

How to Put Algorithms inside Neural Networks?

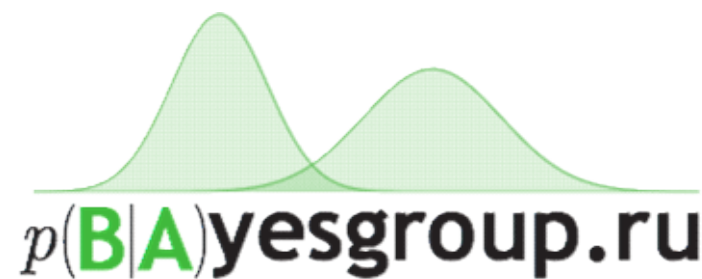
Anton Osokin

CS, National Research University HSE
Samsung-HSE Laboratory



NATIONAL RESEARCH
UNIVERSITY

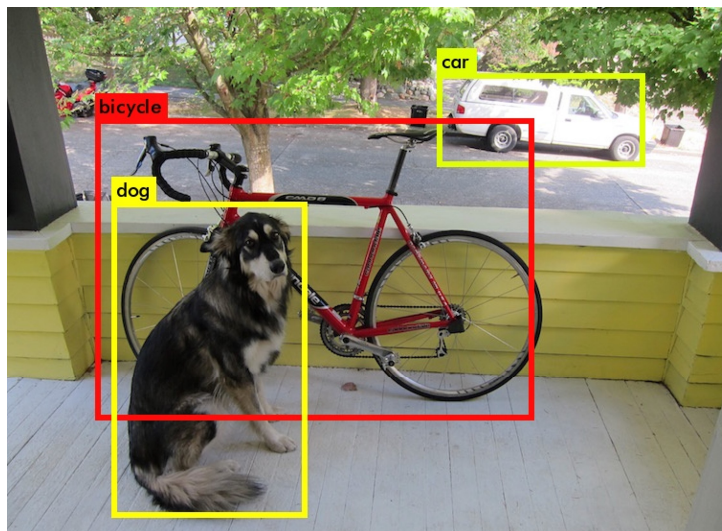
Warsaw, 2019



Neural Nets in Computer Vision



- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



(Redmon&Farhadi, 2017)



(He et al., 2017)

Neural Nets in NLP



Привет, я Алиса

Ваш голосовой помощник,
придуманный в компании Яндекс.
Многие вещи проще делать, говоря со мной.

Neural Nets in Audio

- text2speech – WaveNet (van den Oord et al., 2016)



- Music generation



Performance RNN was trained in TensorFlow on MIDI from piano performances. It was then ported to run in the browser using only Javascript in the deeplearn.js environment.

Why Algorithms in Neural Nets?


- For many tasks, we have good networks
- New tasks often require specialized architectures
- Nets can't just learn everything, they need help!
- We have powerful algorithms in many domains!
- Combining nets and existing solutions
 - Building new layers/architectures
 - Trainable algorithm

Outline

- Structured pooling of activations
- Unrolling iterations into layers
- Analytical derivative w.r.t. the input

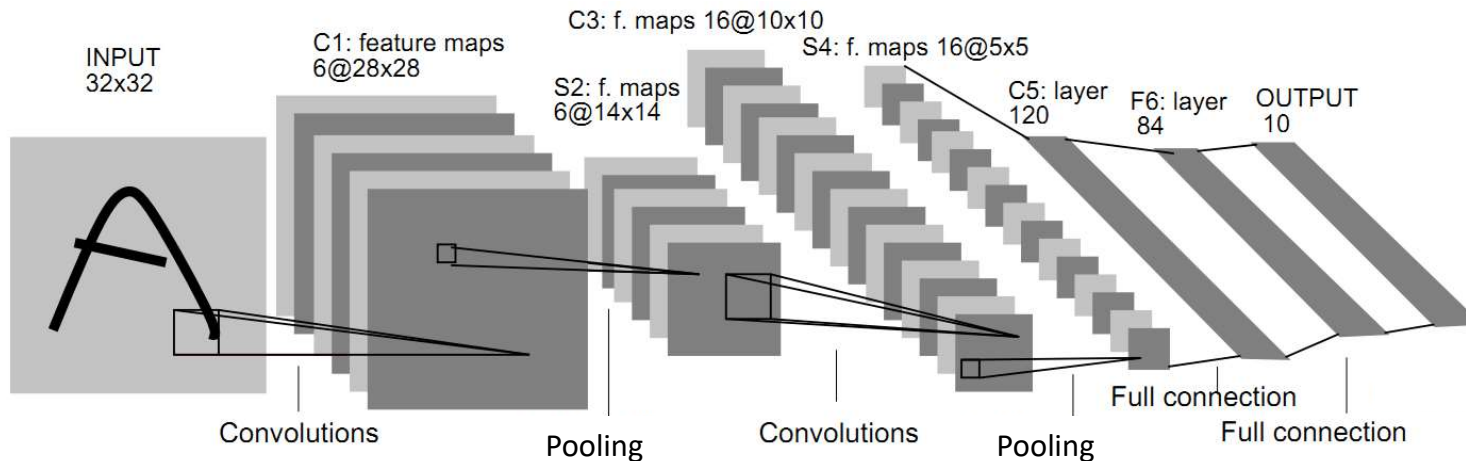
Running Example: Handwriting Recognition

