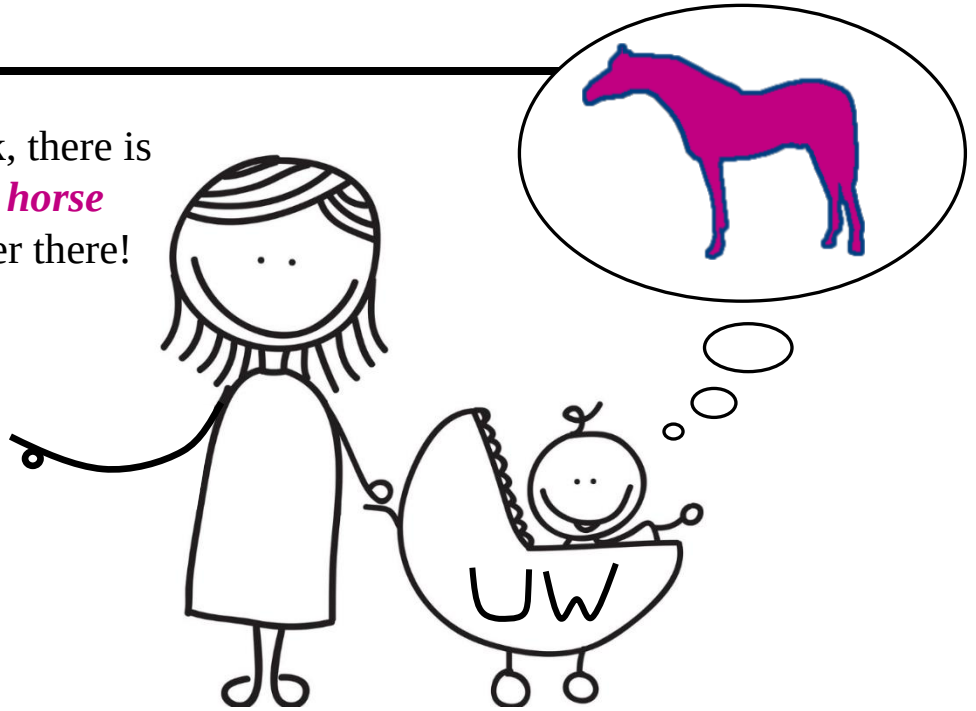


from full supervision to

Weak-supervision,
Semi-supervision,
Self-supervision,
etc.

Weakly-Supervised CNN training (outline)

- **Weakly-supervised** CNN segmentation
  - shortage of training labels
  - proposal-based weak-supervision
  - partial pixel-level supervision, image-level supervision
  - **unsupervised loss functions** from low-level vision (topics 8,9)
- Towards **self-supervision**
  - monocular depth, NERF
  - denoising, super-resolution, inpainting, *e.t.c.*
  - regularized and self-supervised losses, auto-encoders
  - generative models (GAN, VAE, diffusion models)
  - **deep clustering, self labeling, etc.** (optional topic 13)
- Limitations of NNs

Semantic segmentation with CNNs

Topic 11 discussed **fully supervised** CNN training expecting fully labeled **pixel-accurate training masks**

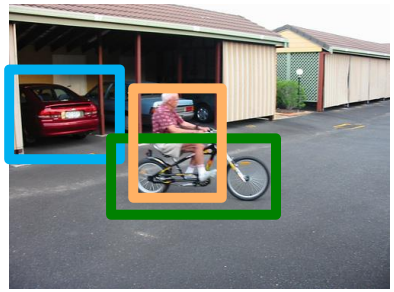
The concept of learning to segment a horse from thousands of pixel-accurate horse segments is “a bit” ridiculous.

In any case, it is **highly expensive**.

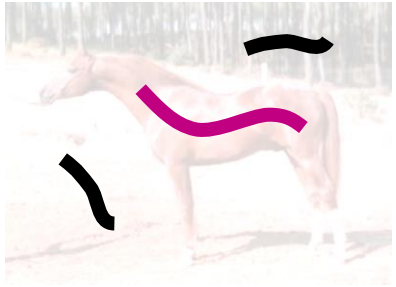
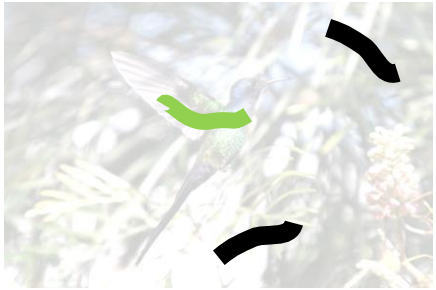
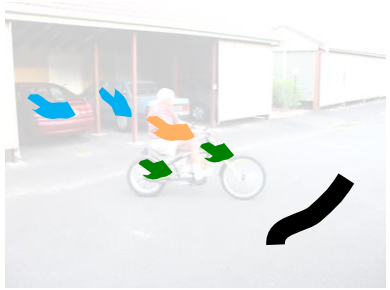
Semantic segmentation with CNNs

First, we discuss training segmentation CNNs using **partial pixel-level supervision**





Boxes
two clicks per object

Seeds
brush stroke per object

significantly cheaper forms of **weak supervision** for segmentation

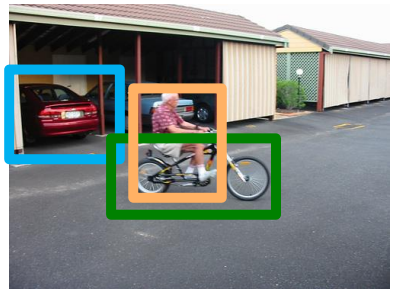
Naïve approach: “proposal generation”

essentially “fake” ground truth

Use available boxes/seeds to generate “**training proposals**”

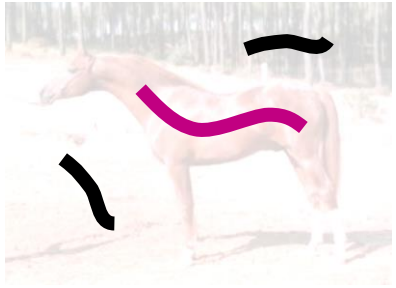
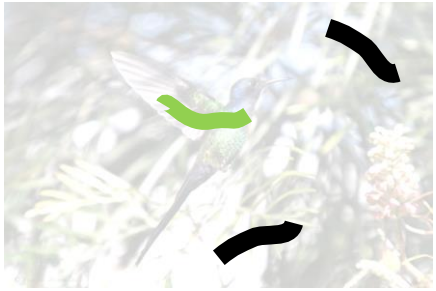
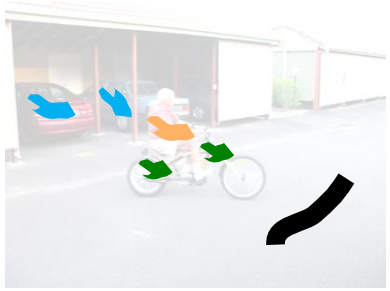
e.g. using interactive low-level segmentation (e.g. graph cuts, Topic 9)





Boxes

two clicks per object

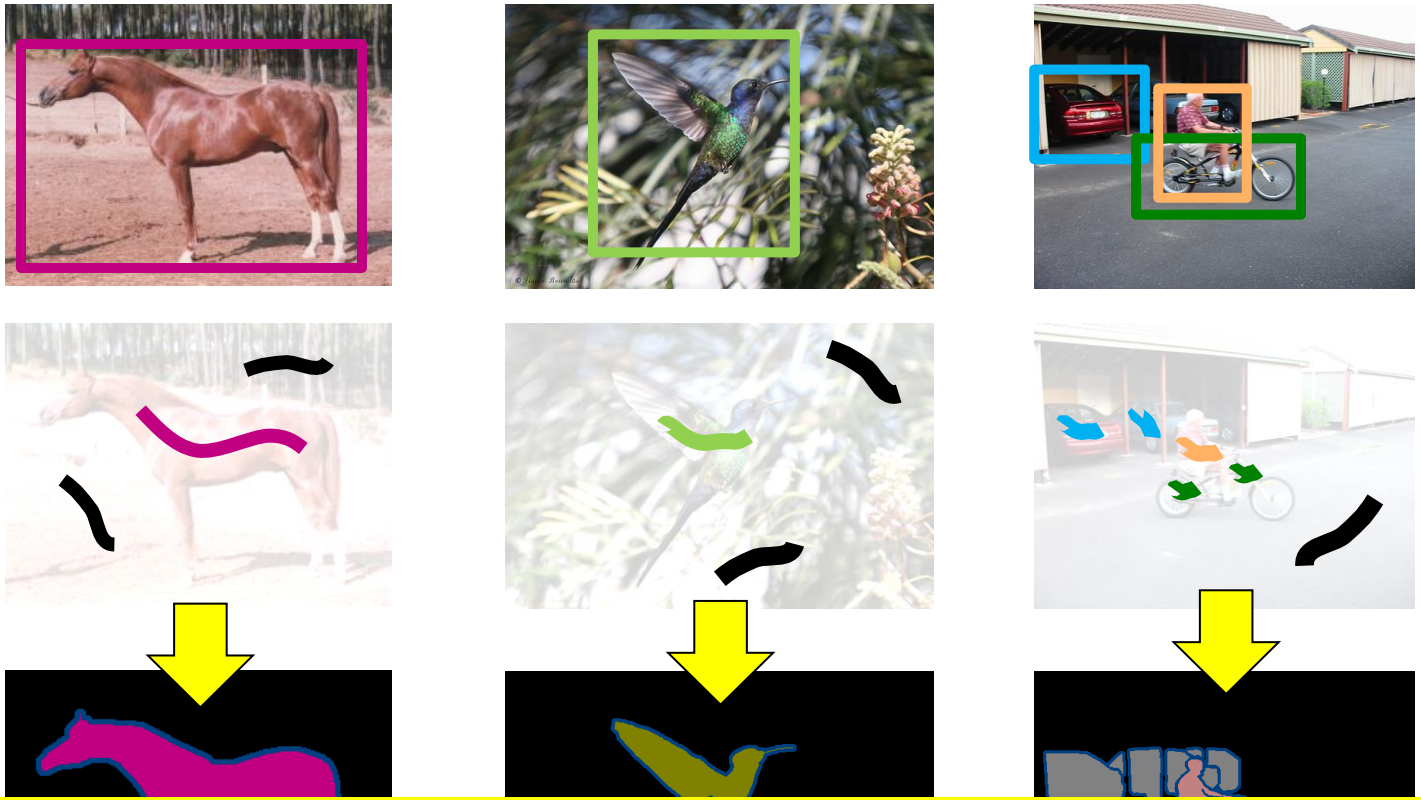
Seeds

brush stroke per object

Naïve approach: "proposal generation"

essentially "fake" ground truth

Use available boxes/seeds to generate "**training proposals**"
e.g. using interactive low-level segmentation (e.g. graph cuts, Topic 9)



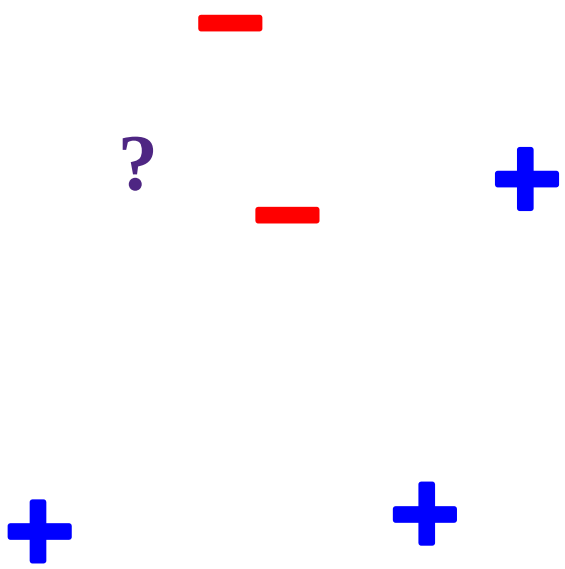
Boxes
two clicks per object

Seeds
brush stroke per object

Problem: in practice, so generated proposals will have mistakes.
Can't use previous losses assuming that target is certain/correct.

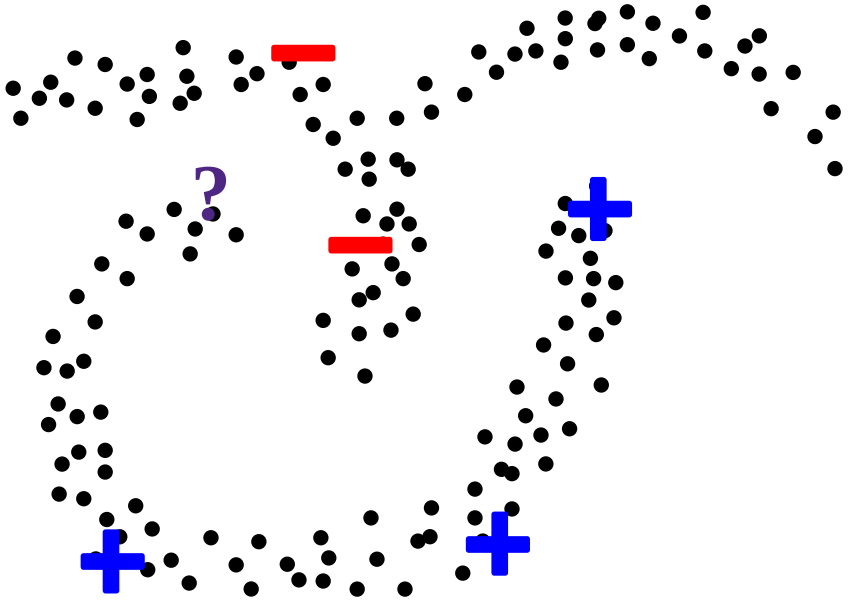
There are better standard ideas

Semi-supervised learning



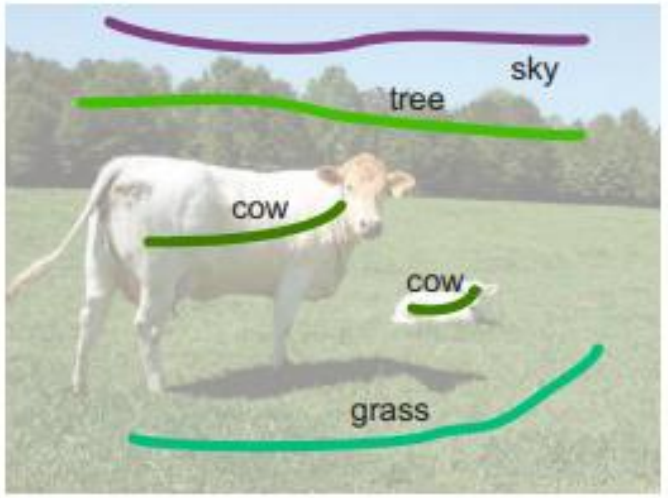
There are better standard ideas

Semi-supervised learning



unlabeled data can be informative

Weakly-supervised segmentation

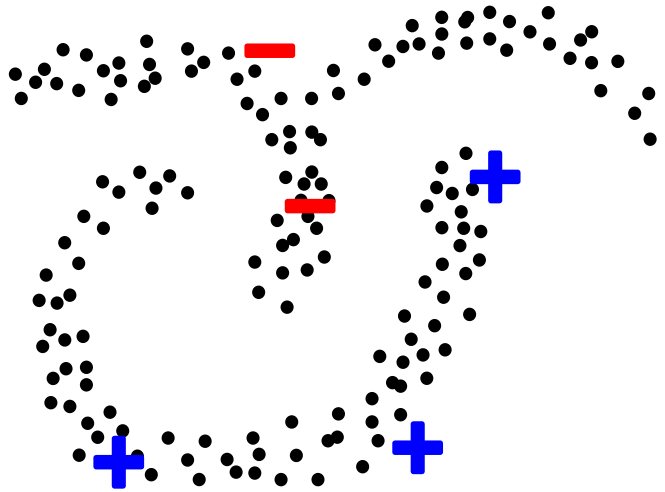


Similarly, unlabeled pixels can be informative

Semi-supervised learning

$\mathbf{y}^i$ - true label or *target*, if any, for data point (feature vector) $\mathbf{x}^i$

Definition: Given M labeled data points $(\mathbf{x}^i, \mathbf{y}^i) \in (\mathcal{X}, \mathcal{Y})$, $i = 1, ..., M$ and U unlabeled data points $\mathbf{x}^i$, $i = M+1, ..., M+U$, learn prediction function $f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$.



