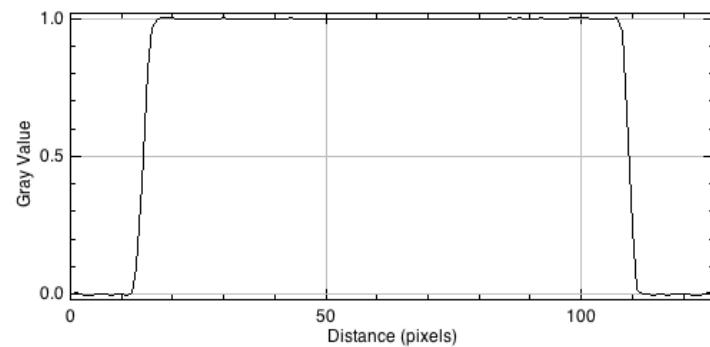
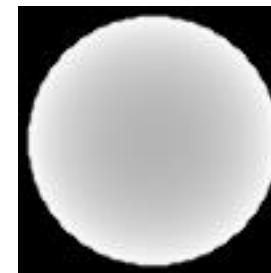
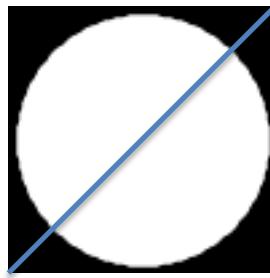
- Implementation by the book:



[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (2)

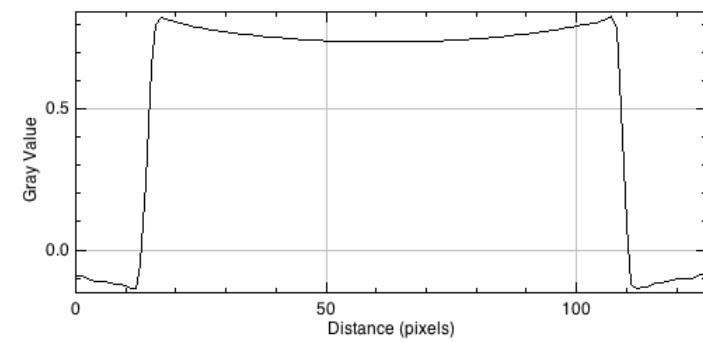
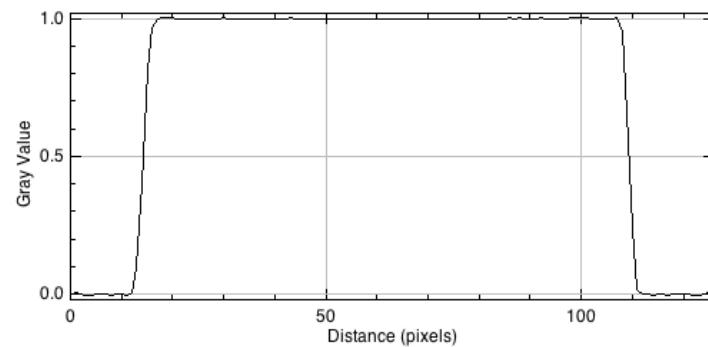
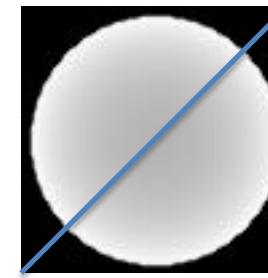
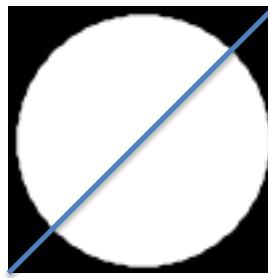
- Implementation by the book:



[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (2)

- Implementation by the book:



[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (3)

- Problem: „Ramp“-filter assumes infinite projections
→ discretization error
- Idea: Neural Networks are discrete end-to-end
→ Network should be able to learn correct solution
- Correct discretization is an intrinsic property of the net

$$\begin{aligned}\mathbf{x} &= \mathbf{A}^{\top}(\mathbf{A}\mathbf{A}^{\top})^{-1}\mathbf{p} \\ \mathbf{x} &= \mathbf{A}^{\top}\mathbf{F}^H\mathbf{K}\mathbf{F}\mathbf{p}\end{aligned}$$

Discretization (3)

- Problem: „Ramp“-filter assumes infinite projections
→ discretization error
- Idea: Neural Networks are discrete end-to-end
→ Network should be able to learn correct solution
- Correct discretization is an intrinsic property of the net

$$\begin{aligned} \mathbf{x} &= \mathbf{A}^{\top}(\mathbf{A}\mathbf{A}^{\top})^{-1}\mathbf{p} \\ \mathbf{x} &= \boxed{\mathbf{A}^{\top}\mathbf{F}^H\mathbf{K}\mathbf{F}\mathbf{p}} \\ &\quad \text{Net} \end{aligned}$$

Discretization (3)

- Problem: „Ramp“-filter assumes infinite projections
→ discretization error
- Idea: Neural Networks are discrete end-to-end
→ Network should be able to learn correct solution
- Correct discretization is an intrinsic property of the net

$$\begin{aligned}\mathbf{x} &= \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{p} \\ \mathbf{x} &= \mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p}\end{aligned}$$

- Associated optimization problem:

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top$$

Recon

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top$$

Error

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top$$

Error
Backpropagation

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \boxed{\mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top}$$

Backpropagation

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \boxed{\mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top}$$

Backpropagation „l-1“

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2} \|\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\|_2^2$$