- Simplified task – each symbol is a separate picture
 - OCR dataset (Taskar et al., 2003)

 → command

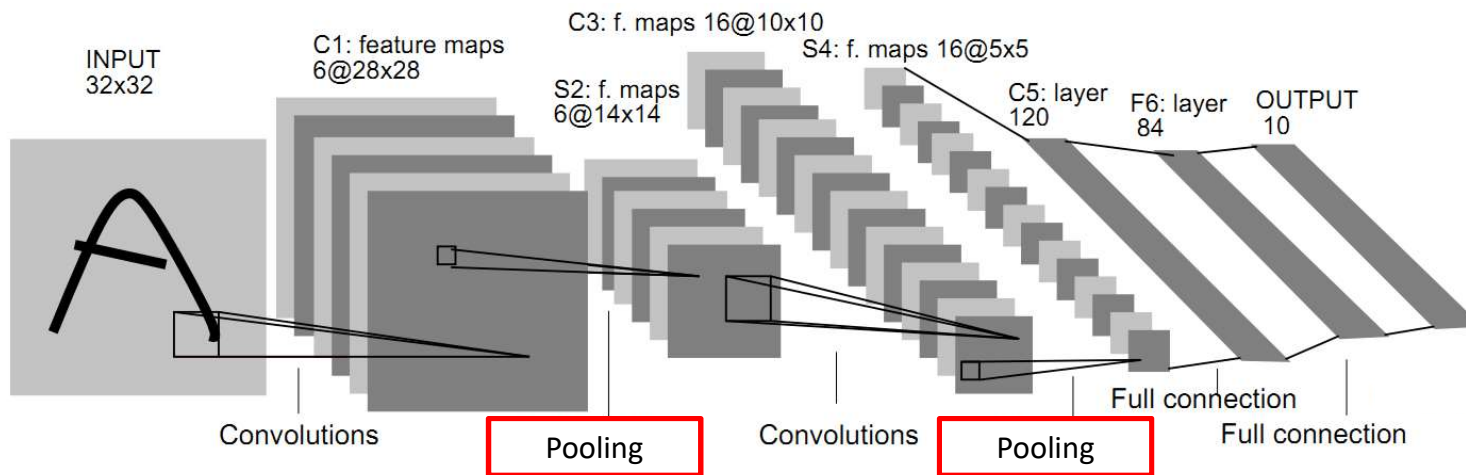
- A MNIST of structured prediction
- Classifying symbols separately: 8% error
- Classifying jointly: 1-3% error

LeNet for OCR



- Linear operations with parameters
 - Convolutions for images
- Nonlinearities (sigmoid, ReLu)
- Pooling to choose activations
- Training by backprop and stochastic optimization