[Zhu & Goldberg, "Introduction to semi-supervised learning", 2009]
[Chapelle, Scholkopf & Zien, "Semi-supervised learning", 2009]

Graph-Based Semi-supervised Learning

Loss function ?

- labelled points should have **consistency with the target**

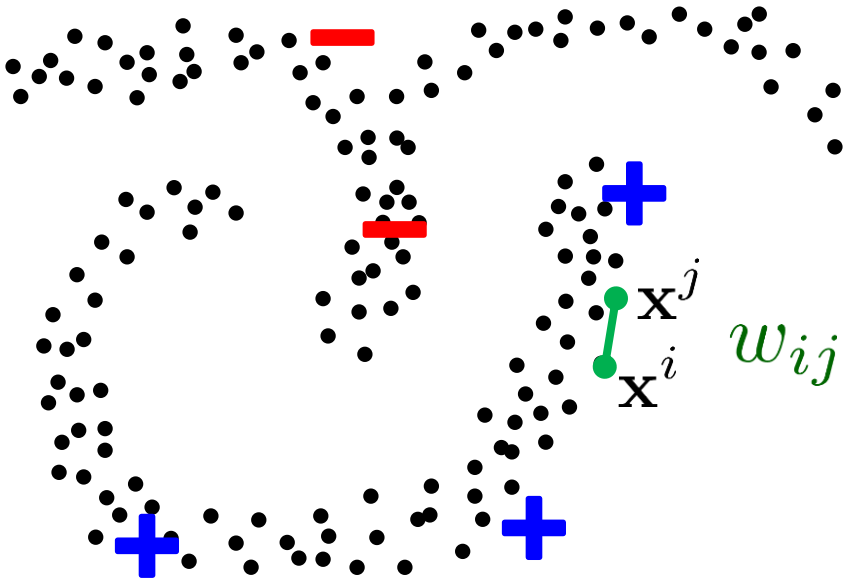
e.g.

$$\sum_{i=1}^{M} \delta(f(\mathbf{x}^i) \neq \mathbf{y}^i)$$

- unlabeled points should be labeled so that there is some agreement between neighbors

i.e. **pairwise regularization**:

$$\sum_{ij \in \mathcal{N}} w_{ij} \, ||f(\mathbf{x}^i) - f(\mathbf{x}^j)||^2$$

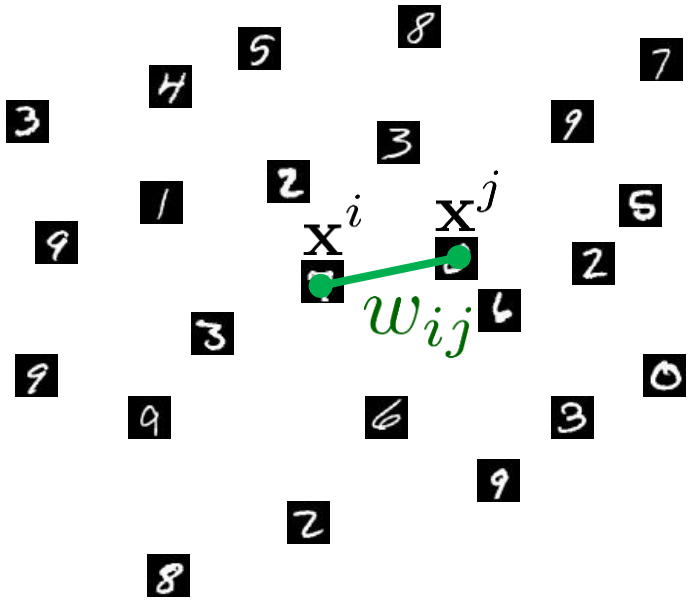


w_{ij} - pre-computed affinities, e.g. based on distance between feature vectors $\mathbf{x}^i$ and $\mathbf{x}^j$ (e.g. Gaussian kernel)

Deep Semi-supervised Learning

Classification

(Weston et al. 2012)



e.g. for **classification CNN** output

$$f(\mathbf{x}^i) = \bar{\sigma}^i \equiv (\bar{\sigma}^i_1, \ldots, \bar{\sigma}^i_K)$$

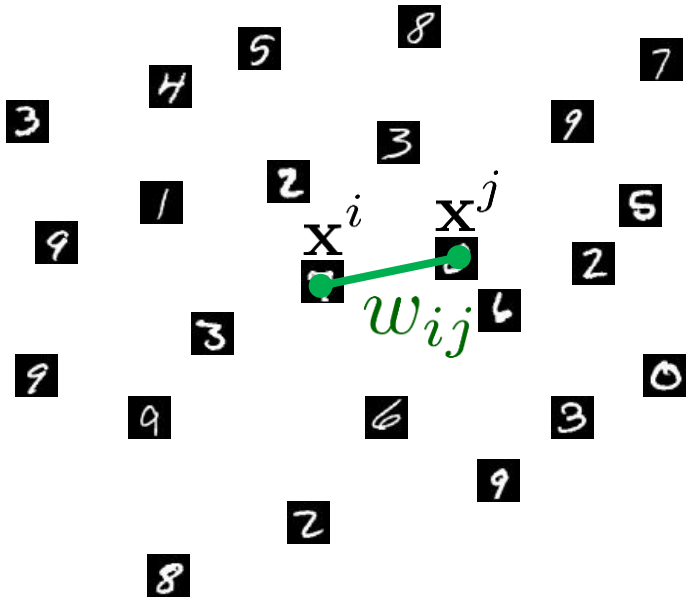
class probabilities at point *i*

$$\sum_{ij \in \mathcal{N}} w_{ij} \quad ||\bar{\sigma}^i - \bar{\sigma}^j||^2$$

Deep Semi-supervised Learning

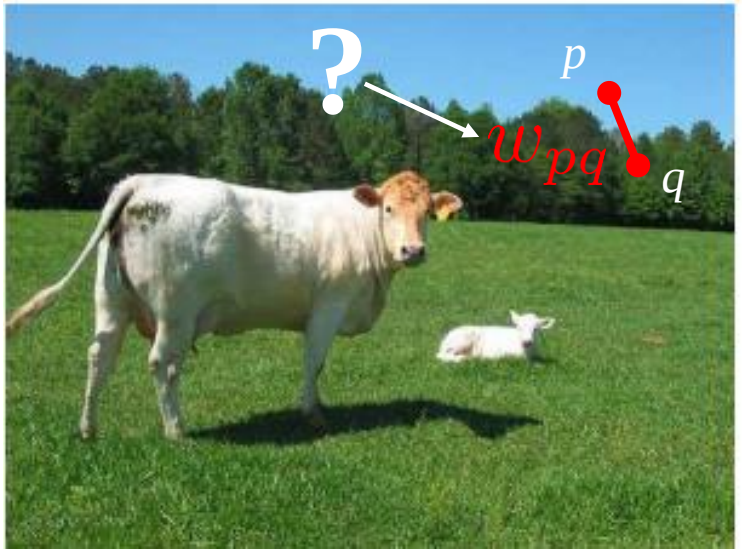
Classification

(Weston et al. 2012)



Segmentation

(Tang et al. CVPR18, ECCV18)



e.g. for **classification CNN** output

$$f(\mathbf{x}^i) = \bar{\sigma}^i \equiv (\bar{\sigma}^i_1, \ldots, \bar{\sigma}^i_K)$$

class probabilities at point *i*

e.g. for **segmentation CNN** output

$$\bar{\sigma}^p \equiv (\bar{\sigma}^p_1, \ldots, \bar{\sigma}^p_K)$$

class probabilities at pixel *p*

$$\sum_{ij \in \mathcal{N}} w_{ij} \quad ||\bar{\sigma}^i - \bar{\sigma}^j||^2$$

$$\sum_{pq \in \mathcal{N}} w_{pq} \quad ||\bar{\sigma}^p - \bar{\sigma}^q||^2$$

Regularized Loss Functions

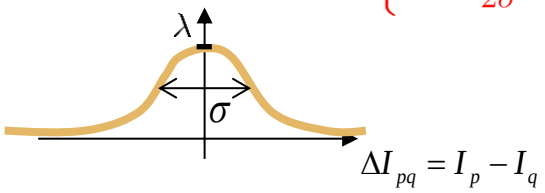
We can use regularization ideas from
unsupervised and interactive segmentation
to exploit low-level segmentation cues
(contrast alignment, boundary regularity, regional color consistency, etc.)
for unlabeled parts of an image

low-level segmentation
(Topic 9)

remember *graph cut* segmentation (topic 9):

Spatial Regularization (unsupervised)

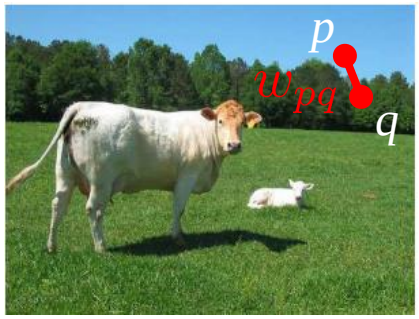
$$w_{pq} = \lambda \exp\left\{-\frac{\|I_p - I_q\|^2}{2\sigma^2}\right\}$$ - contrast weights w_{pq} from topic 9



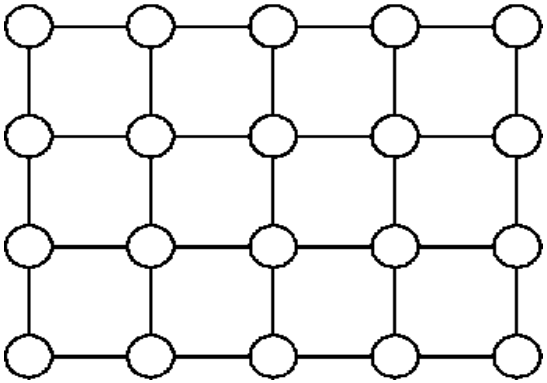
coherence between
discrete labels
at pixels p and q

$$\sum_{pq \in \mathcal{N}} w_{pq} \, [S^p \neq S^q]$$

Iverson brackets

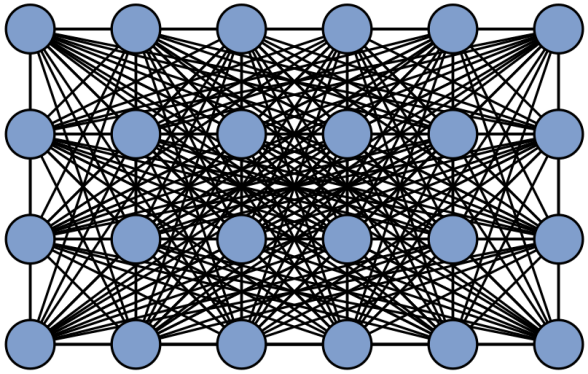


Examples of neighborhood systems $\mathcal{N}$ on pixel grid



sparsely connected

[Geman&Giman'81, BVZ PAMI'01, B&J ICCV'01]



densely connected

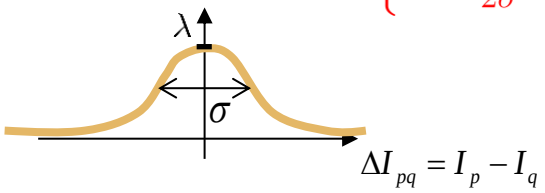
[Dense CRF, Krähenbühl & Koltun, NIPS 2011]

as in part 2 of Assignment 4

weakly-supervised CNN segmentation:

Spatial Regularization (unsupervised)

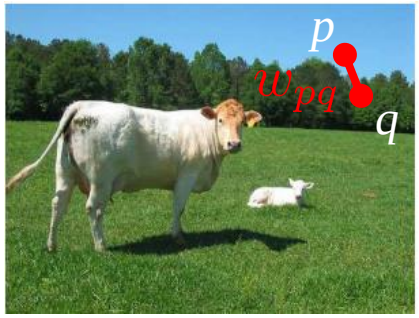
$$w_{pq} = \lambda \exp\left\{-\frac{\|I_p - I_q\|^2}{2\sigma^2}\right\}$$ - contrast weights w_{pq} from topic 9



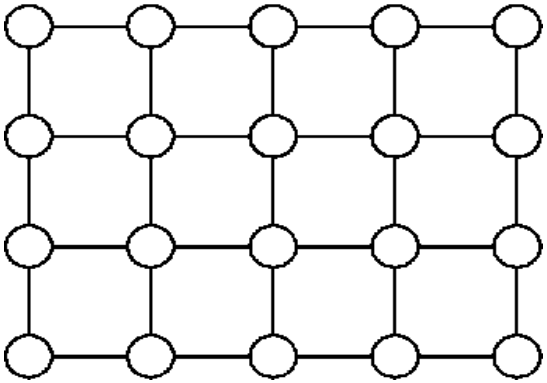
coherence between
probabilistic predictions
at pixels p and q

$$\sum_{pq \in \mathcal{N}} w_{pq} \left|\left|\bar{\sigma}^p - \bar{\sigma}^q\right|\right|^2$$

relaxation of Iverson brackets
for probabilistic predictions

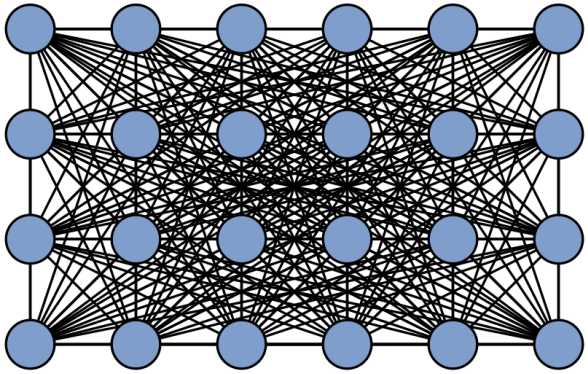


Examples of neighborhood systems $\mathcal{N}$ on pixel grid



sparsely connected

[Geman&Giman'81, BVZ PAMI'01, B&J ICCV'01]



densely connected

[Dense CRF, Krähenbühl & Koltun, NIPS 2011]

as in part 2 of Assignment 4

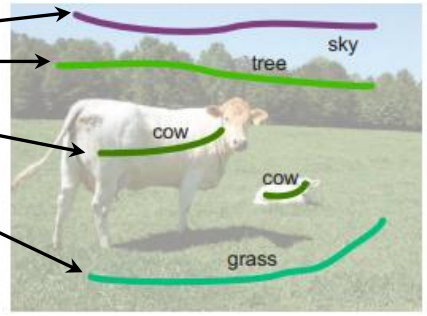
weakly-supervised CNN segmentation:

Partial Cross Entropy Loss

cross entropy

over seeds only

$$-\sum_{p \in seeds} \ln \bar{\sigma}^p_{\mathbf{y}^p}$$



not over complete fake GT mask

inaccurate cow mask

$$\bar{\sigma}^p \equiv (\bar{\sigma}^p_1, \ldots, \bar{\sigma}^p_K)$$

predicted "probabilities" for p to be in each class, e.g. (0,0,..,1,...) in **one-hot** case

Remember: if prediction is one-hot then cross entropy at seed p is equivalent to $0/\infty$ hard constraint

(as in interactive graph cut, Topic 9)

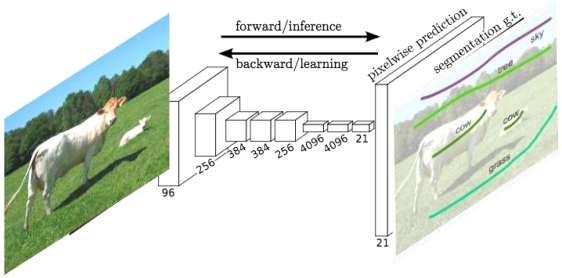
$$-\ln x := \begin{cases} 0 & \text{if } x = 1 \\ \infty & \text{if } x = 0 \end{cases}$$

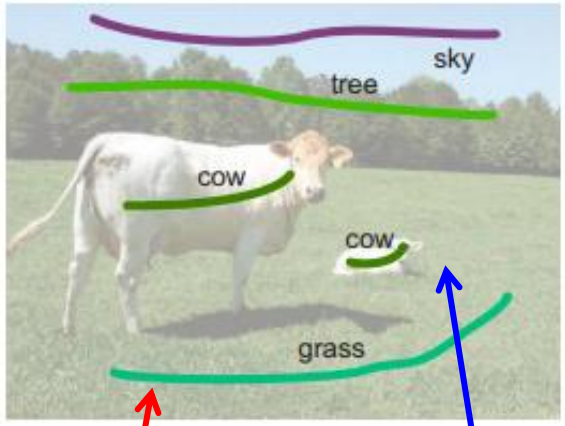
Implications:

- **Cross entropy is a relaxation of hard constraints for probabilistic predictions.**
- **Cross entropy is a bad idea for pixels where targets y^p are corrupted by errors.**
  Remember "fake" ground truths - network tries hard to learn the mistakes.

weakly-supervised CNN segmentation:

Total **Regularized Loss**





scribbles / seeds

unlabeled pixels

$$L(\bar{\sigma}) = \underbrace{-\sum_{p \in seeds} \ln \bar{\sigma}^p_{\mathbf{y}^p}}_{\text{Partial Cross Entropy (PCE)}} + \underbrace{\sum_{pq \in \mathcal{N}} w_{pq} \; ||\bar{\sigma}^p - \bar{\sigma}^q||^2}_{\text{Regularization Loss}}$$

n-links

This is nearly identical to graph cut segmentation loss

(this is its *relaxation*, coincides for one-hot predictions)

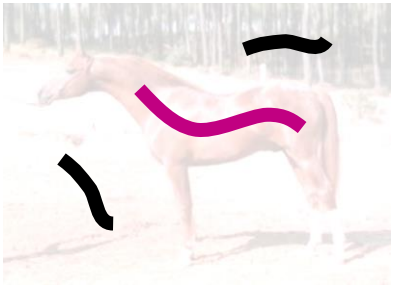
interactive **low-level** segmentation vs. weakly-supervised **semantic** CNN segmentation

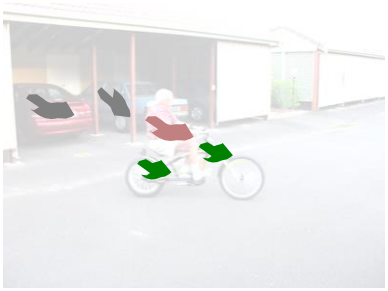
UNIVERSITY OF WATERLOO

In low-segmentation, we optimize segmentation S using loss *L(S)*.

S^p - segmentation (label or one-hot distribution) at pixel p

$\mathbf{y}^p$ - target label at seed pixel p





$$L(S) = \underbrace{-\sum_{p \in seeds} \ln S^p_{\mathbf{y}^p}}_{\text{hard constraints}} + \underbrace{\sum_{pq \in \mathcal{N}} w_{pq} \left[S^p \neq S^q\right]}_{\text{regularizer}}$$

regularized loss in low segmentation (**remember from Topic 9**)

- ~~generate "fake" full target masks~~
- postprocess CNN output

Goal: (directly) optimize segmentation variables $S = \{S^p \mid p \in \Omega\}$ (e.g. via graph cuts)

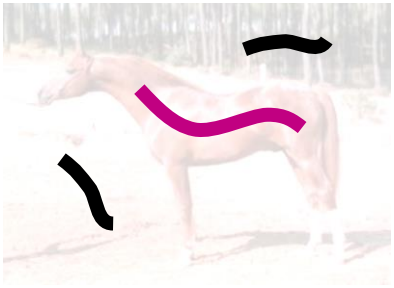
interactive **low-level** segmentation vs. weakly-supervised **semantic** CNN segmentation

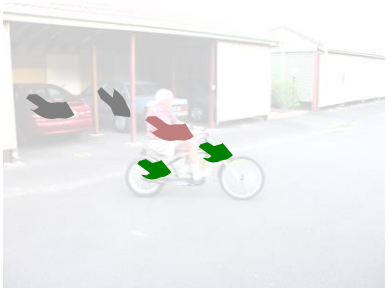
UNIVERSITY OF WATERLOO

In low-segmentation, we optimize segmentation S using loss *L(S)*.
For weakly-supervised network training, we have predictions *σ(W)* and optimize network parameters *W* using loss *L(W) = L(σ(W))*.

$\bar{\sigma}^p \equiv (\bar{\sigma}^p_1, \ldots, \bar{\sigma}^p_K)$ - probabilities at pixel p

$\mathbf{y}^p$ - target label at seed pixel p





$$L(\bar{\sigma}) = \underbrace{-\sum_{p \in seeds} \ln \bar{\sigma}^p_{\mathbf{y}_p}}_{\text{relaxation of hard constraints}} + \underbrace{\sum_{pq \in \mathcal{N}} w_{pq} \cdot ||\bar{\sigma}^p - \bar{\sigma}^q||^2}_{\text{relaxation of regularizer}}$$

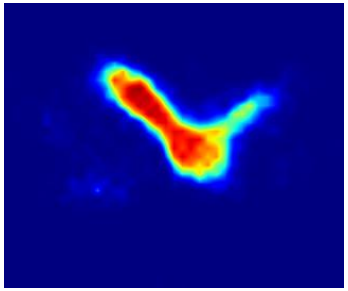
(edge alignment, smoothness)

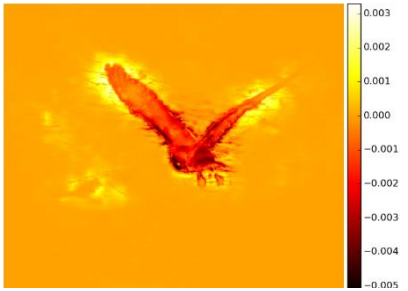
regularized loss for weakly-supervised CNN training

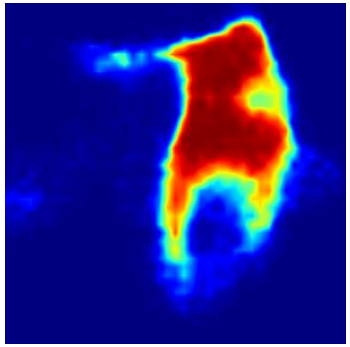
[Tang, et al. ECCV 2018, CVPR 2018]

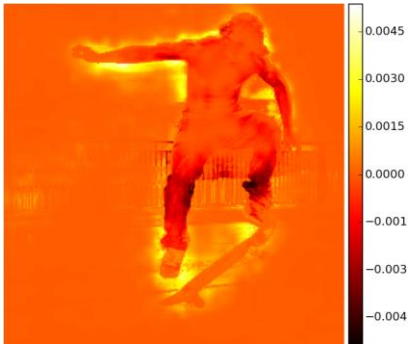
Goal: optimize network weights W giving good predictions $\sigma(W)$ (e.g. via backpropagation)

Regularization Loss Gradients









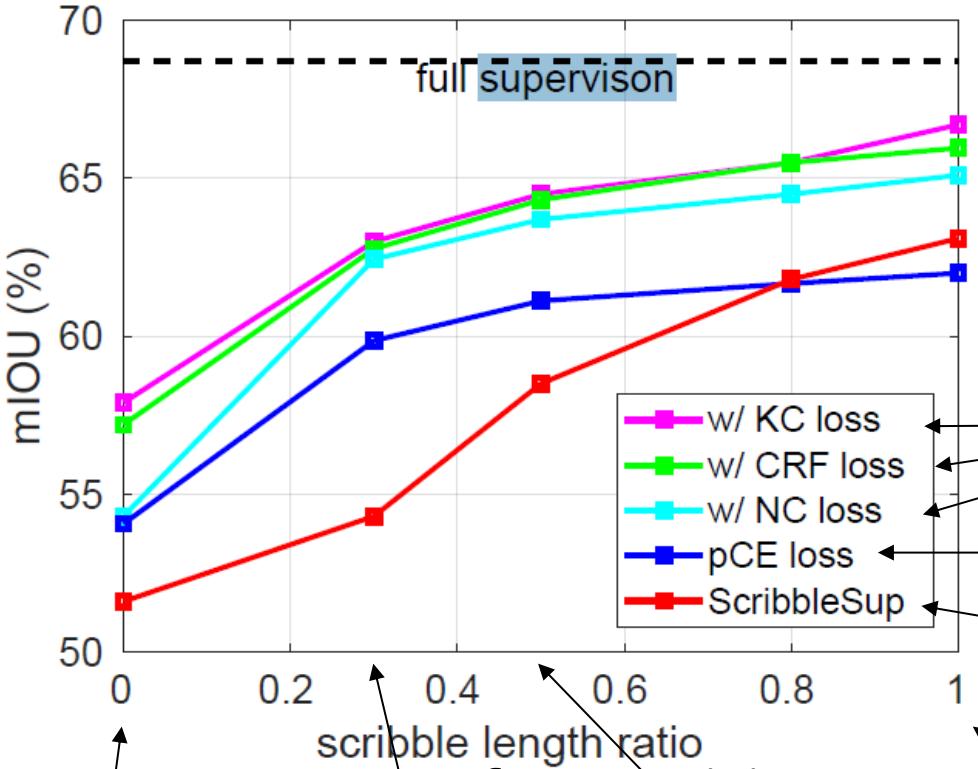
input

network prediction for class k during training $\bar{\sigma}_k^p$

regularization loss gradient $\frac{\partial R(\sigma)}{\partial \sigma_k}$

$$R(\sigma) = \sum_{pq \in \mathcal{N}} w_{pq} \cdot ||\bar{\sigma}^p - \bar{\sigma}^q||^2$$

Weakly-supervised training of CNN segmentation



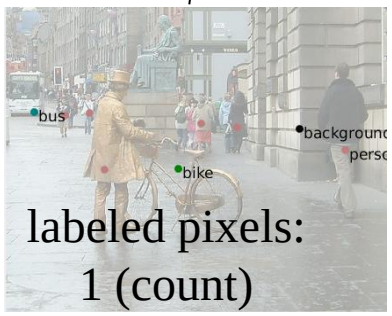
amount of supervision

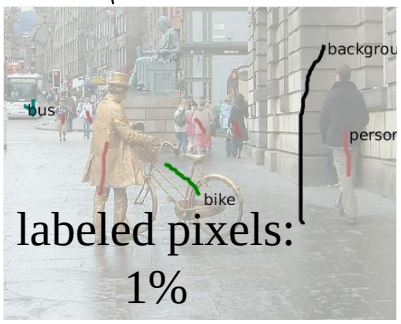
Representative plots
for different CNN architectures

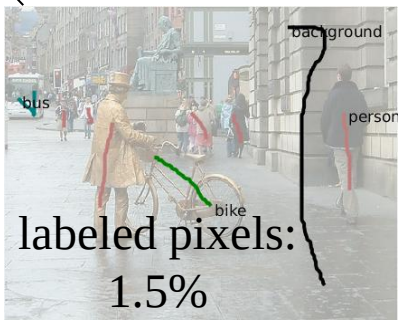
regularized losses

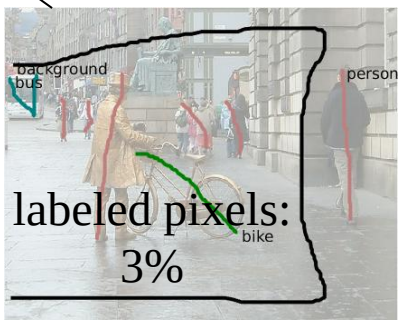
partial cross entropy (pCE) only

proposal generation
("fake" ground truth)



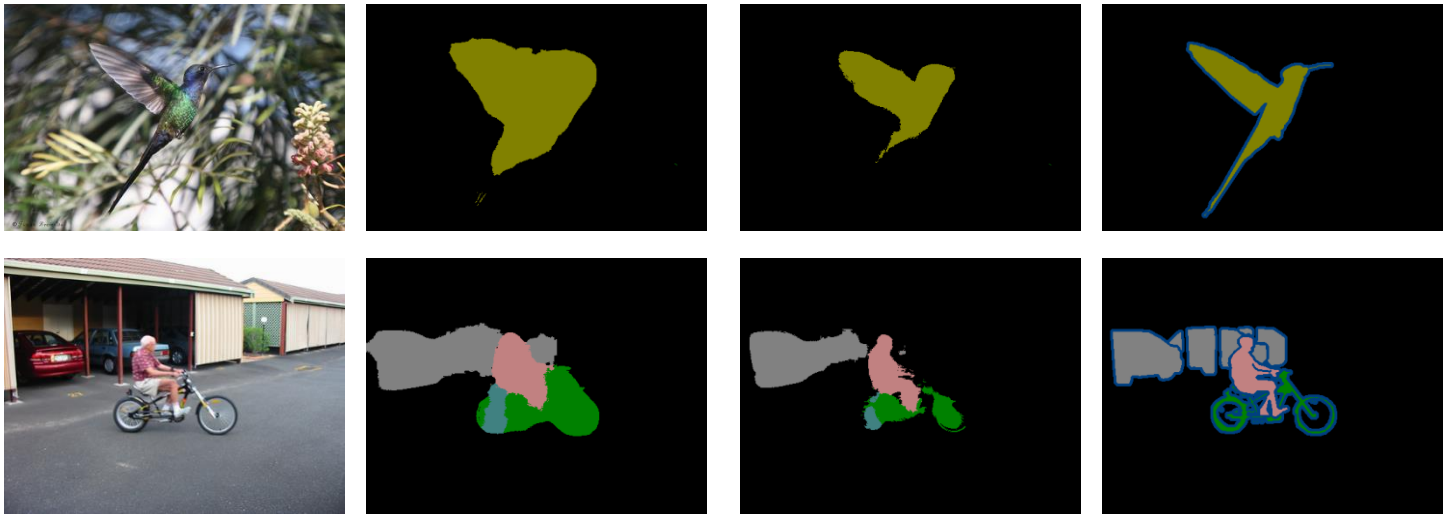






Adding Regularized Losses to pCE

partial
pixel-level labels
(**seeds**)