- **Gradient:**

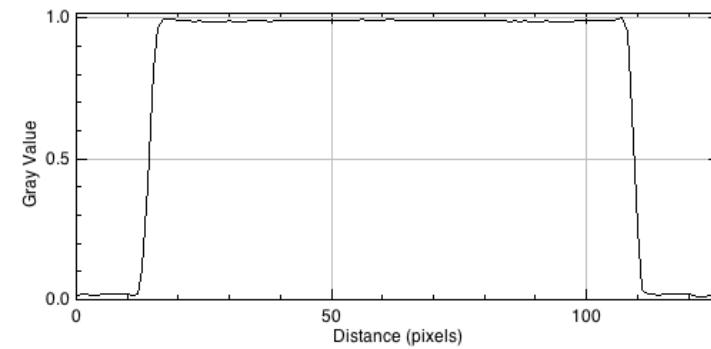
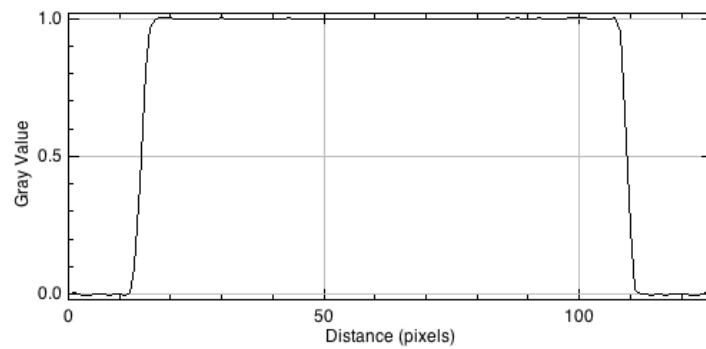
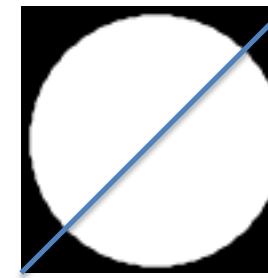
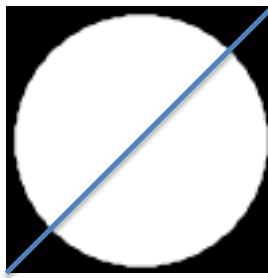
$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \boxed{\mathbf{F} \mathbf{A} (\mathbf{A}^\top \mathbf{F}^H \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}) (\mathbf{F} \mathbf{p})^\top}$$

Backpropagation
„l-1“

$$\Delta w_{ij}^{(l)} = \eta \delta_j^{(l)} y_i^{(l-1)}$$

Discretization (3)

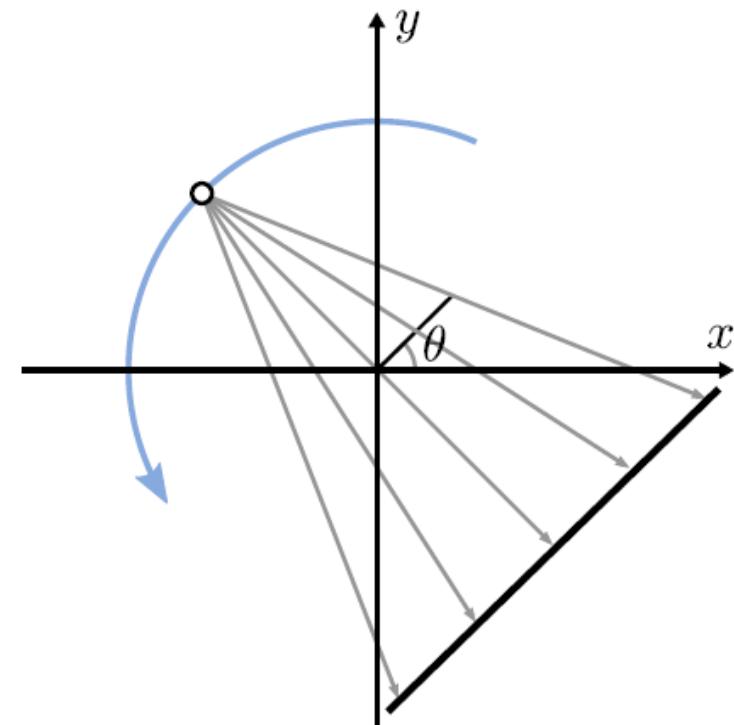
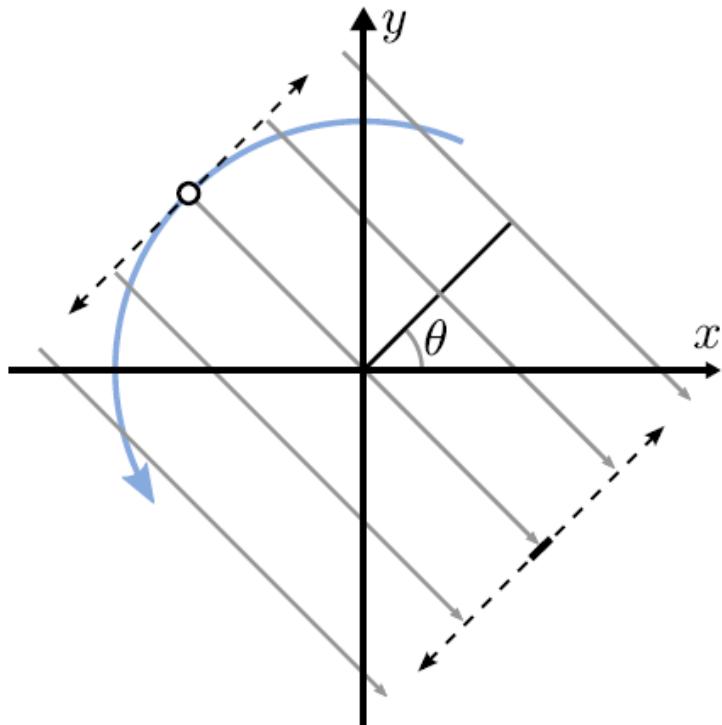
- Filter after "Learning":



[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Computed Tomography using Neural Networks