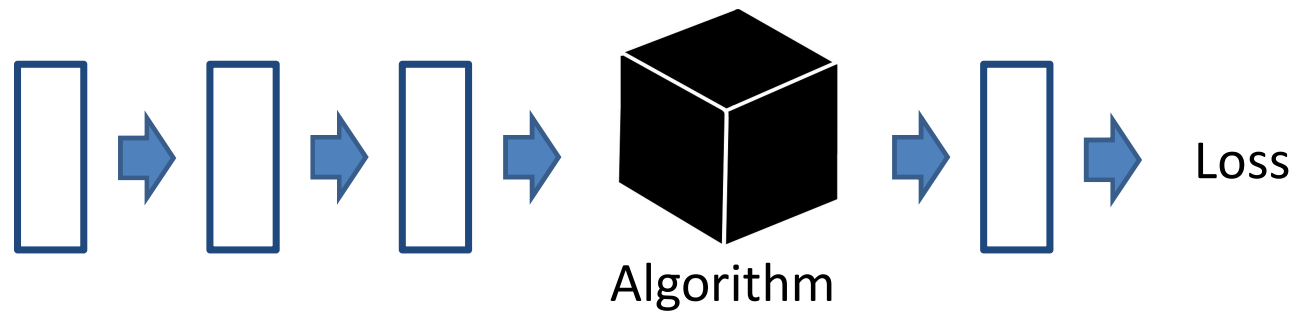
Pooling is an Algorithm



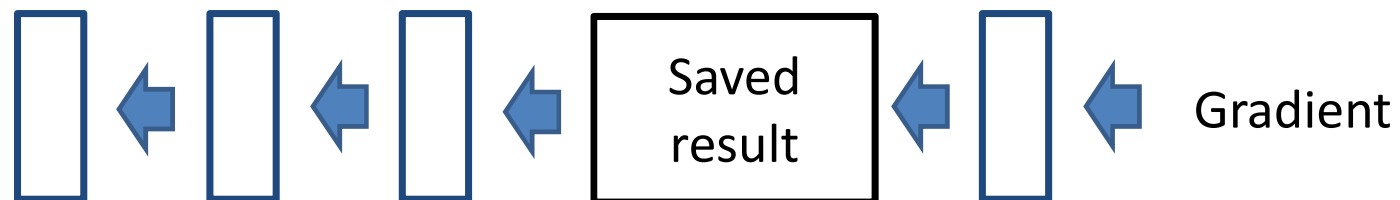
- Pooling aggregates activations (usually max or average)
- Can use others:
 - Quantile (k^{th} best)
 - Median
- Is this differentiable?
 - «Backpropable» - weaker notion of differentiability

Approach 1: Structured Pooling

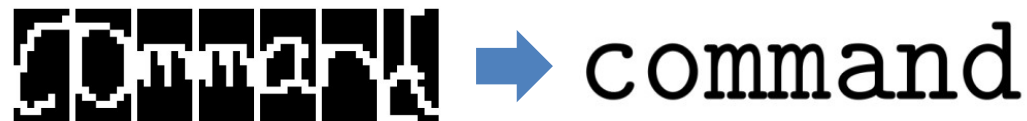
Forward pass



Backward pass: need only the result (not the algorithm)



Conditional Random Field (CRF)



- CRF – a way to define interactions
- A **score function** gives a score for each labeling
$$F(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^T \theta_i(y_i \mid x_i) + \sum_{i=1}^{T-1} \theta_{i,i+1}(y_i, y_{i+1})$$
- \mathbf{y} – labels, \mathbf{x} – images, $\boldsymbol{\theta}$ – potentials (unary and pairwise)
- Unary potentials $\theta_i(y_i \mid x_i)$ are computed by a ConvNet
- Pairwise potentials $\theta_{i,i+1}(y_i, y_{i+1})$ show symbol compatibility
- Chain-like graphical model

CRF: Searching for the Best Labeling

- A **score function** gives a score for each labeling

$$F(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^T \theta_i(y_i \mid x_i) + \sum_{i=1}^{T-1} \theta_{i,i+1}(y_i, y_{i+1})$$

- \mathbf{y} – labels, \mathbf{x} – images, $\boldsymbol{\theta}$ – potentials (unary and pairwise)

- Best labeling – **the one with max score F**

- **Dynamic programming** (Viterbi algorithm)

- Iteratively computing the Bellman function:

$$V_{i+1}(y_{i+1}) = \theta_{i+1}(y_{i+1} \mid x_{i+1}) + \max_{y_i} \left(\theta_{i,i+1}(y_i, y_{i+1}) + V_i(y_i) \right)$$