Test image | pCE loss | + pairwise regularization | ground truth

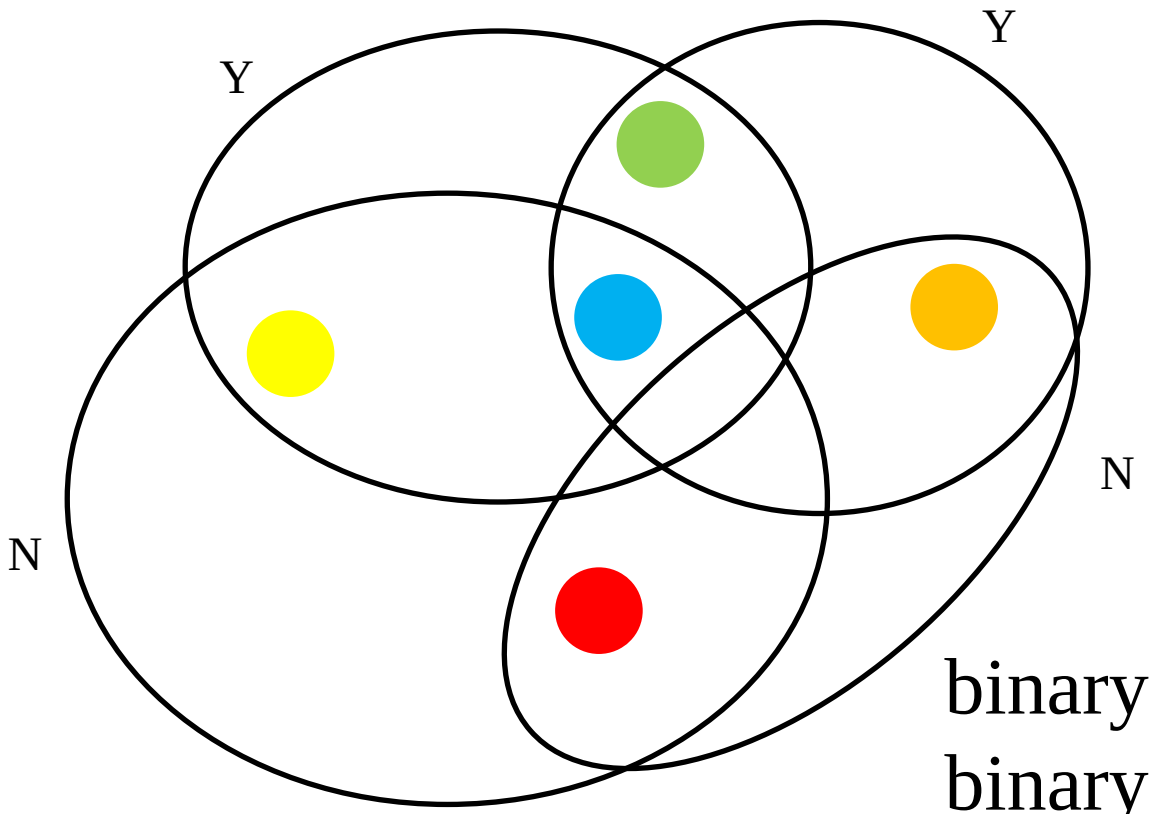
better edge alignment and smoothness

What if **image-level labels only** ?

First, consider a simple related example:

find working molecule (drug discovery)

instead of individual examples, training labels are available only for sets (bags) of examples



binary tags

binary classification

Multiple Instance Learning (MIL)

What if **image-level labels only** ?

For simplicity, assume pixel colors are discriminative enough features.

That is, to segment, we just need to learn **what color is sky, grass, and sand** ?

From these three images, we can segment pixels by matching **green to grass**, **blue to sky**, and **beige to sand**.

{ sky, grass, sand } **{ sky, sand }** **{ grass, sand }**

image-level tags

multi-class tags

multi-class classification

In general, segmentation network must learn BOTH

(deep) **discriminative pixel-level features** AND **their match with class tags**

What if **image-level labels only** ?

Example:
linear classifier $\sigma(WX)$
trained on three images using raw colors features (X = RGB) and *tag-consistency* loss

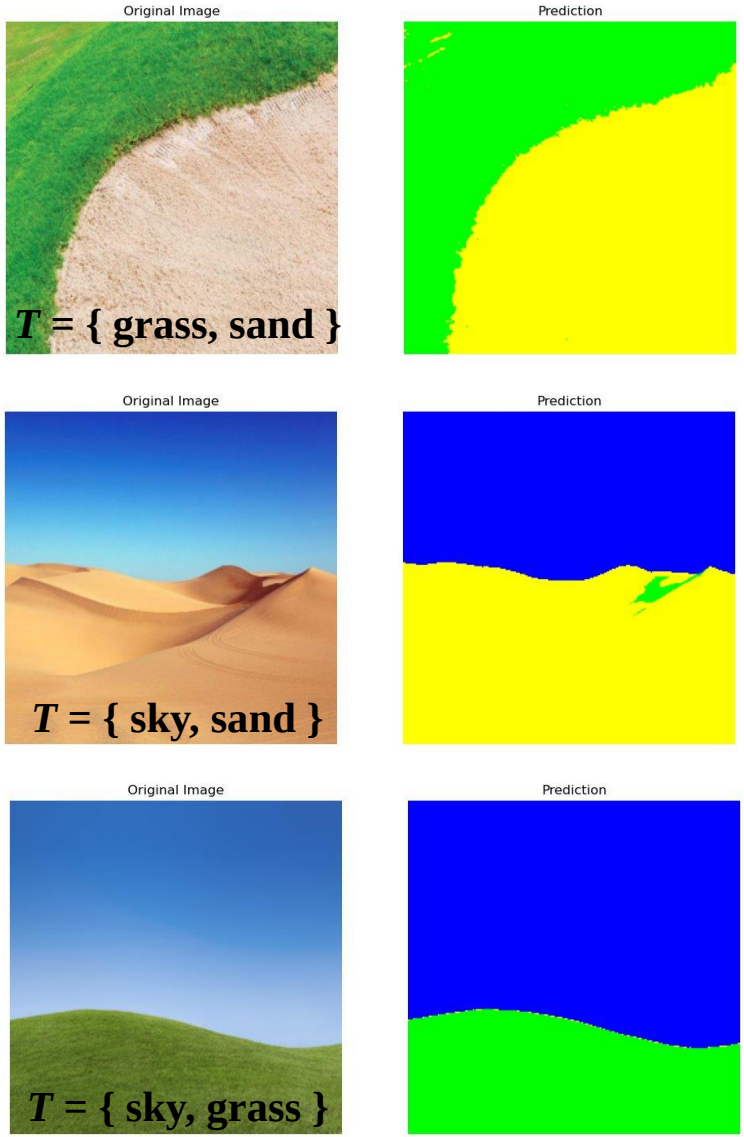
$$L(W) = -\sum_p \ln \underbrace{\left(\sum_{k \in T} \sigma_p^k \right)}_{\sigma_p^T}$$

This sum can be interpreted as prediction (or probability) that pixel p has one of the classes in T ➡ σ_p^T

the loss encourages $\sigma_p^T \to 1$

and $\sigma_p^{\neg T} := (1 - \sigma_p^T) \equiv \sum_{k \notin T} \sigma_p^k \to 0$

NOTE: practical problems on real image datasets:
a) "background" class is a trivial solution for all images
b) colors are semantically non-discriminative

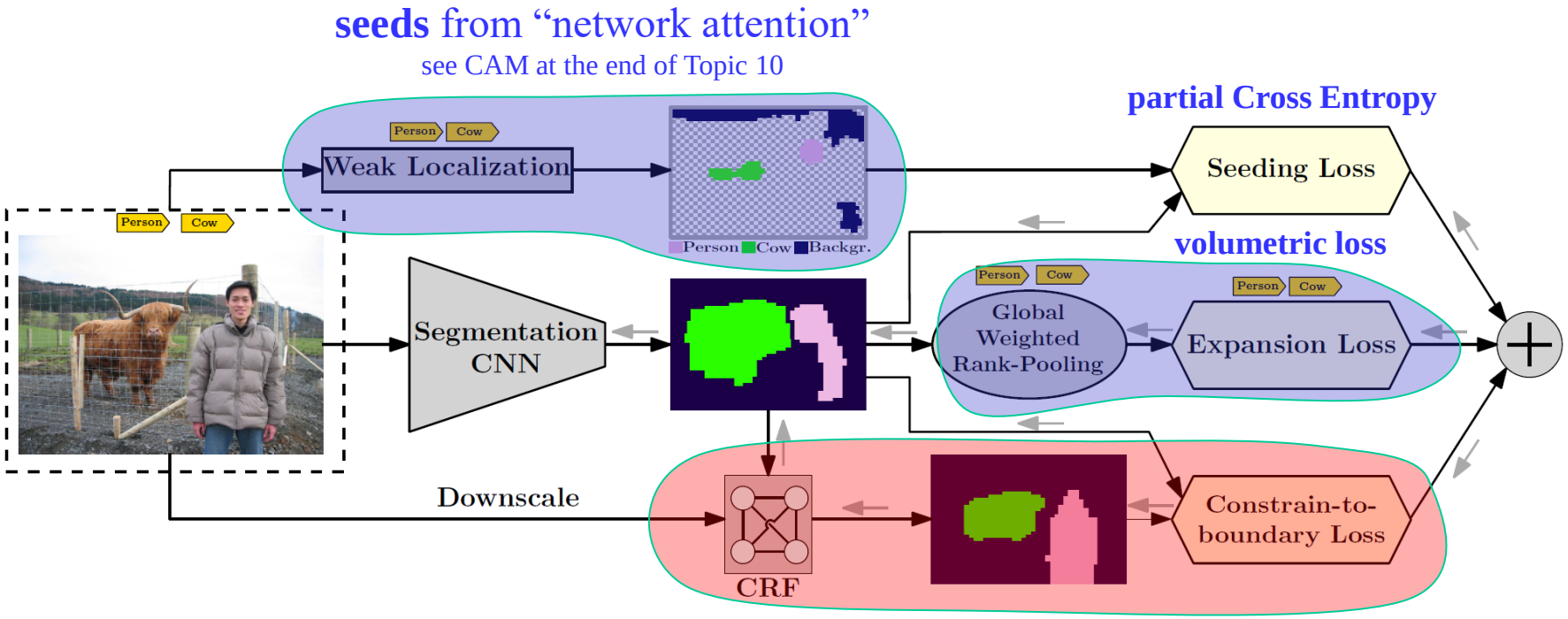


results generated by Jiahao Zhang

What if **image-level labels only** ?

Some ideas for real datasets:

[Kolesnikov & Lampert ECCV 2016]



Can be simplified using

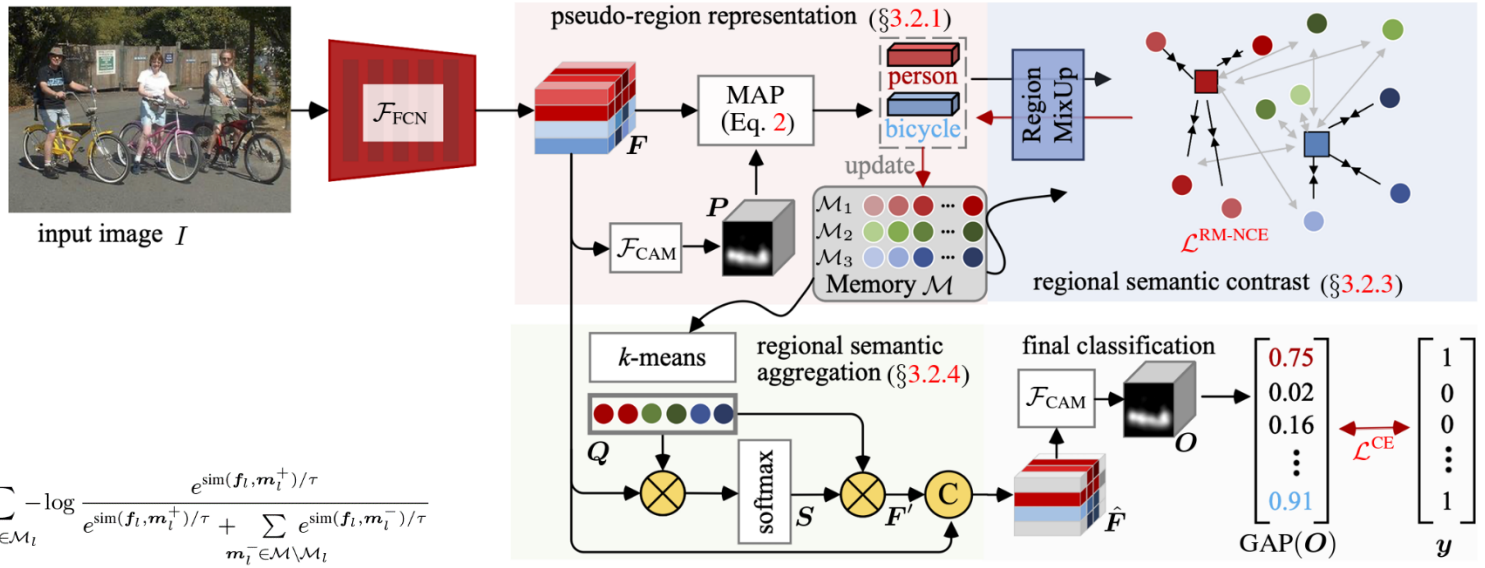
regularization loss

in the previous slides

What if **image-level labels only** ?

Some ideas for real datasets:

Zhou, Tianfei, et al. "Regional semantic contrast and aggregation for weakly supervised semantic segmentation." CVPR 2022.



$$\mathcal{L}_l^{\text{NCE}}(\boldsymbol{f}_l, y_l) = \frac{1}{|\mathcal{M}_l|} \sum_{\boldsymbol{m}_l^+ \in \mathcal{M}_l} -\log \frac{e^{\text{sim}(\boldsymbol{f}_l, \boldsymbol{m}_l^+)/\tau}}{e^{\text{sim}(\boldsymbol{f}_l, \boldsymbol{m}_l^+)/\tau} + \sum_{\boldsymbol{m}_l^- \in \mathcal{M} \setminus \mathcal{M}_l} e^{\text{sim}(\boldsymbol{f}_l, \boldsymbol{m}_l^-)/\tau}}$$

Contrastive Learning for Features

More recently, the state of the art for segmentation from image-level supervision is approaching full pixel-level supervision.

UNIVERSITY OF WATERLOO

Can use unsupervised low-level vision

Full pixel-accurate ground truth is practically impossible. But the **shortage of targets** can be compensated by domain specific losses from low-level vision, e.g.