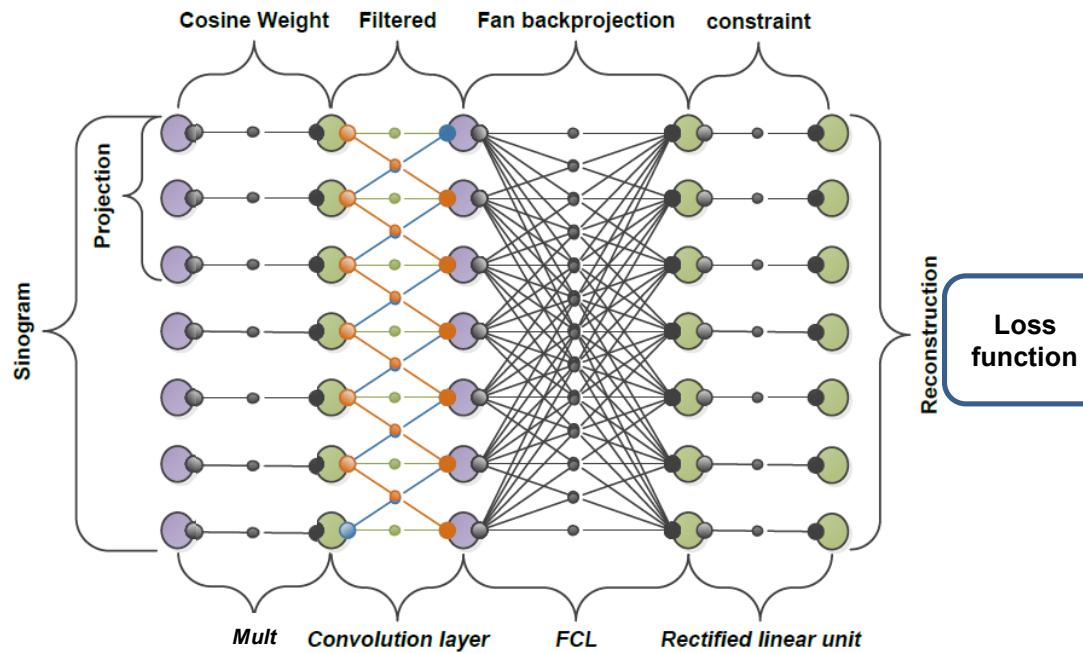
- Parallel vs. fan-beam reconstruction:



Computed Tomography using Neural Networks

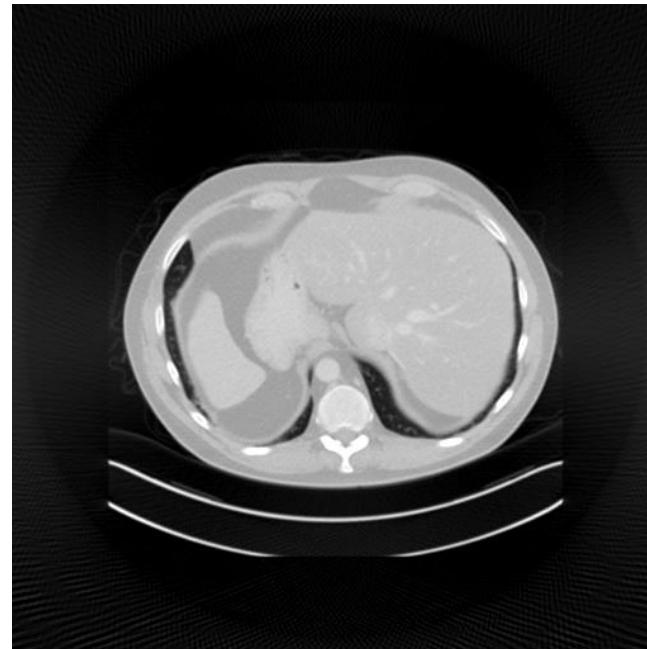
- Fan-beam reconstruction formula:

$$x = \mathbf{A}^\top \mathbf{C} \mathbf{W} \mathbf{p}$$



Computed Tomography using Neural Networks

- Application to incomplete scans [2]

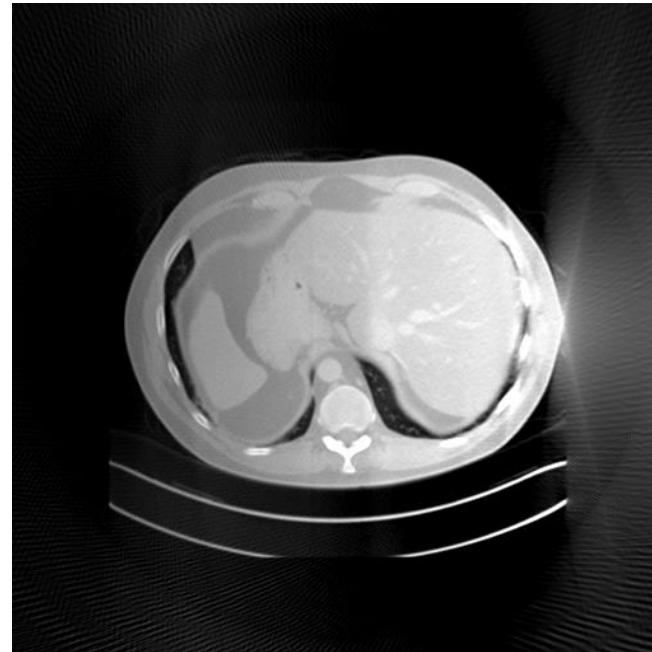


Reconstruction with 360 deg

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

Computed Tomography using Neural Networks

- Application to incomplete scans [2]