- Find the optimal value: $F^* = \max_{y_T} V_T(y_T)$

- Retrieve the optimal configuration with parent pointers

Learning Potentials of a CRF – Structured SVM

- Labeled training set $\left\{ \mathbf{x}_n, \mathbf{y}_n \right\}_{n=1}^N$
- Learning = optimizing the loss $\min_{\theta} \frac{1}{N} \sum_{n=1}^N \Psi(\mathbf{x}_n, \mathbf{y}_n \mid \theta)$
- **Structured SVM** is a generalization of SVM for structured outputs

$$\Psi(\mathbf{x}_n, \mathbf{y}_n \mid \theta) = \max_{\mathbf{y}} \left[F(\mathbf{y} \mid \mathbf{x}_n, \theta) + \Delta(\mathbf{y}, \mathbf{y}_n) \right] - F(\mathbf{y}_n \mid \mathbf{x}_n, \theta)$$

- One learning iteration
 1. Compute the potentials (NN forward pass)
 2. Compute the loss (dynamic programming)
 3. Compute the gradient w.r.t. the potentials
 4. Backprop the gradient to NN parameters
 5. Step of an optimizer

Applications of Structured Pooling

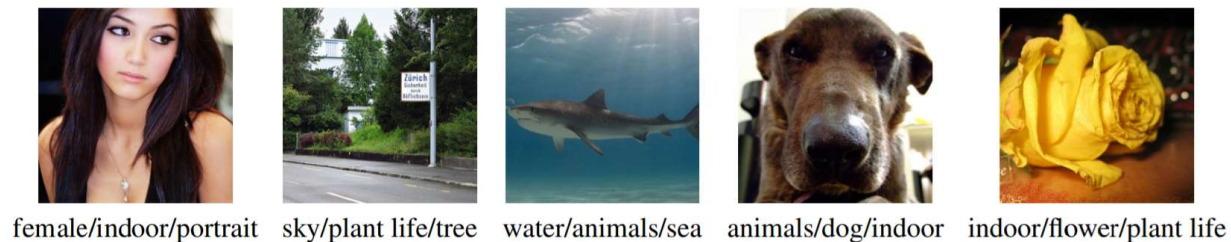
- Free text recognition (Jaderberg et al., ICLR 2015)



- Detection of multiple objects (Vu et al., ICCV 2015)