- volumetric constraints
- boundary regularization (in segmentation)
- spatial regularity of the output (depth or motion field smoothness)
- photo-consistency (e.g. in stereo and 3D reconstruction)

“self-supervision”

no oracle (ground truth) is used

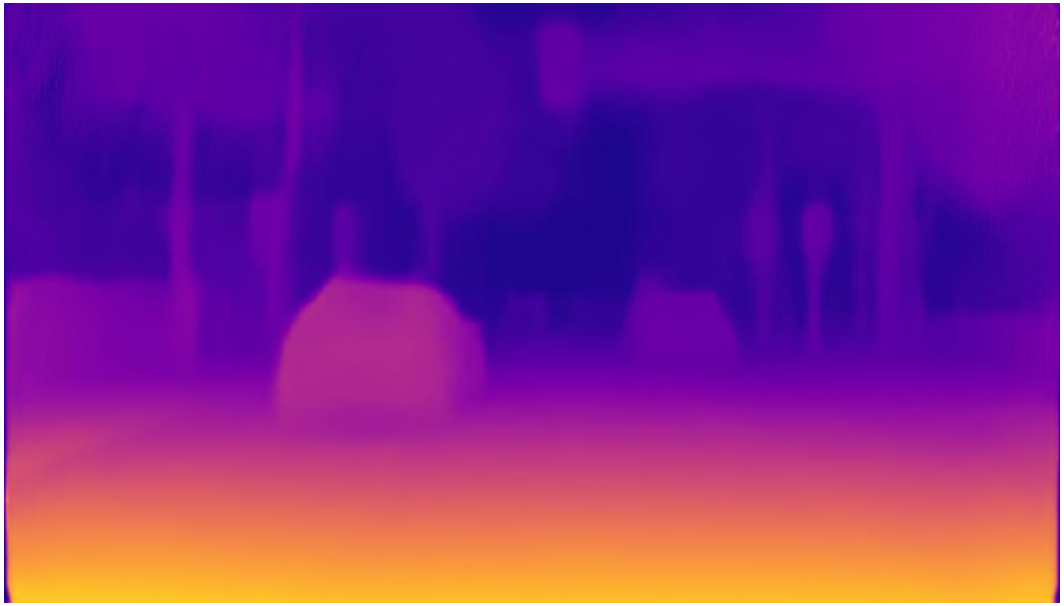
CNN (FCN) for pixel labeling problems

where … full pixel-accurate ground truth is typically impossible

- semantic segmentation - classification
- depth estimation
- motion estimation
- restoration
- denoising
- inpainting
- e.t.c.

- regression problem (no soft-max), but still can use networks (e.g. FCN) to produce output (predictions) with spatial resolution

Example: Depth from **Single View**



Unsupervised Monocular Depth Estimation with Left-Right Consistency

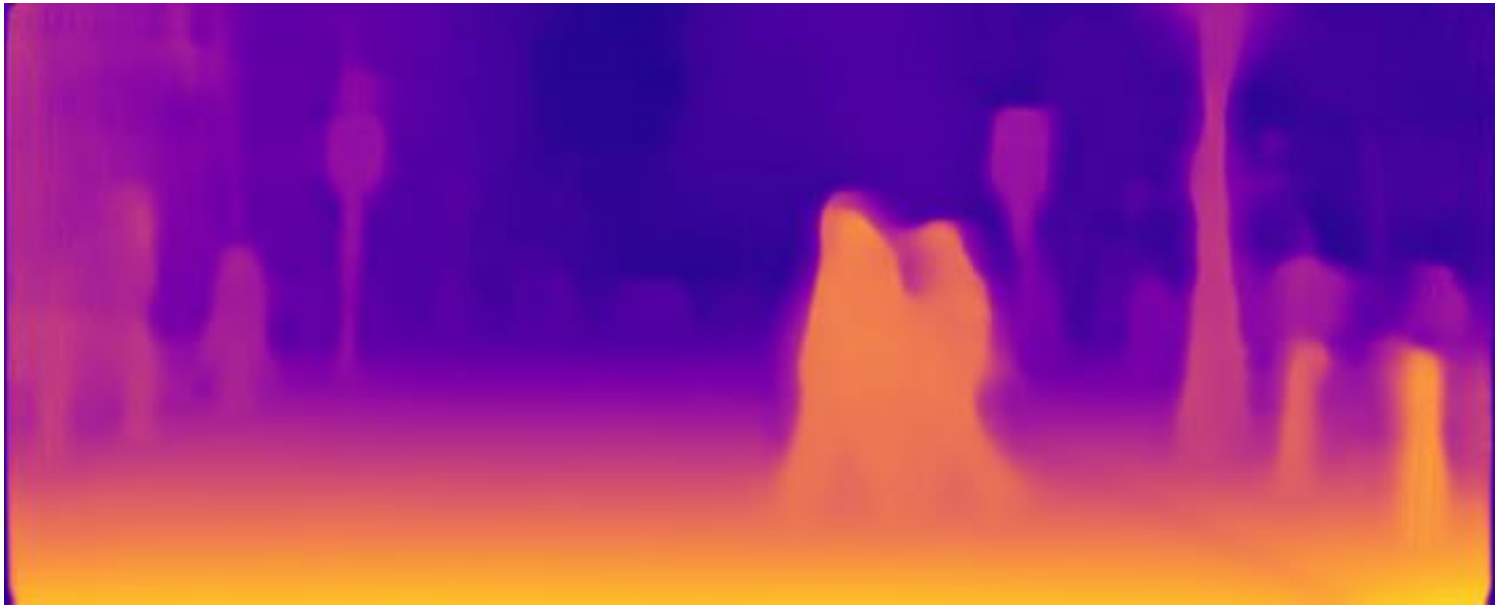
Godard, Aodha, Brostow
CVPR 2017

Example: Depth from **Single View**

Unsupervised Monocular Depth Estimation with Left-Right Consistency

Godard, Aodha, Brostow
CVPR 2017

Example: Depth from **Single View**

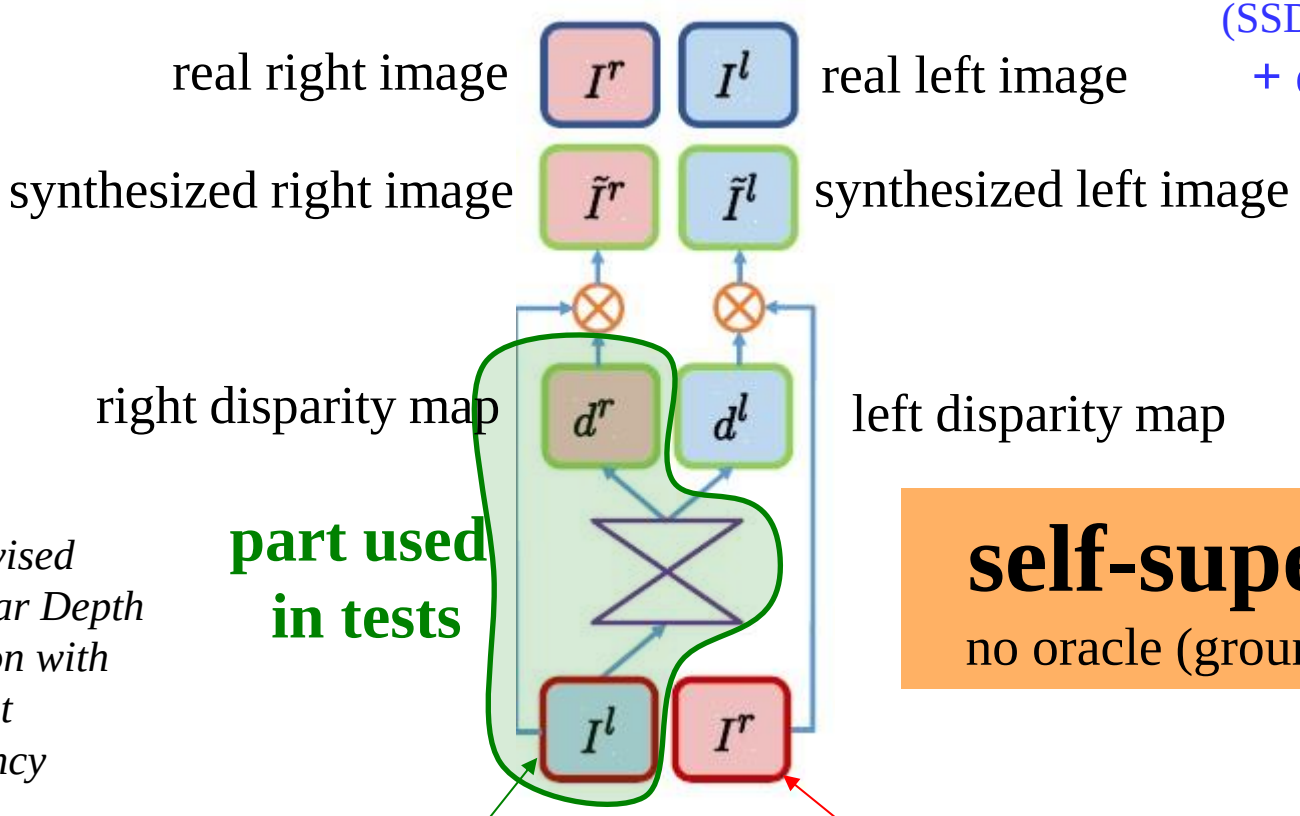


Unsupervised Monocular Depth Estimation with Left-Right Consistency

Godard, Aodha, Brostow
CVPR 2017

Example: Depth from **Single View**

How does this work?



Loss:

photo-consistency

(SSD for synthetic - real)

+ disparity maps regularity

self-supervision:

no oracle (ground truth) is used

part used in tests

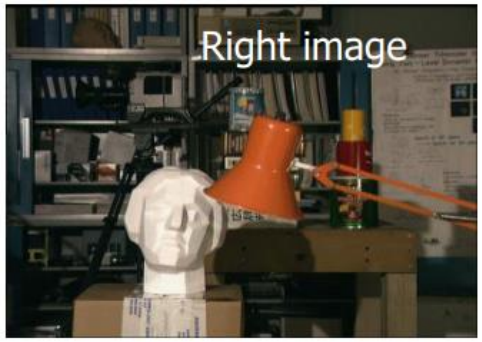
network uses only one (left) image as its input

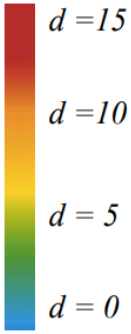
right (stereo) image is needed only during training

Unsupervised Monocular Depth Estimation with Left-Right Consistency

Godard, Aodha, Brostow
CVPR 2017

Remember Loss for Stereo (topic 8)





$$E(\mathbf{d}) = \sum_{p \in G} \underbrace{D_p(d_p)}_{\| \; |I_p - I'_{p \oplus d_p}|} + \sum_{\{p,q\} \in N} \underbrace{V(d_p, d_q)}_{\| \; w_{pq} \cdot |d_p - d_q|}$$

photo consistency

spatial coherence

disparity map regularity

NOTE: we can use gradient descent over NN parameters *w* to optimize disparity map **d** just for a given stereo pair (*I, I'*) as an alternative to optimization in topic 8 (e.g. graph cut). May benefit from NN's "inductive bias".

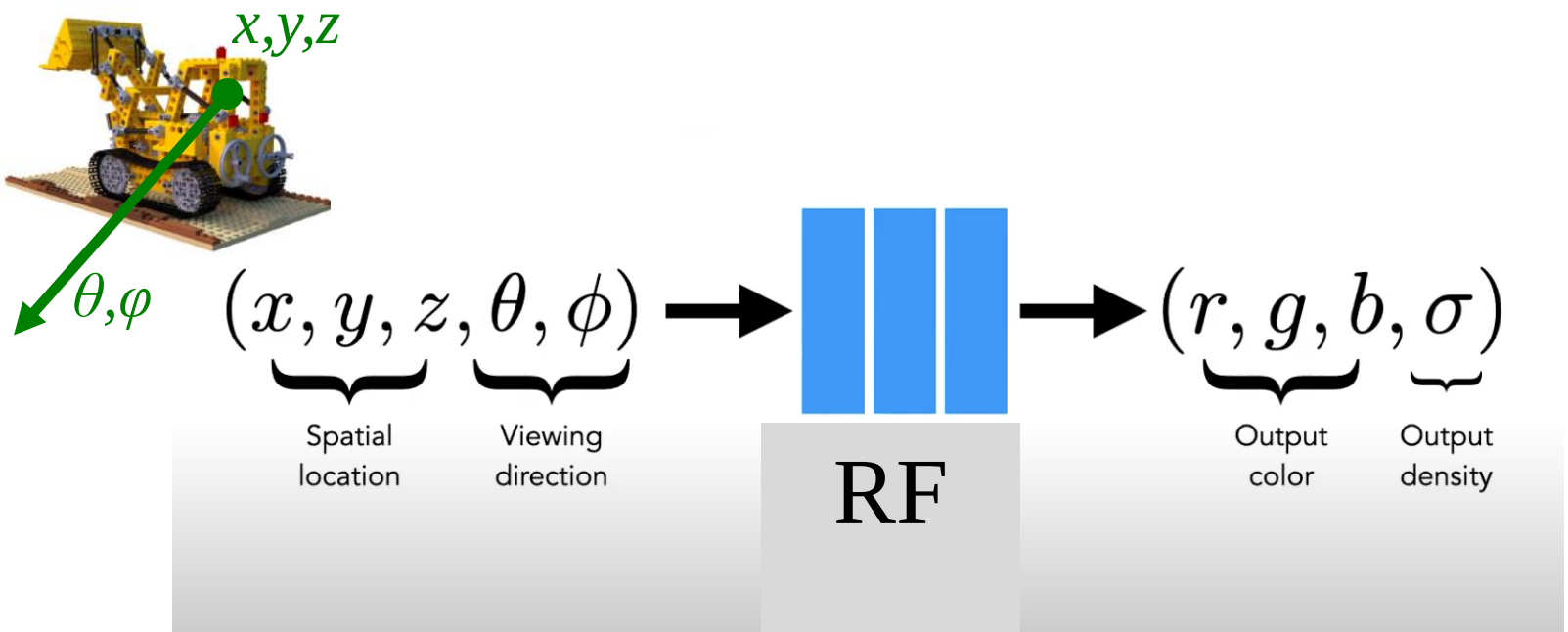
In fact, NN as optimization tool is used for 3D reconstruction – see NeRF.