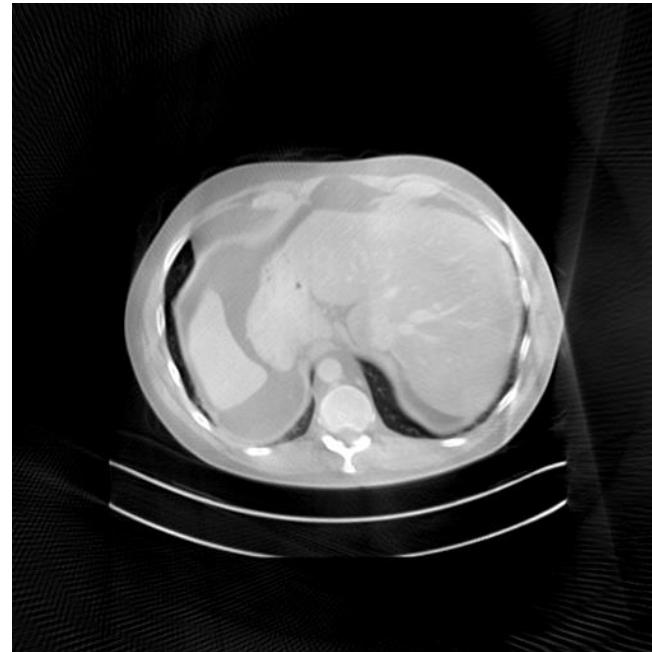


Reconstruction with 180 deg (FBP)

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

Computed Tomography using Neural Networks

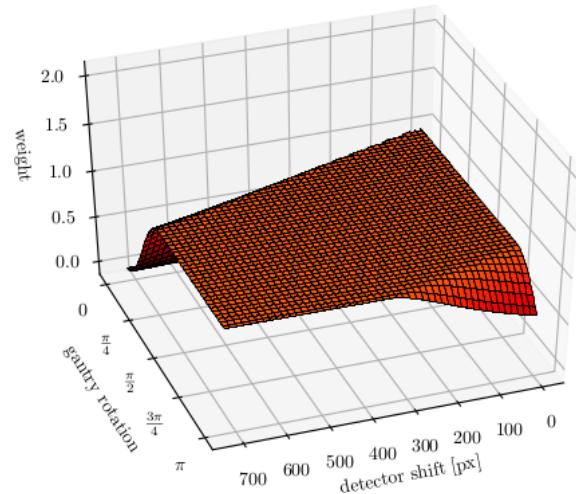
- Application to incomplete scans [2]



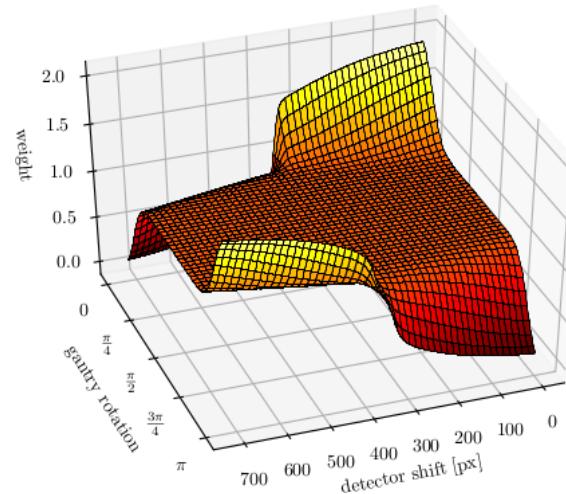
Reconstruction with 180 deg (NN)

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

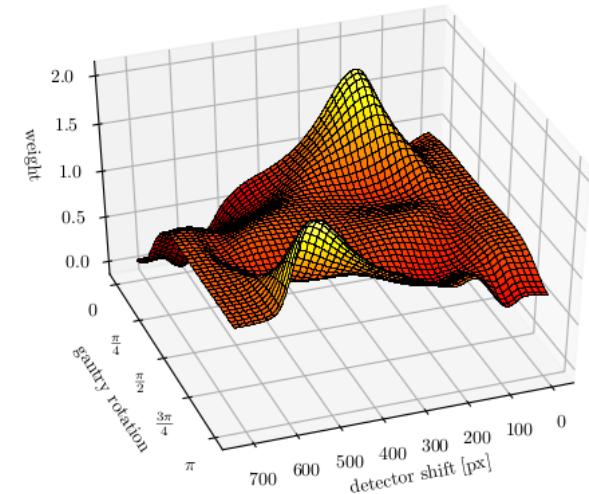
Learned Weights



Parker et al.
1982
„works well“



Schäfer et al.
2017
„works better“



Würfl et al.
2016
data optimal

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

Further Extensions

- Add non-linear de-streaking and de-noising step:

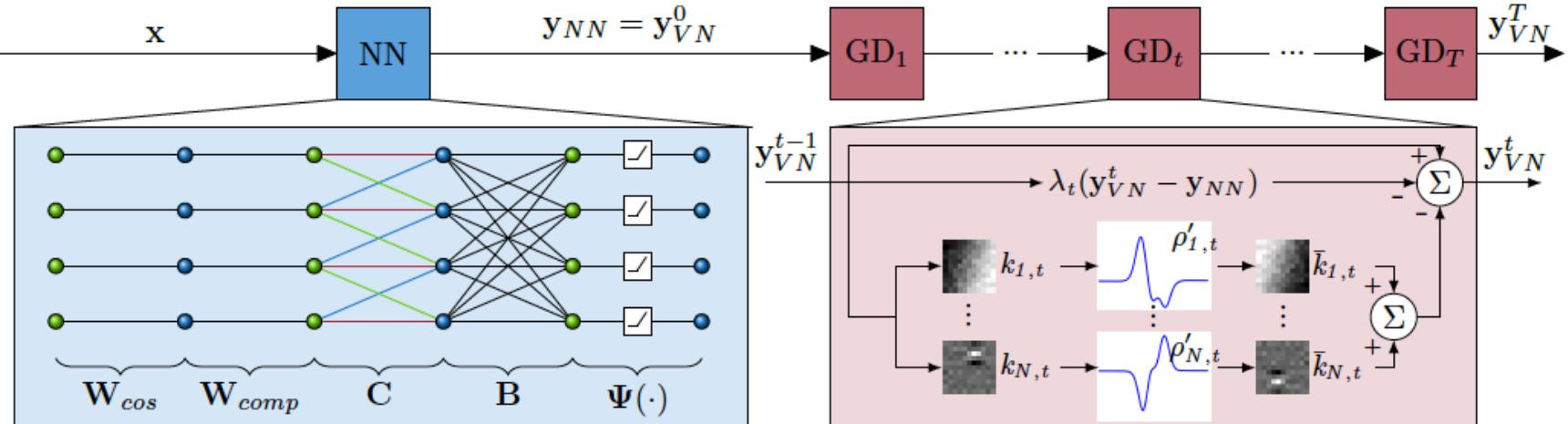
$$E(\mathbf{y}) = \frac{\lambda}{2} \|\mathbf{y}_{VN} - \mathbf{y}_{NN}\|_2^2 + \sum_{i=1}^{N_k} \rho_i(\mathbf{K}_i \mathbf{y}_{VN})$$