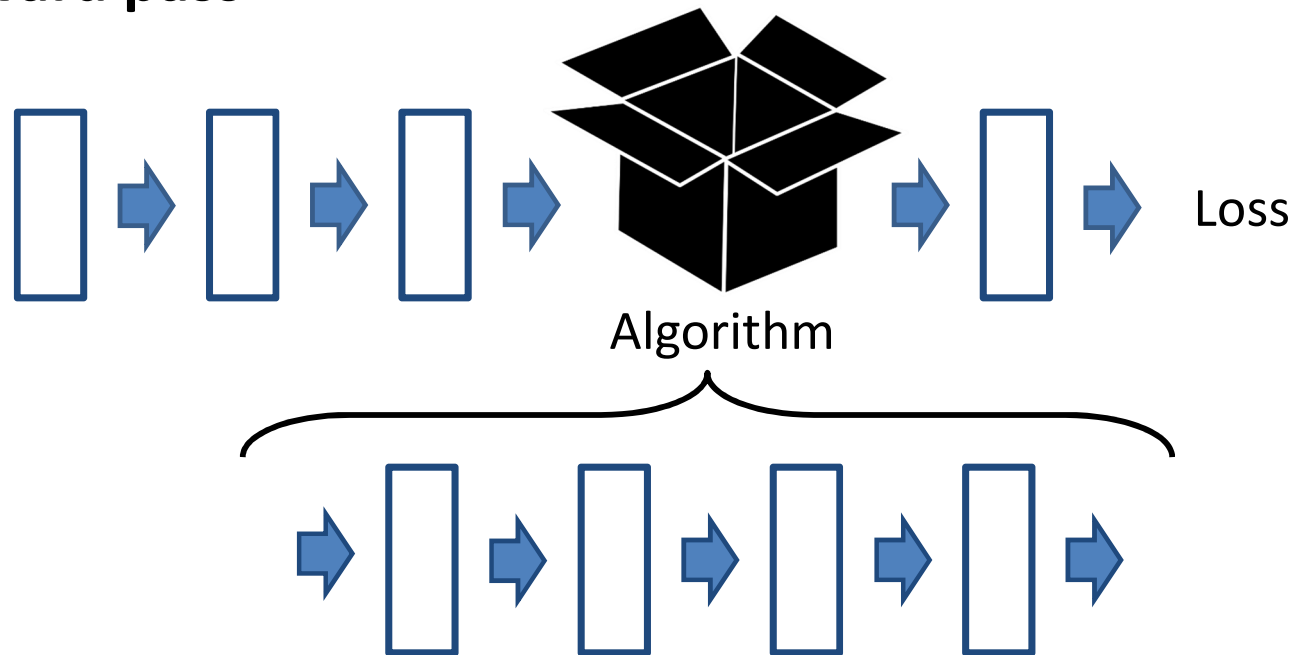


- Image tagging (Chen et al., ICML 2015)



Approach 2: Unrolling Iterations into Layers

Forward pass



Backward pass – regular backprop

Learning Potentials of a CRF – Maximum Likelihood

 → command

- Labeled training set $\{x_n, y_n\}_{n=1}^N$
- Learning = optimizing the loss $\min_{\theta} \frac{1}{N} \sum_{n=1}^N \Psi(x_n, y_n | \theta)$
- **Maximum likelihood approach**
$$\Psi(x_n, y_n | \theta) = -\log P(y | x_n, \theta), \quad P = \frac{1}{Z(x_n, \theta)} \exp(F(y | x_n, \theta))$$
- Z – normalization constant (partition function)
$$Z(x_n, \theta) = \sum_y \exp(F(y | x_n, \theta))$$
- Z = the sum of the exponential number of summands
- **We need an Algorithm!**

Learning Potentials of a CRF – Maximum Likelihood

- Z – normalizing constant

$$Z(\mathbf{x}_n, \boldsymbol{\theta}) = \sum_{\mathbf{y}} \exp(F(\mathbf{y} \mid \mathbf{x}_n, \boldsymbol{\theta}))$$

- Score function F has structure:

$$\exp(F(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})) = \prod_{i=1}^T \exp(\theta_i(y_i \mid x_i)) \prod_{i=1}^{T-1} \exp(\theta_{i,i+1}(y_i, y_{i+1}))$$

- Again **dynamic programming** (sum-product)

- Iteratively computing the Bellman function :

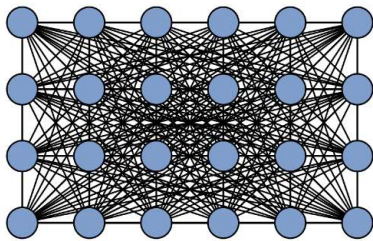
$$V_{i+1}(y_{i+1}) = \exp(\theta_{i+1}(y_{i+1} \mid x_{i+1})) \sum_{y_i} \left(\exp(\theta_{i,i+1}(y_i, y_{i+1})) V_i(y_i) \right)$$

- Result: $Z = \sum_{y_T} V_T(y_T)$

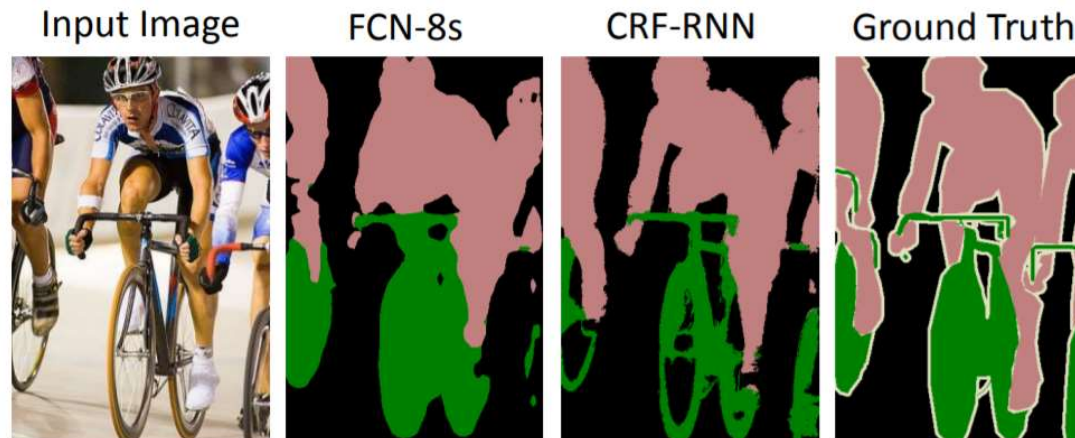
Example: Image Segmentation (CRF as RNN)