Towards 3D reconstruction - NeRF

Instead of learning a model (function) producing depth map

$$depth\ map\ =\ f_\theta\,(im1,...)$$ from one or many images

one can "learn" the ***radiance field*** *function*



Problem formulation: given N views, "learn" RF specific to the scene

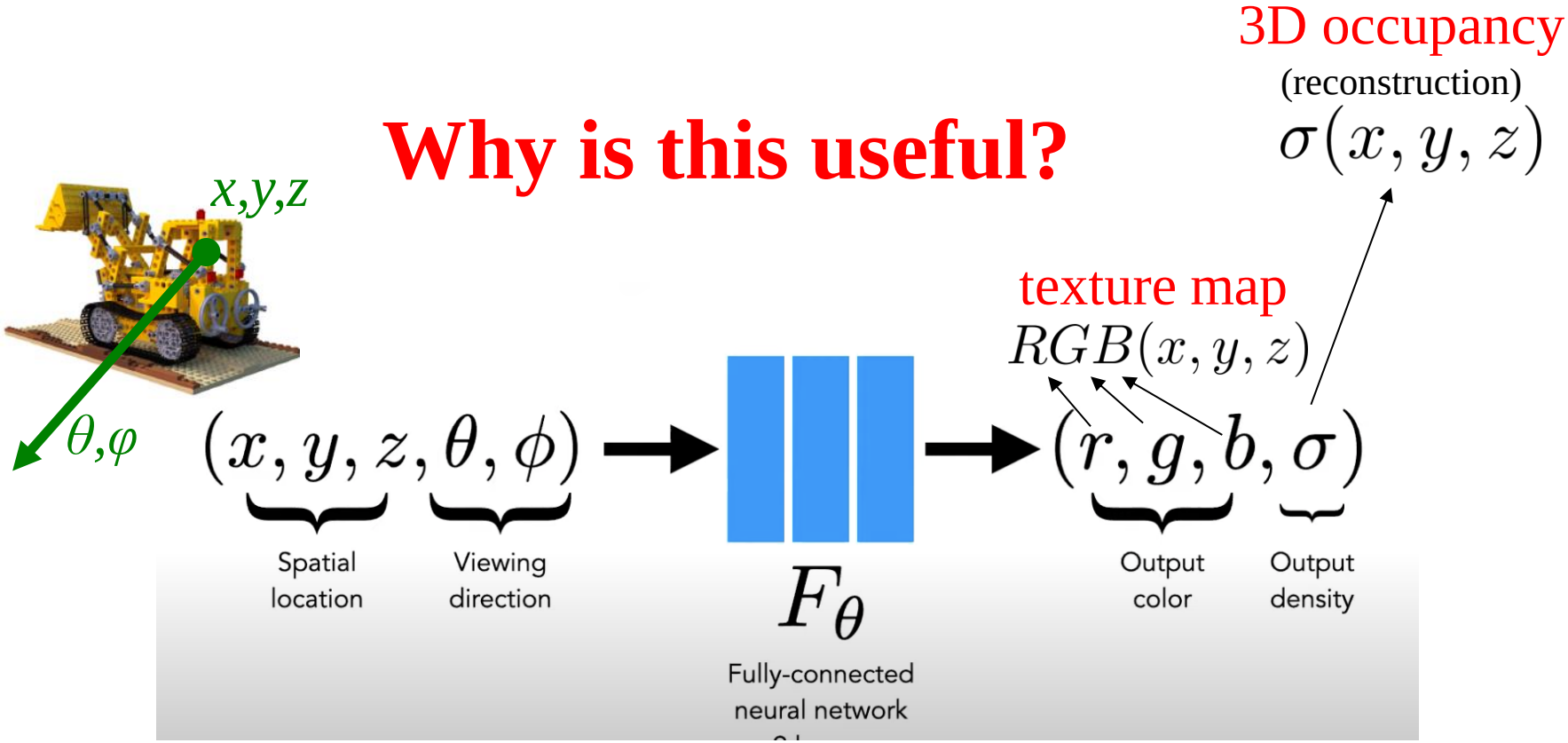
Illustration credits: B.Mildenhall, P.Srinivasan, M.Tancik, J.Barron, R,Ramamoorthi, R.Ng

NeRF

Instead of learning a model (function) producing depth map

$$depth\ map\ =\ f_\theta\,(im1,...)$$ from one or many images

one can "learn" the ***radiance field*** *function*



Problem formulation: given N views, "learn" RF specific to the scene

Illustration credits: B.Mildenhall, P.Srinivasan, M.Tancik, J.Barron, R,Ramamoorthi, R.Ng

NeRF

Assuming images from K cameras with known positions (projection matrices) one can one "learn" the ***radiance field*** function RF_θ **specific to a given scene**

First, consider how RF defines pixel colors in given camera k

Color rendering model for ray $r(t) = p + t\,\delta t$ defined by RF_θ for pixel p in camera k

RF-projected color at pixel

$$p \overset{\forall t}{=} P_k\, r(t)$$

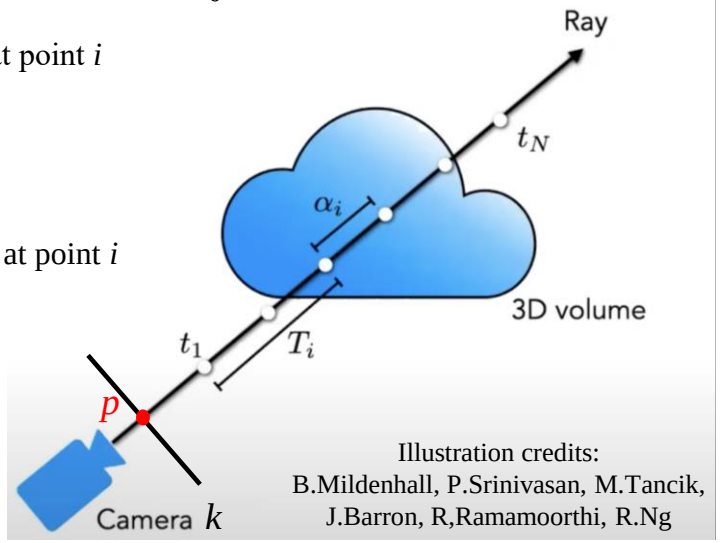
known projection matrix for camera k

color "density" or strength of RF at point i

$$C^k_{RF}(p) \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

color output by RF at point i

Transparency (or "density" integral) along the ray before point i



Light contributed by ray segment i

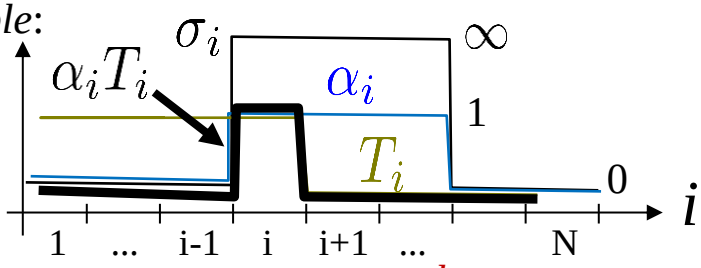
$$\alpha_i = 1 - e^{-\sigma_i\,\delta t}$$

density output by RF

Transparency along the ray before segment i

$$T_i = \prod_{j=1}^{i-1}(1-\alpha_j) \;\equiv\; (1-\alpha_{i-1})\,T_{i-1}$$

Example:



Thus, for any pixel p in any camera P_k we can get RF-projected colors $C^k_{RF}(p)$

NeRF

Assuming images from K cameras with known positions (projection matrices) one can one "learn" the ***radiance field*** function RF_θ **specific to a given scene**

Training Loss: **photo-consistency** between *RF* projected colors and colors I_p observed in K images

$$\sum_k \sum_{p \in I_k} \| I_p - C^k_{RF_\theta}(p) \|^2$$

NeRF model parameters

self–supervision

learned RF gives arbitrary view rendering, 3D object shape and its mapped texture

see cool demos at www.matthewtancik.com/nerf

NeRF

Assuming images from K cameras with known positions (projection matrices) one can one "learn" the ***radiance field*** function $\boldsymbol{RF_\theta}$ **specific to a given scene**

Training Loss: **photo-consistency** between *RF* projected colors and colors I_p observed in K images

$$\sum_k \sum_{p \in I_k} \| I_p - C^k_{RF_\theta}(p) \|^2$$

NeRF model parameters

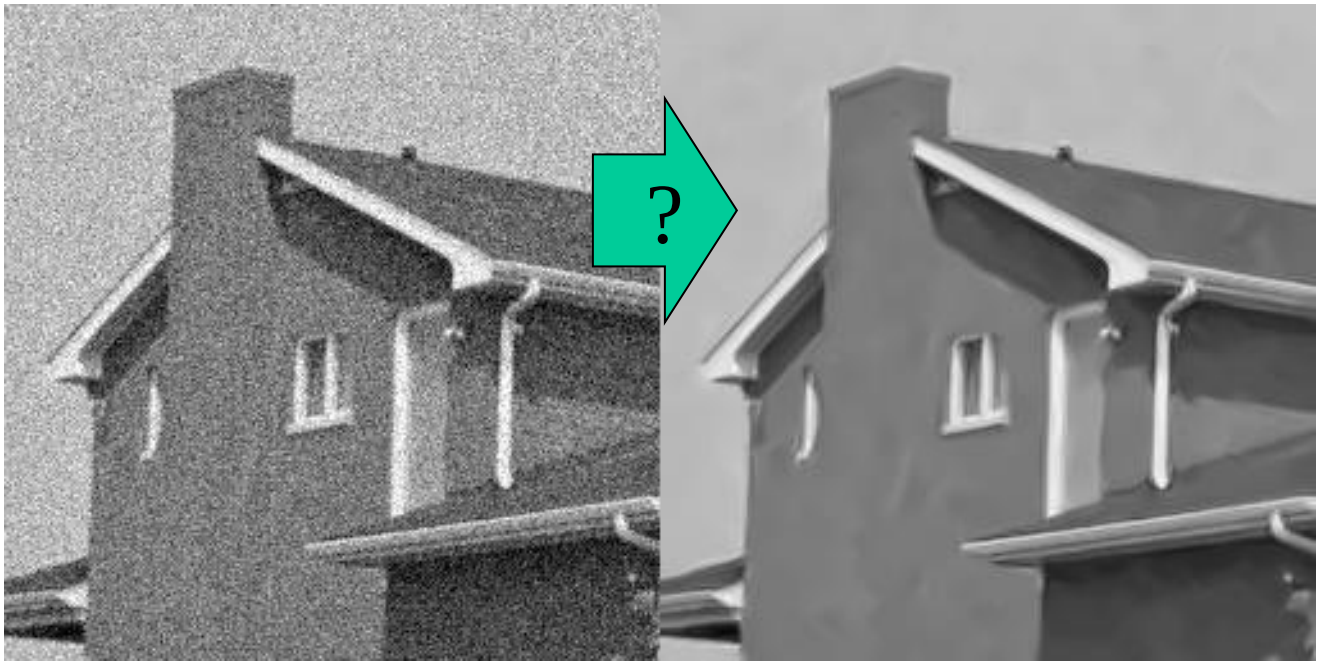
NOTE: NeRF model is **overfit** to given K images with known poses.

$$RF_\theta(x, y, z \| im1, im2, \ldots, imK)$$

Unlike mono-depth example, NeRF model should be recomputed for each new scene. This is different from traditional network models, e.g. $MonoDepth_\theta(im)$ **that work for any image (after training).**

More self-supervision: *image denoising*

noisy image restored/denoised image



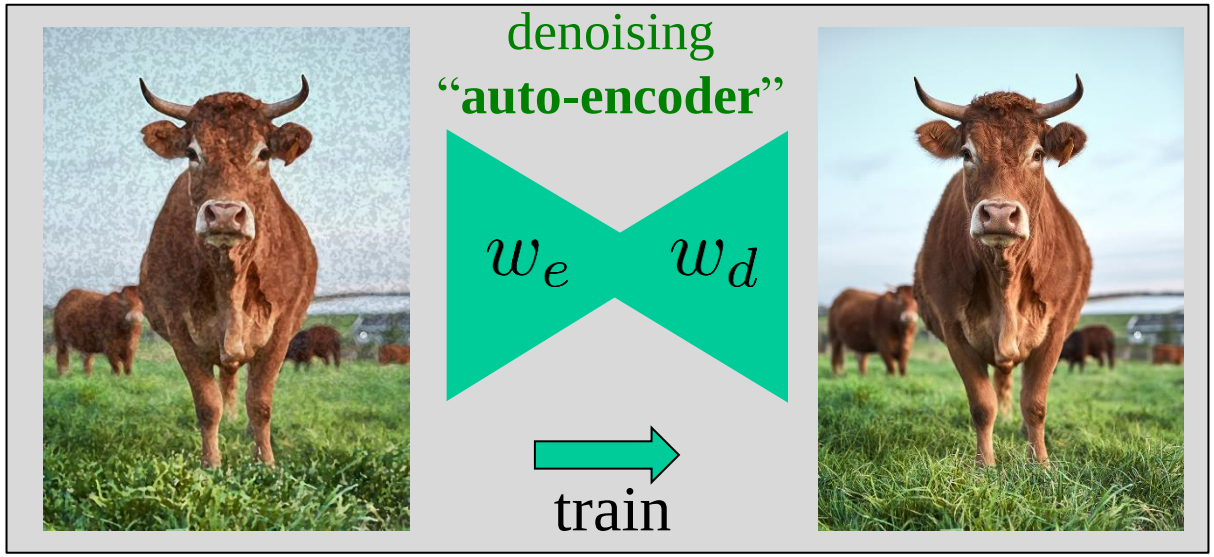
Remember *mean* and *median filtering* (see Topic 3)

- not so easy to do well (e.g. avoid boundary blurring)
- now we will “learn” how to do it

Example: image denoising

add noise

corrupted image $I+\varepsilon$ ← good image I



denoised image $I_w(I+\varepsilon)$

$$\min_w \sum_i \|I_w(I_i+\epsilon_i)-I_i\|^2$$

self-supervision

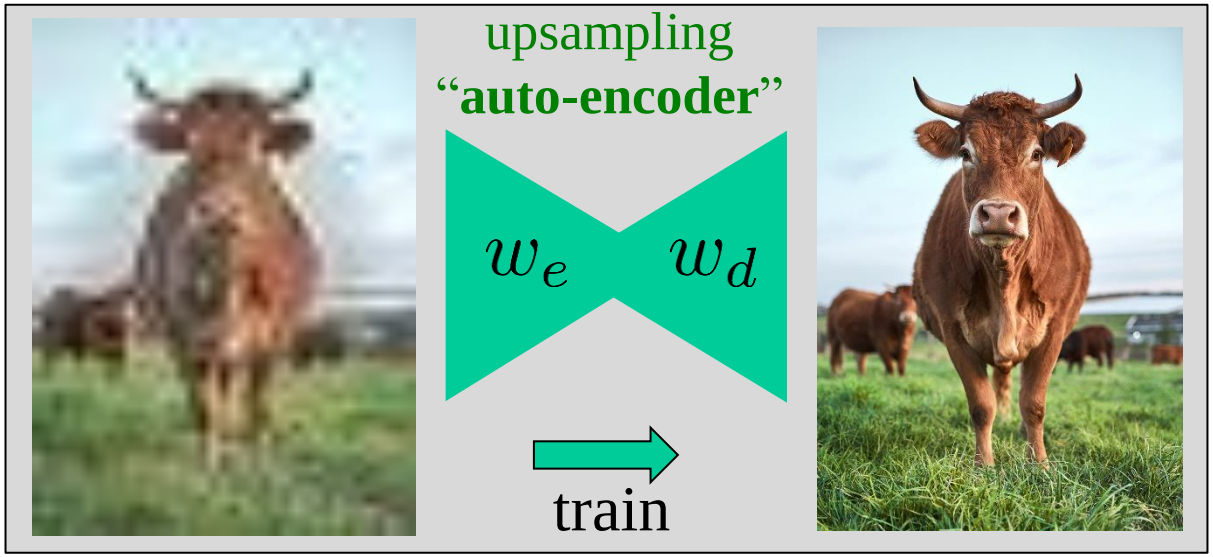
using any good images for training

Example: super-resolution

blur/downsample + noise

low-res image
$c(I)+\varepsilon$

good image
I



upsampled image
$I_w(c(I)+\varepsilon)$

self-supervision
using any good images for training

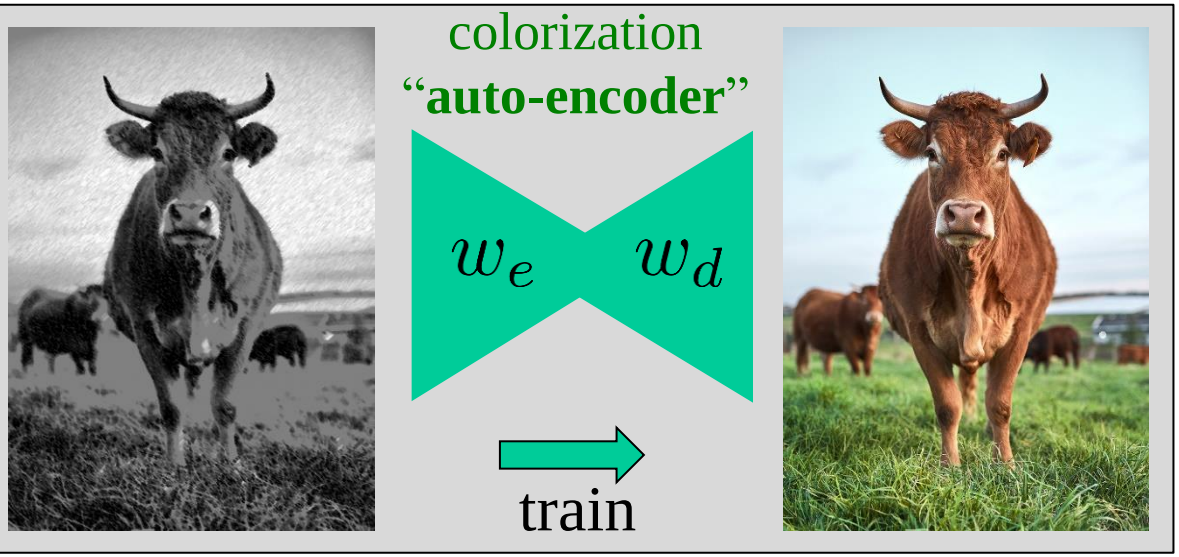
$$\min_w \sum_i \|I_w(c(I_i)+\epsilon) - I_i\|^2$$