$$\mathbf{y}_{VN}^t = \mathbf{y}_{VN}^{t-1} - \sum_{i=1}^{N_k} \mathbf{K}_{i,t}^T \rho'_{i,t}(\mathbf{K}_{i,t} \mathbf{y}_{VN}^{t-1}) - \lambda_t (\mathbf{y}_{VN}^{t-1} - \mathbf{y}_{NN})$$

[7] Hammernik, Kerstin, et al. "A deep learning architecture for limited-angle computed tomography reconstruction." *Bildverarbeitung für die Medizin 2017*. Springer Vieweg, Berlin, Heidelberg, 2017. 92-97.

Further Extensions

- Add non-linear de-streaking and de-noising step:



[7] Hammernik, Kerstin, et al. "A deep learning architecture for limited-angle computed tomography reconstruction." *Bildverarbeitung für die Medizin 2017*. Springer Vieweg, Berlin, Heidelberg, 2017. 92-97.

Further Extensions

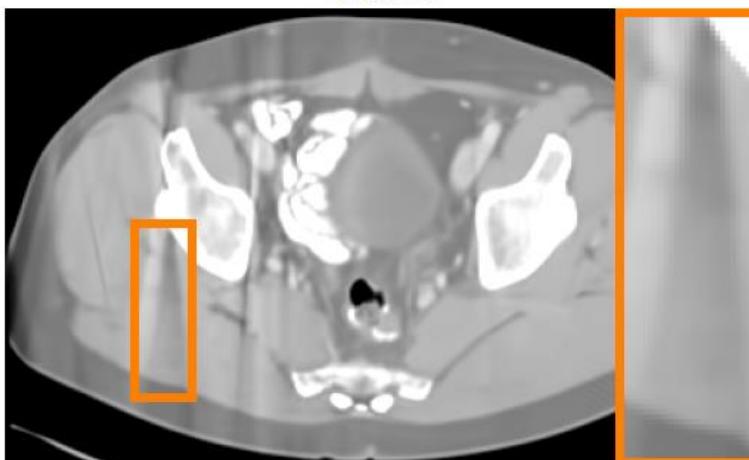
Full Scan Reference



Neural Network Input



BM3D



Variational Network ($k = 13$)