(Zheng et al., ICCV 2015)

- Variational inference for dense CRF



- All operations are implemented with convolutions



Example: Image Denoising

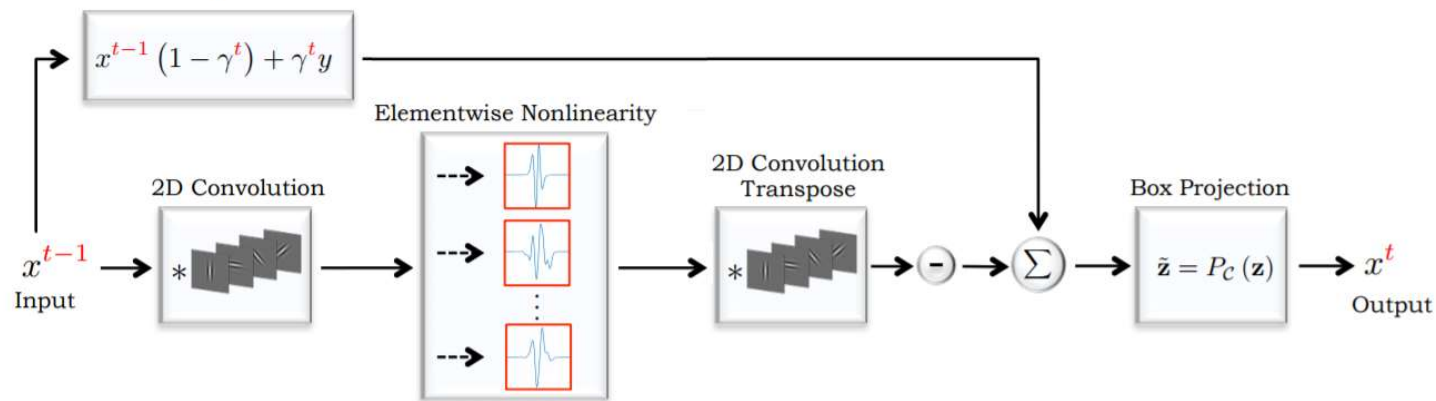
(Lefkimmiatis, CVPR 2017)

- Given a noisy image \mathbf{y} , we want a clean image \mathbf{x} :

$$\mathbf{x}^* = \arg \min_{a \leq x_n \leq b} \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \sum_{r=1}^R \phi(\mathbf{L}_r \mathbf{x})$$

- Algorithm – proximal gradient optimization

$$\mathbf{x}^t = P_C \left(\mathbf{x}^{t-1} (1 - \gamma^t) + \gamma^t \mathbf{y} - \alpha^t \sum_{r=1}^R \mathbf{L}_r^T \psi(\mathbf{L}_r \mathbf{x}^{t-1}) \right)$$



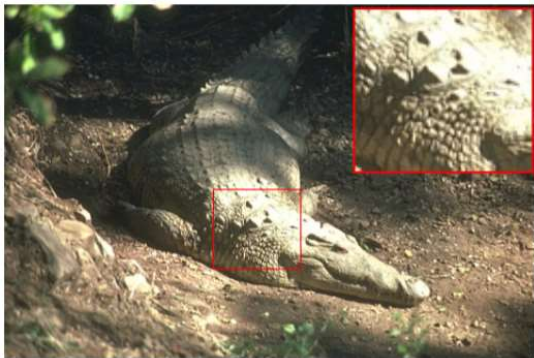
Example: Image Denoising

(Lefkimmiatis, CVPR 2017)

- Given a noisy image \mathbf{y} , we want a clean image \mathbf{x} :

$$\mathbf{x}^* = \arg \min_{a \leq x_n \leq b} \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \sum_{r=1}^R \phi(\mathbf{L}_r \mathbf{x})$$