Example: image colorization

remove color + noise

gray image
$g(I)+\varepsilon$

good image
I



colored image
$I_w(g(I)+\varepsilon)$

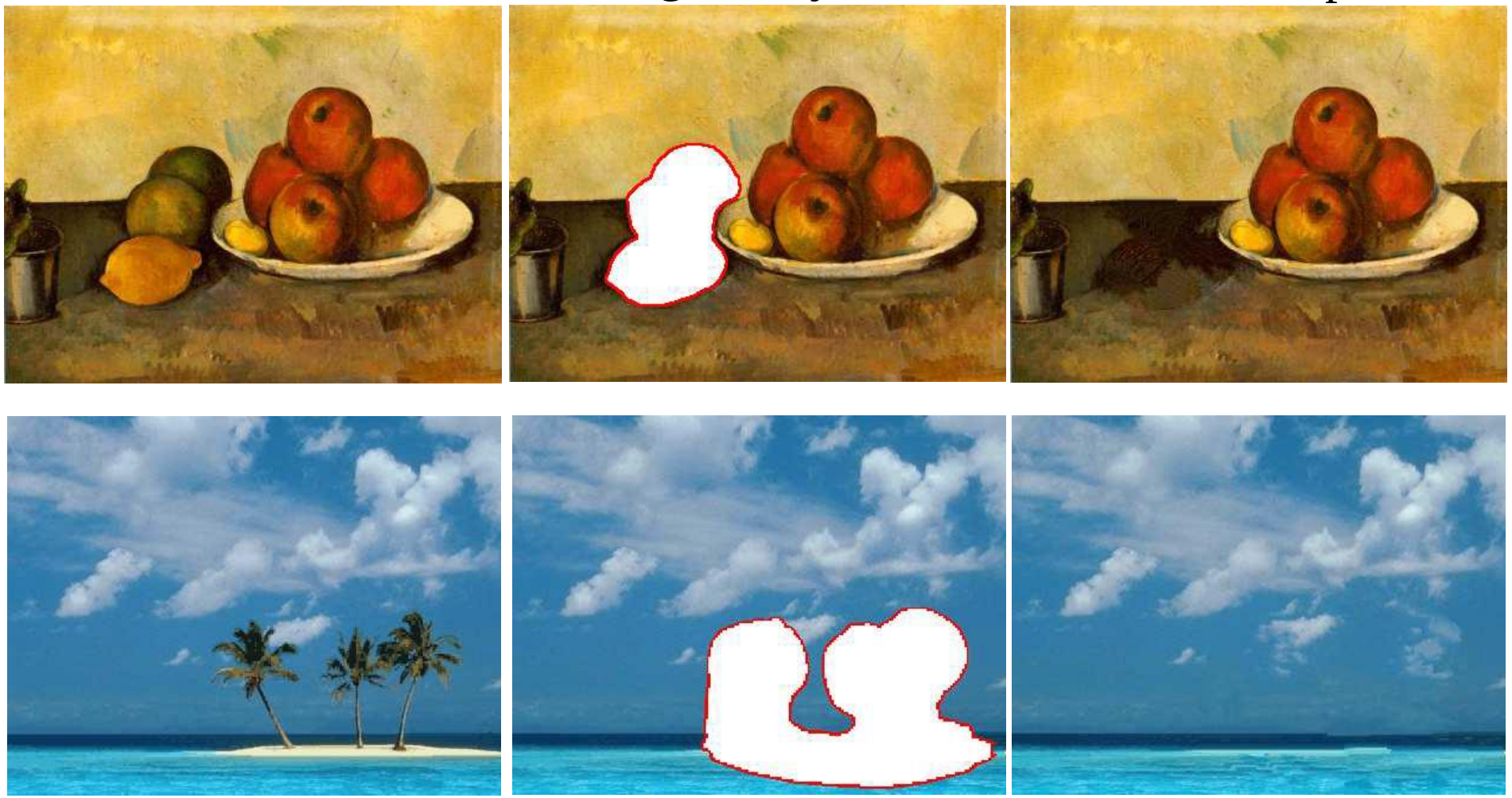
self-supervision
using any good images for training

$$\min_w \sum_i \|I_w(g(I_i)+\epsilon) - I_i\|^2$$

Example: image inpainting (a.k.a. object removal)

mask given by a user

hole filled / inpainted



examples from “Region Filling and Object Removal by Exemplar-Based Image Inpainting”
A. Criminisi, P. Perez and K. Toyama, TPAMI 2004

Example: image inpainting (a.k.a. object removal)

mask given by a user

hole filled / inpainted

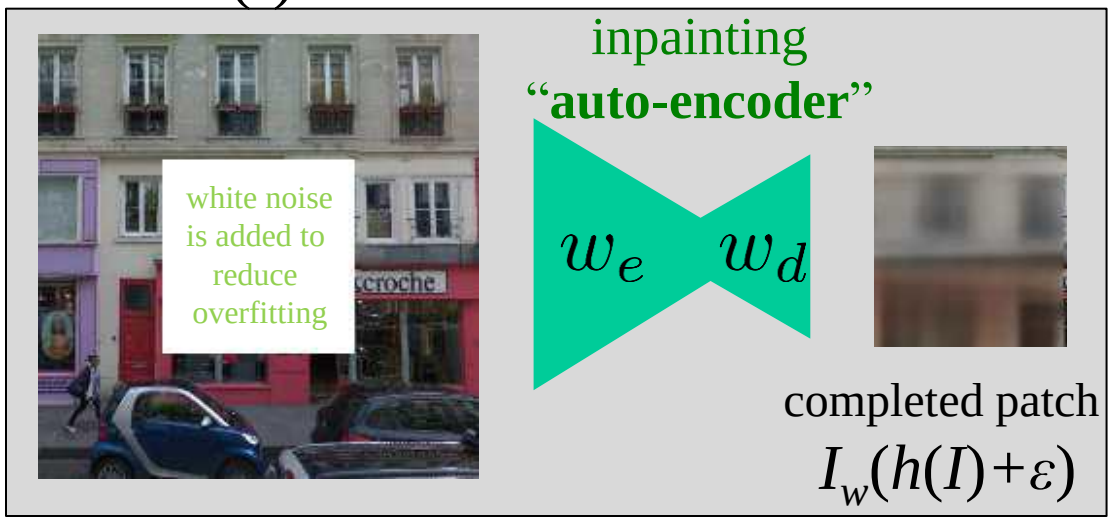
basic approach for **rectangular holes**

Example: image inpainting (a.k.a. object removal)

image with a hole
$h(I)$

remove a patch + noise

good image
I



self-supervision
using full images for training

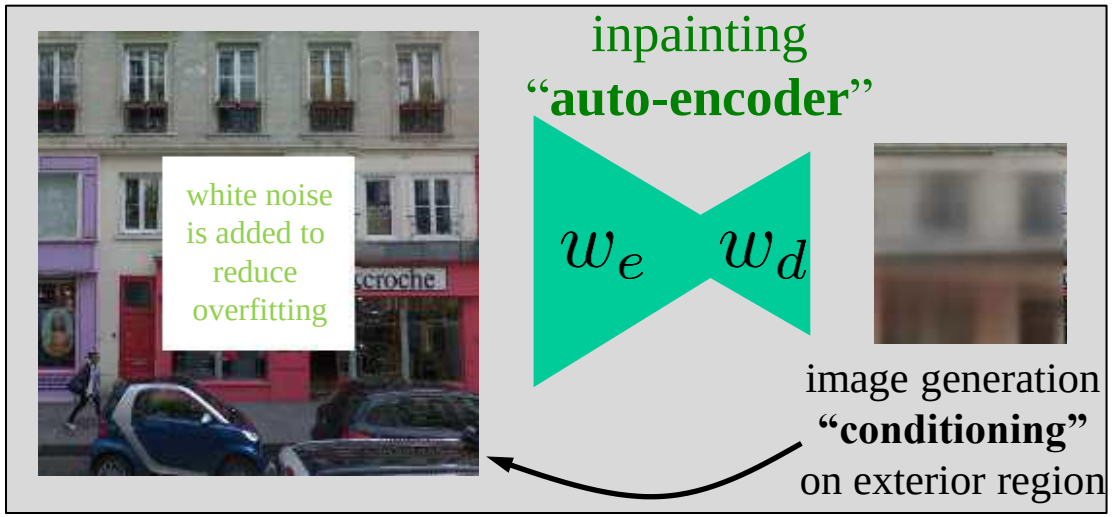
$$\min_w \sum_i \|I_w(h(I_i)+\epsilon) - I_i\|^2$$

L2 or L1 losses for reconstruction errors

"Context Encoders: Feature Learning by Inpainting"
D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. Efros, CVPR 2016

basic approach for **rectangular holes**

Example: image inpainting (a.k.a. object removal)



L2 or L1 losses for reconstruction errors produce blurry results

self-supervision

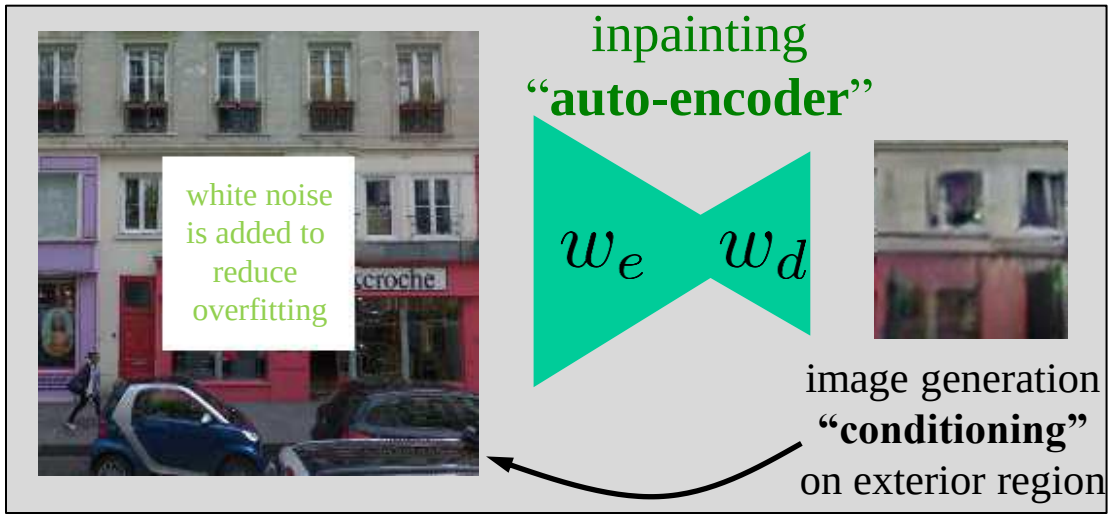
using full images for training

"Context Encoders: Feature Learning by Inpainting"

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. Efros, CVPR 2016

basic approach for **rectangular holes**

Example: image inpainting (a.k.a. object removal)



L2 or L1 losses for reconstruction errors produce blurry results

self-supervision
using full images for training

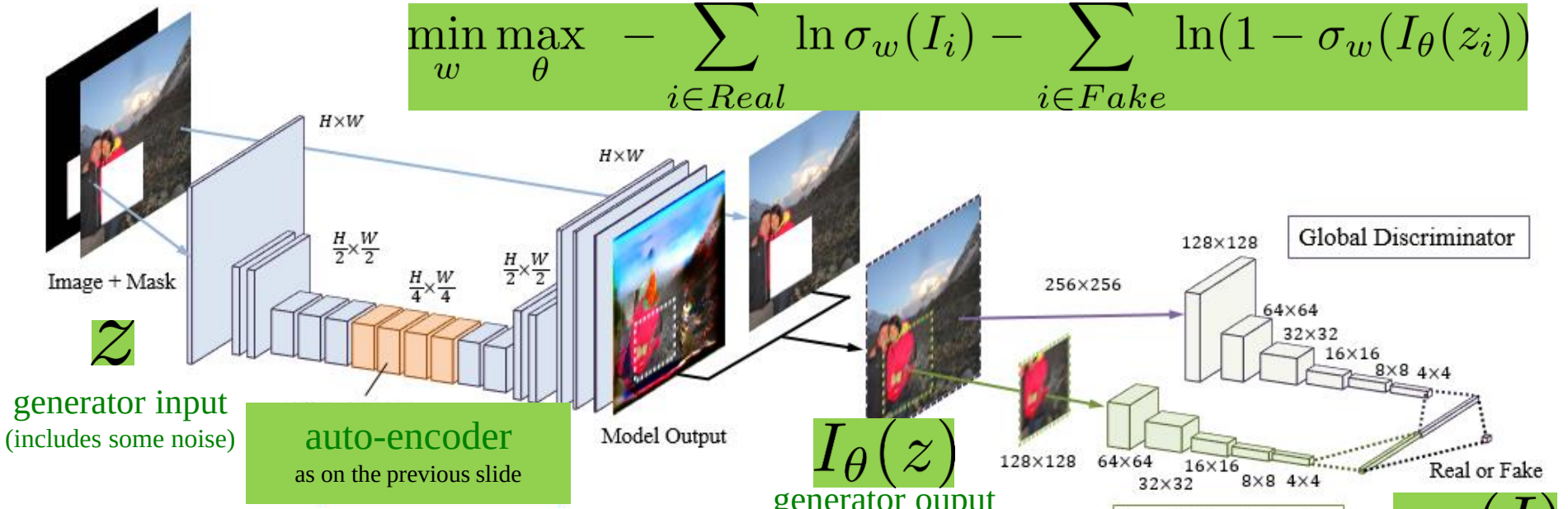
More realistic results are obtained with **adversarial training**

"Context Encoders: Feature Learning by Inpainting"
D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. Efros, CVPR 2016

Example: image inpainting (a.k.a. object removal)

Generative Adversarial Network (GAN) example

$$\min_w \max_\theta \; - \sum_{i \in Real} \ln \sigma_w(I_i) - \sum_{i \in Fake} \ln(1 - \sigma_w(I_\theta(z_i)))$$



self-supervision

using a mix of real and inpainted images

- "Discriminator" is shown either true or generated image. It has to tell "real" from "fake".
- "Generator" (completion network, auto-encoder) tries to fool the discriminator.

S. Iizuka, E. Simo-Serra, H. Ishikawa "Globally and locally consistent image completion"
ACM Transactions on Graphics (ToG), 2017

Generative network models

Typically use **self-supervision** to train auto-encoder networks to generate images for **classical computer vision** problems like *image denoising*, *inpainting*, *super-resolution*, and many "**graphic arts**" problems like *text-to-image*, *text-to-video* etc.