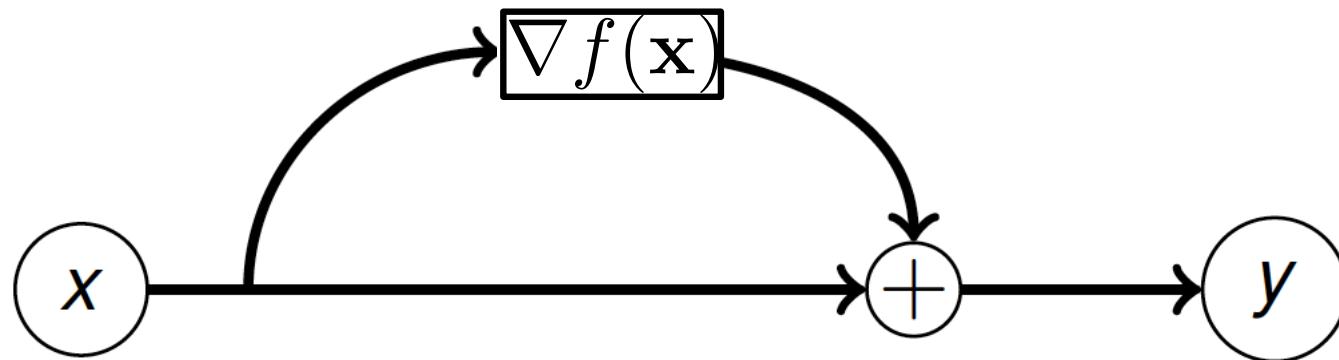
ResNets Revisited

General Function Optimization: Find maxima of

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

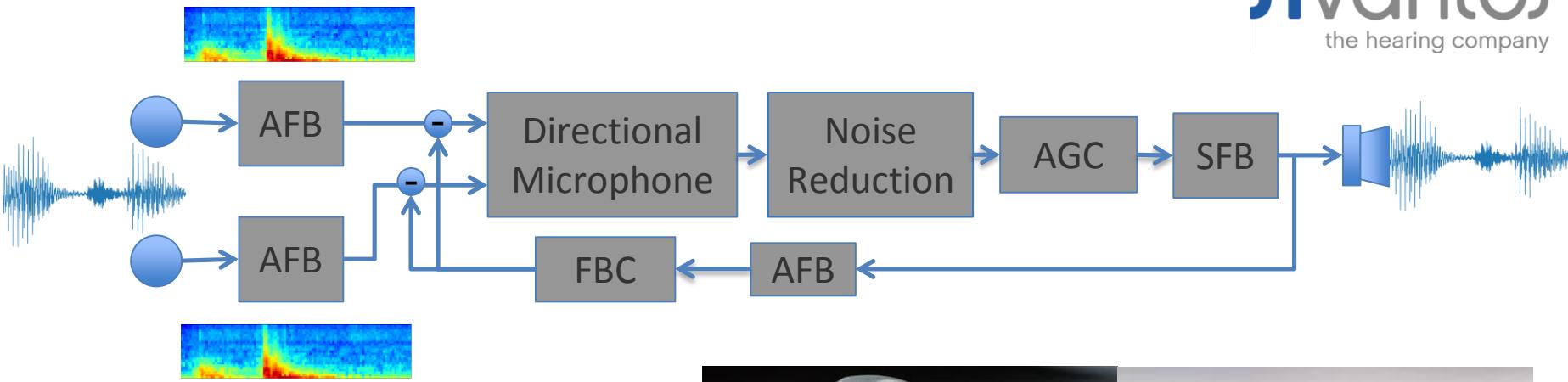
Idea: follow gradient direction

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \nabla f(\mathbf{x}_n)$$



Simplified Modern Hearing Instrument Pipeline

sivantos
the hearing company



AFB = Analysis Filterbank

AGC = Automatic Gain Control

SFB = Synthesis Filterbank

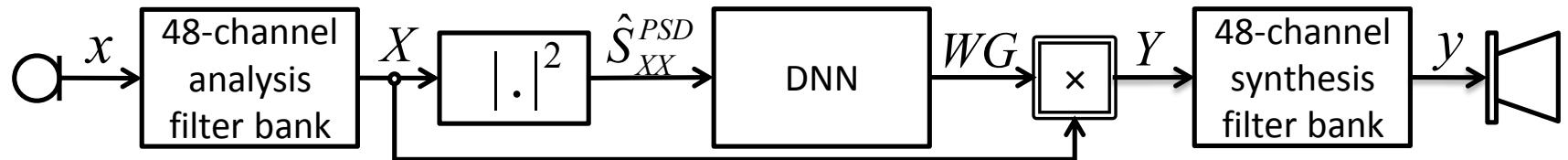
FBC = Feedback Canceler



Images from [Carat&Insio](#) (cropped)

[8] Aubreville, Marc, et al. "Deep Denoising for Hearing Aid Applications." 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC). IEEE, 2018.

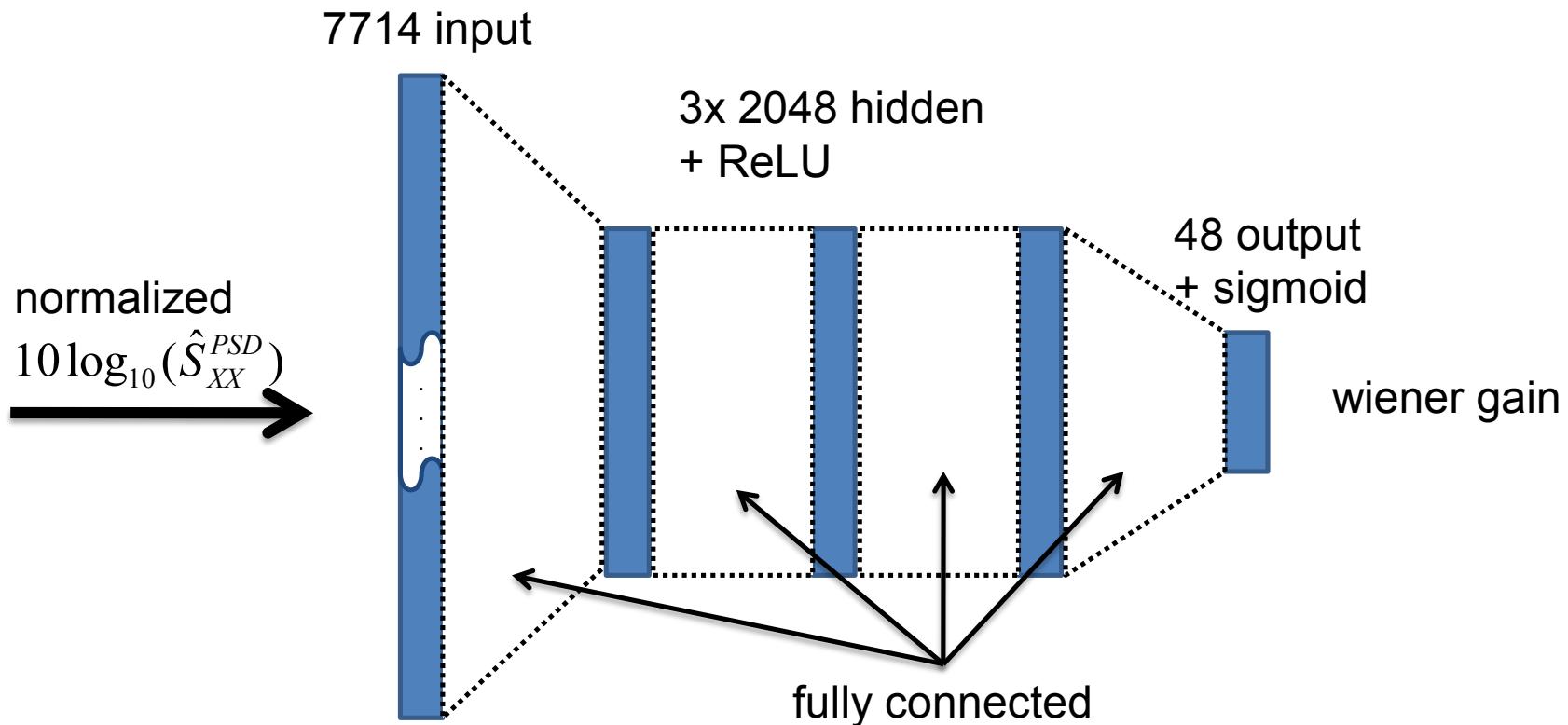
DL Hearing Instrument Pipeline



Dense Neural Network (DNN)

- **Network input:** $10 \log_{10}(\hat{S}_{XX}^{PSD})$
 - **200 frames from past**
 - **current frame**
 - **2 frames from future → delay of 2 ms + filter bank delay**
 - **normalized to zero-mean & unit variance frequency-bin-wise**
- **3 hidden layer /w 2048 hidden nodes + ReLU**
- **Network output:**
 - **current frame (=48 nodes) + Sigmoid**
 - **prediction of wiener gain**

$$WG = \frac{\hat{S}_{SS}^{PSD}}{\hat{S}_{SS}^{PSD} + \hat{S}_{NN}^{PSD}}$$



Used with permission from Kai Ehrensperger and Marc Aubreville

Dataset

Database ($fs=24\text{kHz}$):