- Results:



Original



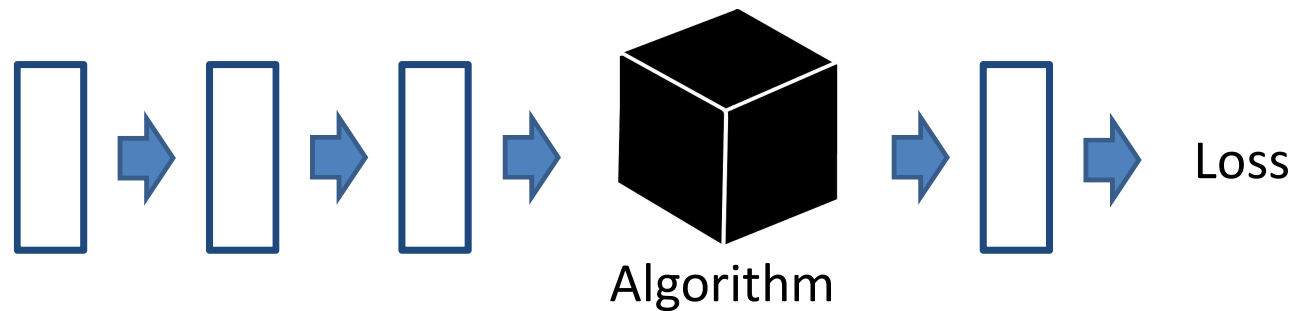
Original + noise



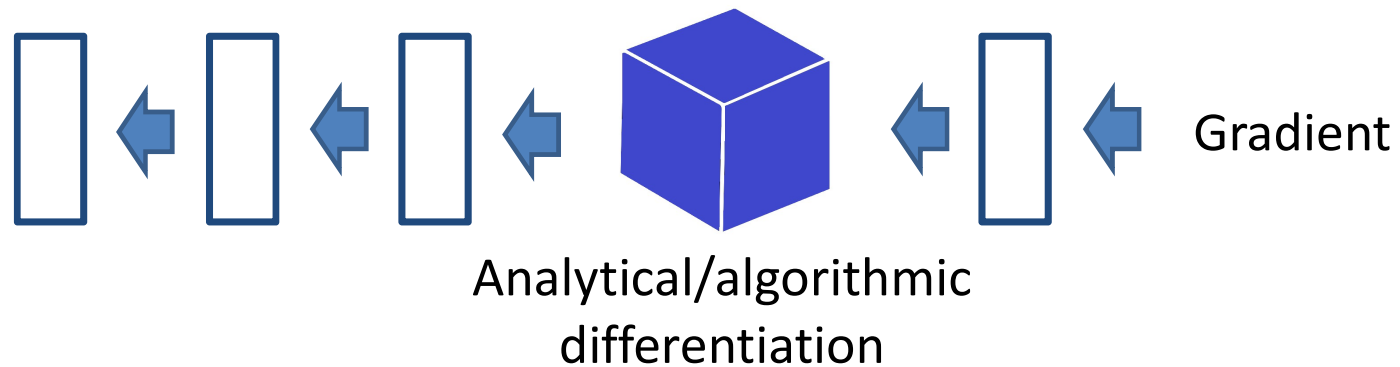
Result

Approach 3: Analytical derivative w.r.t. the input

Forward pass



Backward pass: we need another computation/algorithm



Gaussian CRF for Image Segmentation

(Chandra&Kokkinos, ECCV 2016)

- Quadratic score function – continuous analog of CRF:

$$F(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T(A + \lambda I)\mathbf{y} + \mathbf{b}^T\mathbf{y} \rightarrow \max_{\mathbf{y}}$$

- \mathbf{y} – variables, A , \mathbf{b} – parameters (come from a neural network)
- **Algorithm** – solving a system of linear equations: $\mathbf{y} = (A + \lambda I)^{-1}\mathbf{b}$
- Task of differentiation:
 - Input: parameters A , \mathbf{b} , solution \mathbf{y} , gradient $\frac{d\Psi}{d\mathbf{y}}$
 - Find gradients $\frac{d\Psi}{d\mathbf{b}}$ and $\frac{d\Psi}{dA}$
- Gradients can be computed by another system of linear equations

$$\frac{d\Psi}{d\mathbf{b}} = (A + \lambda I)^{-T} \frac{d\Psi}{d\mathbf{y}} \quad \frac{d\Psi}{dA} = -\frac{d\Psi}{d\mathbf{b}}\mathbf{y}^T$$