259 clean speech signals:

expected duration: 21 s

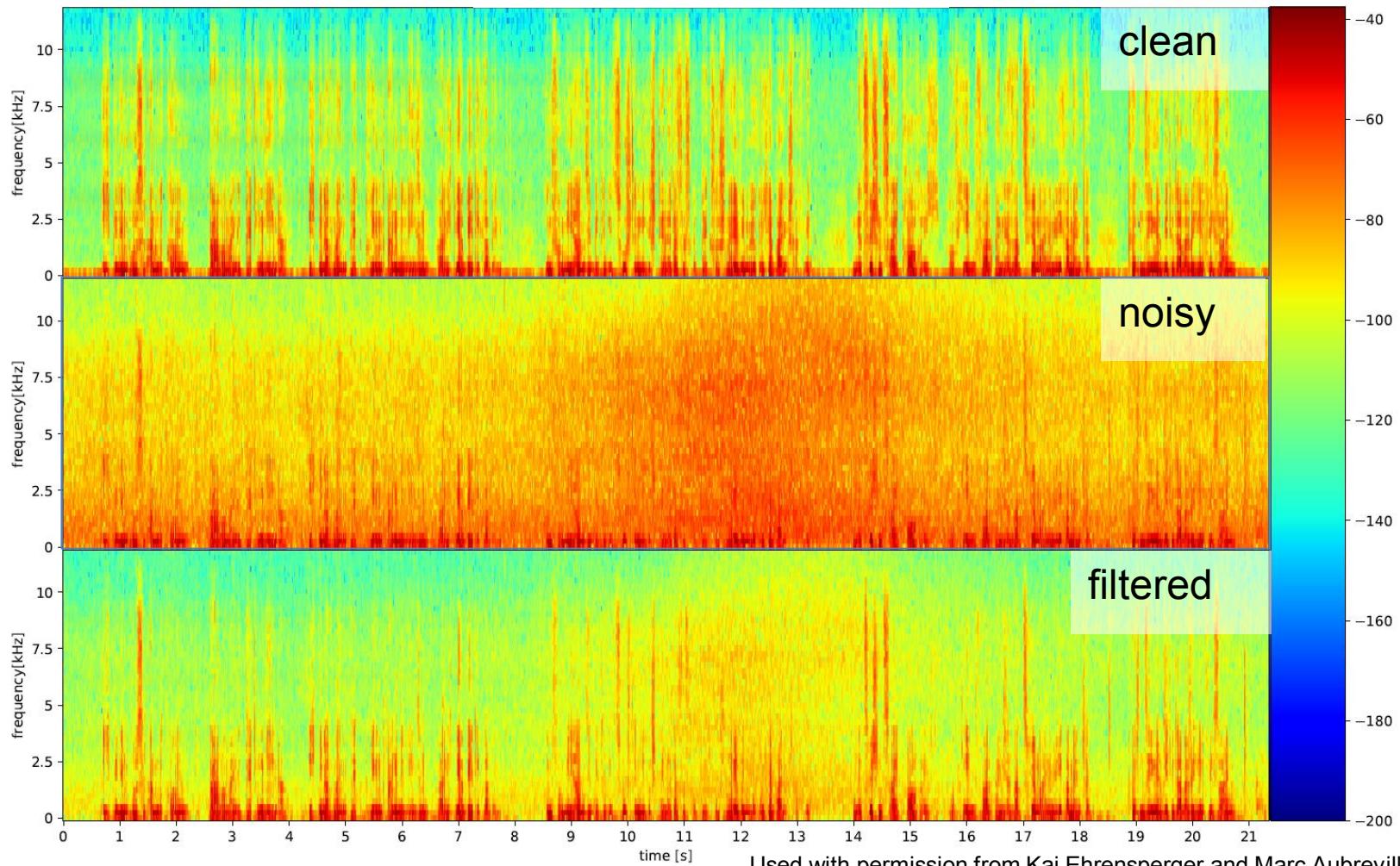
48 non-stationary noise signals

expected duration: 118 s



Results

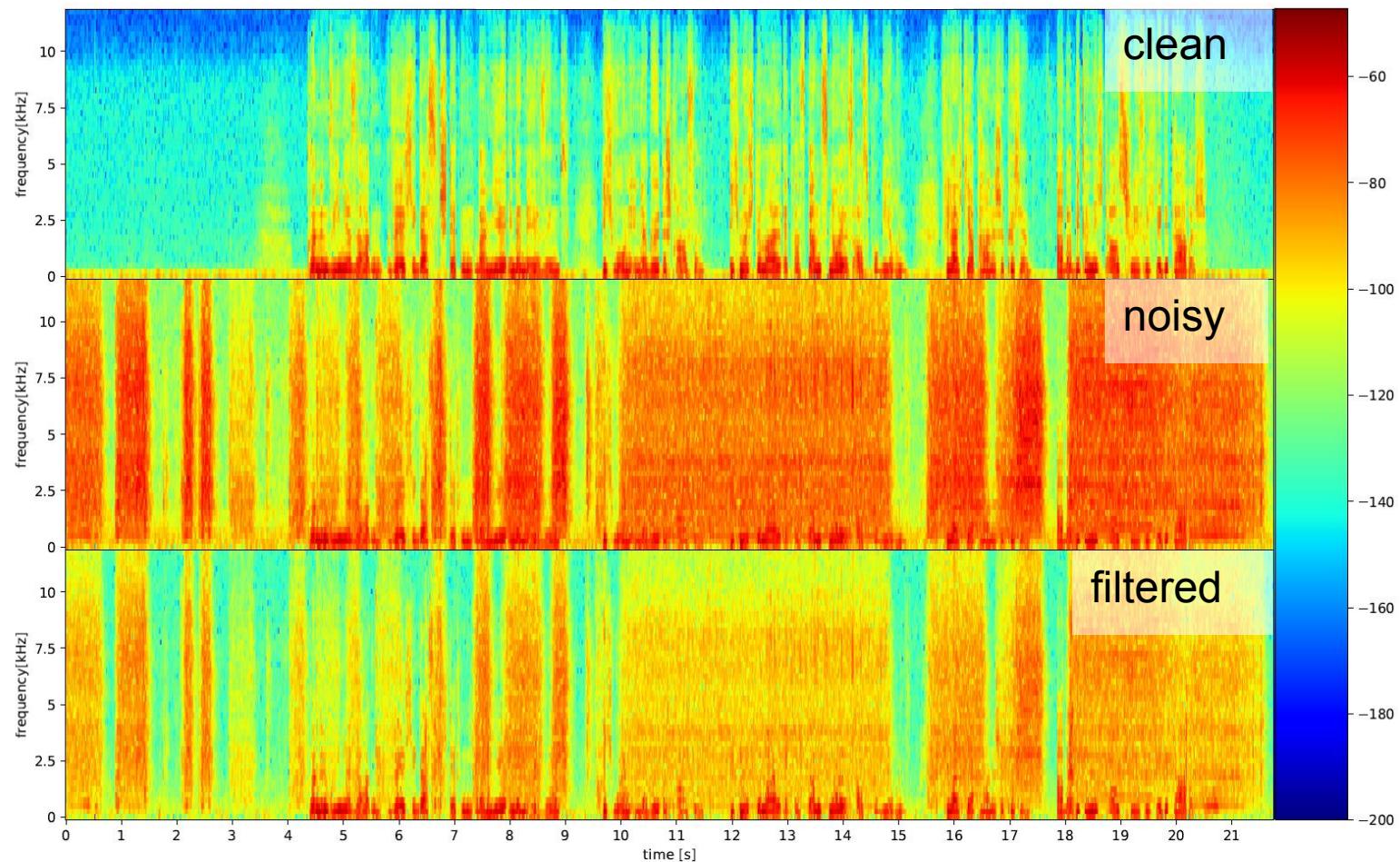
SNR: 10 → 13 dB



Used with permission from Kai Ehrenspäger and Marc Aubreville

Results

SNR: -5 → 6.7 dB



[8] Aubreville, Marc, et al. "Deep Denoising for Hearing Aid Applications." 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC). IEEE, 2018.

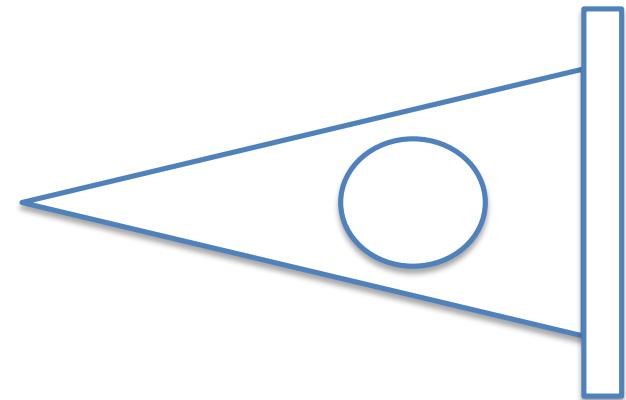
Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$



Cone-beam acquisition

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

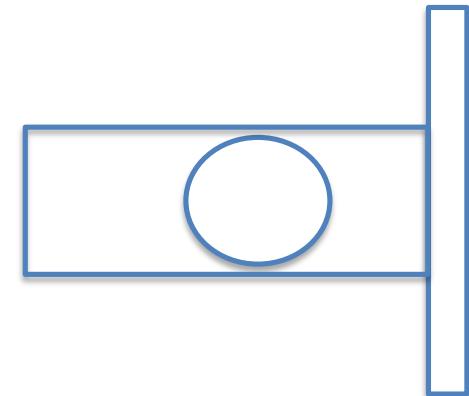
$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$
$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

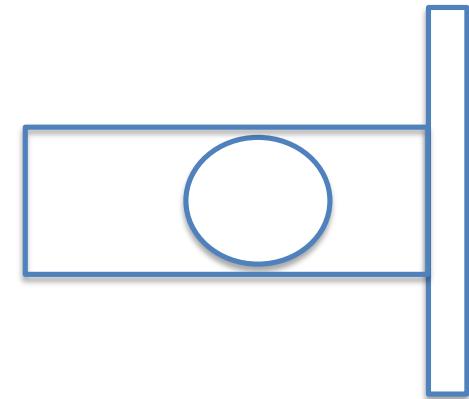


Parallel projection

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$
$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$



Parallel projection
Can't be measured!

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

$$x = \mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

$$x = \mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

$$\mathbf{p}_{PB} = \mathbf{A}_{PB}\mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

$$x = \mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

$$\mathbf{p}_{PB} = \mathbf{A}_{PB}\mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

$$\mathbf{p}_{PB} = \mathbf{A}_{PB}\mathbf{A}_{CB}^\top F^H K F \mathbf{p}_{CB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

$$x = \mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

$$\mathbf{p}_{PB} = \mathbf{A}_{PB}\mathbf{A}_{CB}^\top (\mathbf{A}_{CB}\mathbf{A}_{CB}^\top)^{-1} \mathbf{p}_{CB}$$

$$\mathbf{p}_{PB} = \boxed{\mathbf{A}_{PB}\mathbf{A}_{CB}^\top F^H K F \mathbf{p}_{CB}}$$

New Net
Topology ?

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. <https://arxiv.org/abs/1807.03057>

Known Operator Learning

- **Many traditional approaches mathematically equivalent to neural networks and vice versa**
- **Learned algorithms are again traditional algorithms**
- **Side effect: Learned parameters can be interpreted**
- **Many state-of-the art methods can be integrated**
- **Methods are efficient and interpretable**

NEXT TIME
ON DEEP LEARNING