More Examples


- **SVD decomposition** $X = U\Sigma V^T$ (Ionescu et al., ICCV 2015)

$$\frac{\partial L \circ f}{\partial X} = DV^\top + U \left(\frac{\partial L}{\partial \Sigma} - U^\top D \right)_{diag} V^\top + 2U\Sigma \left(K^\top \circ \left(V^\top \left(\frac{\partial L}{\partial V} - VD^\top U\Sigma \right) \right) \right)_{sym} V^\top$$

$$K_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & i \neq j \\ 0, & i = j \end{cases} \quad D = \left(\frac{\partial L}{\partial U} \right)_1 \Sigma_n^{-1} - U_2 \left(\frac{\partial L}{\partial U} \right)_2^\top U_1 \Sigma_n^{-1}$$

- **Submodular (discrete!) optimization** (Djolonga&Krause, NIPS 2017)
 - Use Lovász to extension to build equivalent continuous problem
 - Use specialized algorithms
- **Ordinary differential equations** (Chen et al., NeurIPS 2018)
 - Gradient – solving another ODE

Common Strategy: Implicit Differentiation

- Algorithm as the implicit layer
 - Explicit layer $\mathbf{y} := f(\mathbf{x})$
 - Implicit layer $h(\mathbf{x}, \mathbf{y}(\mathbf{x})) = 0$ (defined by an equation)
 - Algorithm solves the equation
- How to backprop?
 1. $\frac{d}{d\mathbf{x}} h(\mathbf{x}, \mathbf{y}(\mathbf{x})) = 0$
 2. $\frac{\partial h}{\partial \mathbf{x}} + \frac{\partial h}{\partial \mathbf{y}} \frac{d\mathbf{y}}{d\mathbf{x}} = 0$  Automatic differentiation
 3. $\frac{d\mathbf{y}}{d\mathbf{x}} := -\left(\frac{\partial h}{\partial \mathbf{y}}\right)^{-1} \frac{\partial h}{\partial \mathbf{x}}$
 - A. Jacobian invertible? (+ regularization)
 - B. Implicit methods to solve

Examples of Implicit Layers

- OptNet: Differentiable Optimization as a Layer [Amos&Kolter, 2017]
- Example: quadratic program (QP)

$$\hat{\mathbf{y}}(\mathbf{x}) := \arg \min_{\mathbf{y}} \frac{1}{2} \mathbf{y}^T Q(\mathbf{x}) \mathbf{y} + \mathbf{y}^T \mathbf{c}(\mathbf{x})$$

$$\text{s.t. } A(\mathbf{x}) \mathbf{y} = \mathbf{b}(\mathbf{x})$$

$$G(\mathbf{x}) \mathbf{y} \leq \mathbf{s}(\mathbf{x})$$

Example application:
image denoising

- Equations: KKT conditions

$$Q\mathbf{y} + \mathbf{c} + A^T \boldsymbol{\nu} + G^T \boldsymbol{\lambda} = 0$$

$$A\mathbf{y} - \mathbf{b} = 0$$

$$\text{diag}(\boldsymbol{\lambda})(G\mathbf{y} - \mathbf{s}) = 0$$

More examples:

- SATNet [Wang et al., 2019]
 - Relaxation of discrete optimization
- Deep equilibrium models [Bai et al, 2019]
 - Repeated usage of ResNet/Transformer
- Physics engine
[de Avila Belbute-Peres et al., 2018]
 - Laws Of Physics = equations

Conclusion

- Many algorithms can be put into neural networks
 - Structured pooling (algorithm to choose activations)
 - Unrolling the iterations (algorithm => a sequence of layers)
 - Analytical differentiation (need another algorithm)
- We can put existing method into networks (more layers!)
 - Get better networks
 - Get learnable algorithms
- **Open problems – what is hard?**
 - Control flow depends on data (if, while) – one path, all paths, sampling
 - More complicated training – convergence issues
 - Bottlenecks – amount of GPU memory, speed

Thank you for attention!