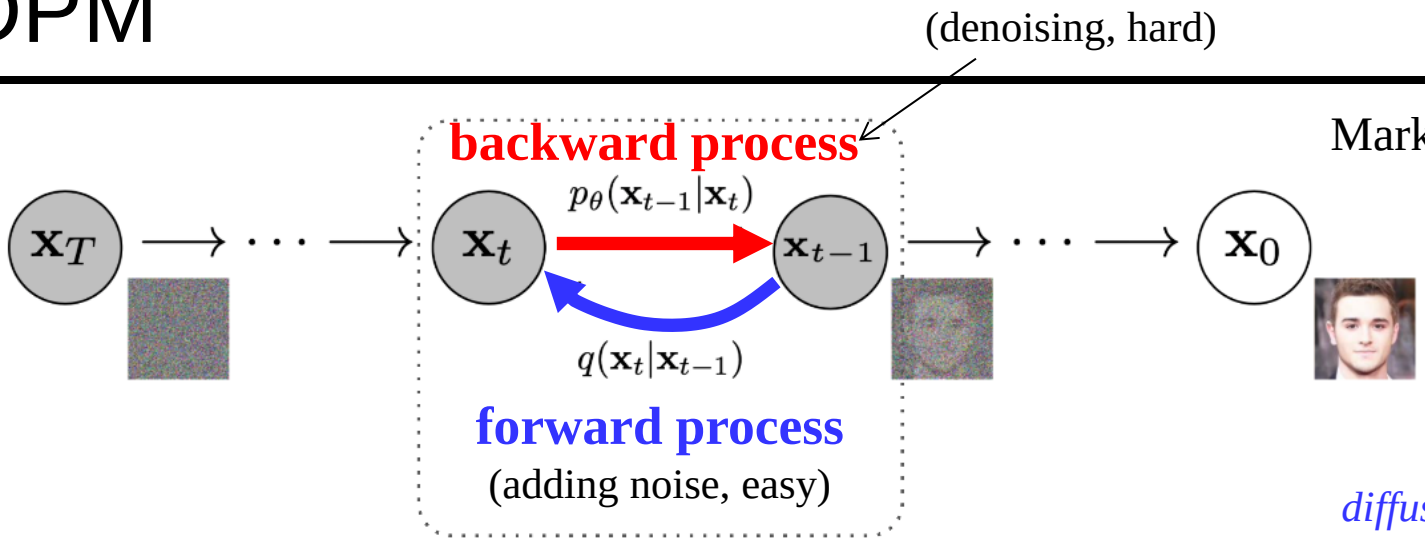
typically, **white noise** (sampling from Gaussian distribution) is part of input

- Auto-encoder with adversarial training (GAN)
  - hard to train (complex min-max optimization), prone to overfitting
- Variational auto-encoder (VAE)
  - latent space (AE bottleneck) is regularized to have features with $N(0,I)$ distribution. At test time, **decoder generates images from white noise**.
  - uses (variational inference) math related to EM algorithm for GMMs
- Diffusion models **more in CS480/680**
  - inject noise at each level and learn to *reverse* it restoring data from noise
  - at test time, **the network generates images from white noise**

we will discuss *Denoising Diffusion Probabilistic Models* (DDPM)…

[J. Ho, A. Jain, P. Abbeel, NeurIP 2020]

DDPM



diffusion rate

Forward process: adding Gaussian noise in very small steps ($\beta_t << 1$)

Let $\alpha_t := (1-\beta_t) \lesssim 1$ and $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i \to 0$

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}$$

$$= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \qquad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

(recursively derived)

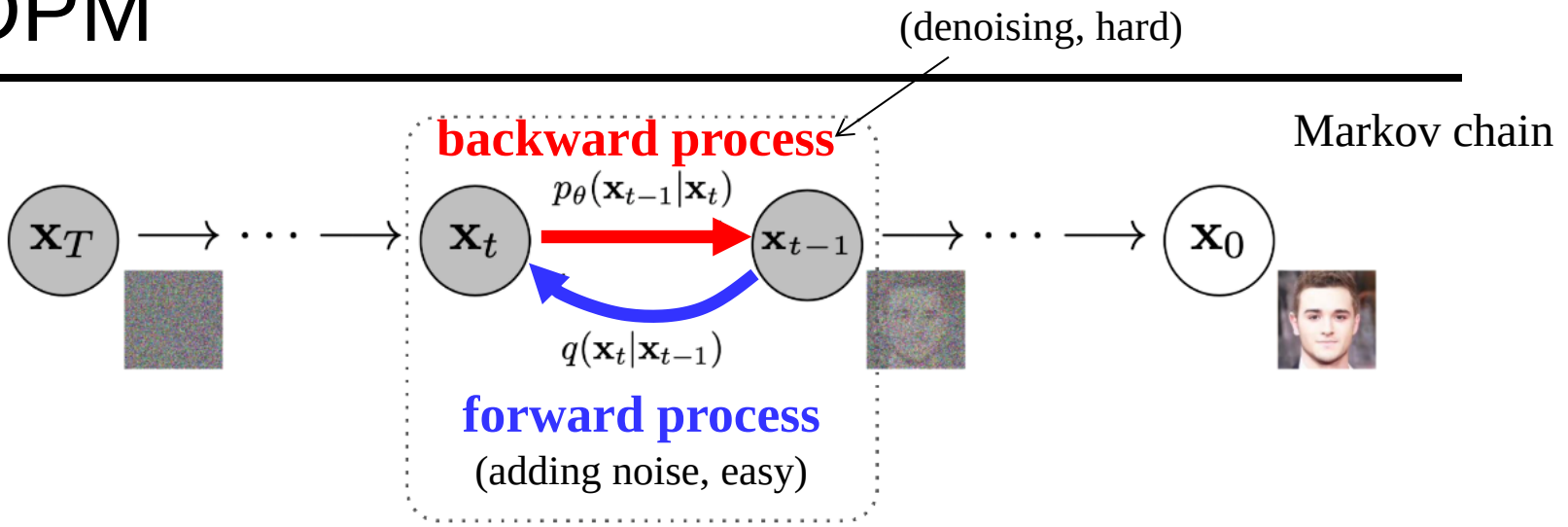
when $t \to \infty$, x_t converges to isotropic Gaussian noise

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

mean covariance

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

DDPM



Backward process: $p(x_{t-1} \mid x_t)$ is not easy to estimate, so we learn it as a parametric model ~~$p_\theta(x_{t-1} \mid x_t)$~~ $\epsilon_\theta(\mathbf{x}_t, t)$

denoising model

Training from existing images :

1: **repeat**

2: select random training image $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

3: select random layer t

4: generate white noise ε and recursively "corrupted" images

$\mathbf{x}_t = \sqrt{\alpha_t}\ \mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\ \epsilon_{t-1}$ (as on the previous slide)

5: learn to "denoise", take one gradient descent step on SSD

$$\min_\theta \left\| \left(\mathbf{x}_{t-1} - \frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\ \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}}\right) \right\|^2$$

$$\propto \|\epsilon_{t-1} - \epsilon_\theta(\mathbf{x}_t, t)\|^2$$

i.e. learn to predict the noise

6: **until** "converged"

Sampling new images :

1: generate white noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

2: **for** $t = T, \ldots, 1$ **do**

3: use trained models $\epsilon_\theta(\mathbf{x}, \mathbf{t})$ to recursively generate images

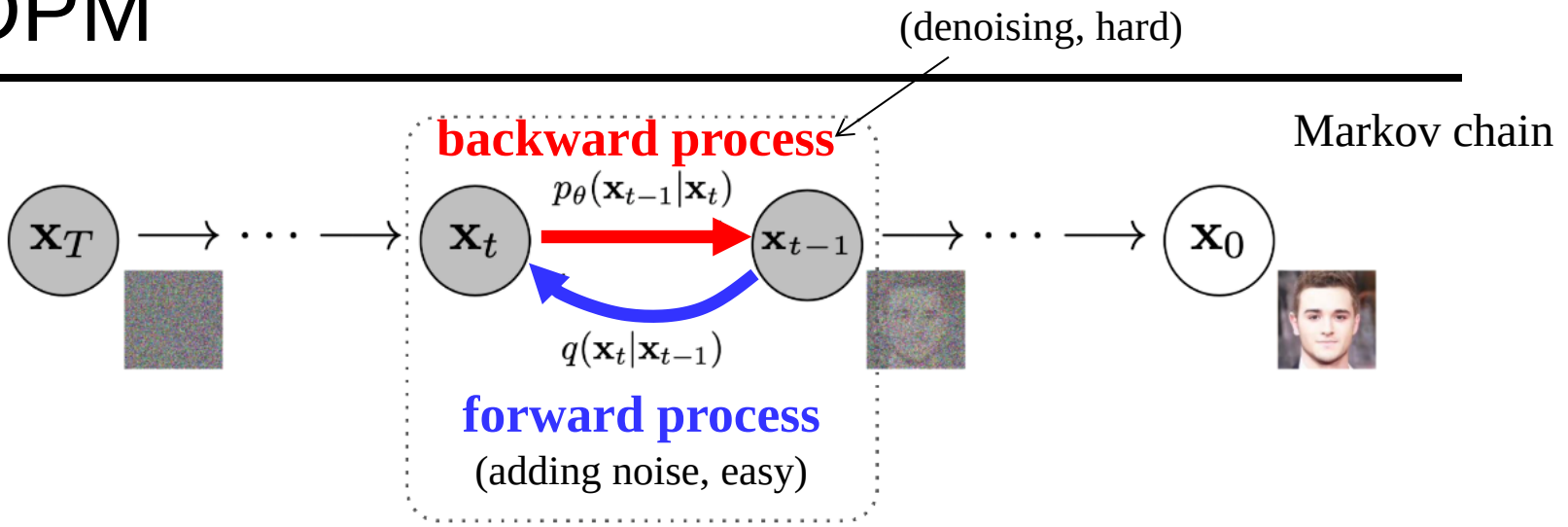
$$\mathbf{x}_{t-1} = \frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\ \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} + \sigma_t \mathbf{z}$$

using some white noise **z**

5: **end for**

6: **return** $\mathbf{x}_0$

DDPM



Backward process: $p(x_{t-1} \mid x_t)$ is not easy to estimate, so we learn it as a parametric model ~~$p_\theta(x_{t-1} \mid x_t)$~~ $\epsilon_\theta(\mathbf{x}_t, t)$ **denoising model**

Training from existing images :

1: **repeat**

2: select random training image $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

3: select random layer t

4: generate white noise ε and recursively "corrupted" images $\mathbf{x}_t = \sqrt{\alpha_t}\ \mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\ \epsilon_{t-1}$ (as on the previous slide)

5: learn to "denoise", take one gradient descent step on SSD

$$\min_\theta \left\| \left(\mathbf{x}_{t-1} - \frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\ \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}}\right) \right\|^2$$

$$\propto \|\epsilon_{t-1} - \epsilon_\theta(\mathbf{x}_t, t)\|^2$$

i.e. learn to predict the noise

6: **until** "converged"

Sampling new images :

1: generate white noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

2: **for** $t = T, \ldots, 1$ **do**

3: use trained models $\epsilon_\theta(\mathbf{x}, \mathbf{t})$ to recursively generate images

$$\mathbf{x}_{t-1} = \frac{\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\ \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} + \sigma_t \mathbf{z}$$

using some white noise **z**

"formally" derived

5: **end for**

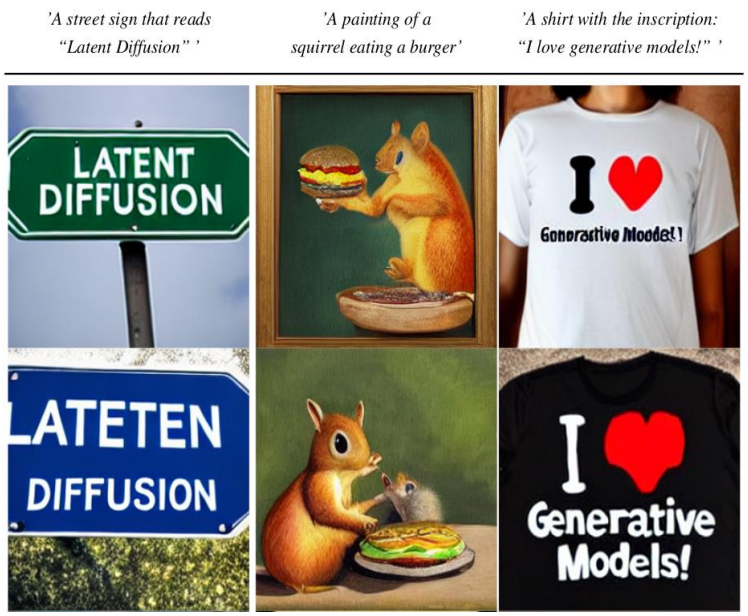
6: **return** $\mathbf{x}_0$

Applications of Diffusion Models

- Super-resolution

Left: 128x128 low-resolution image. Right: 512x512 resolution image

- Text-to-image Synthesize



use "conditional" generation

- Image inpainting

UNIVERSITY OF WATERLOO

Generative network models

implicitly learn probability density $P(I)$ for images
(enough to sample/generate images)

What about **generative classification**?

iClicker

Question: What best describes **generative model for classification**?

A: when a model generates class label
B: when a model generates image of a given class
C: Bayes posterior using densities for each class
D: log-likelihood ratio test

Challenges for image understanding

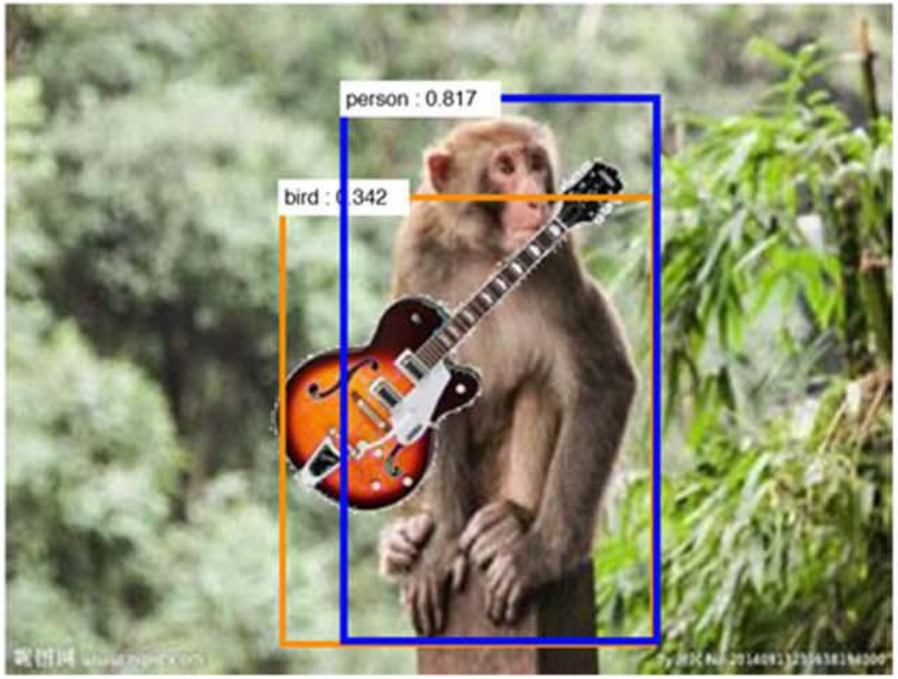
- Self-supervised *generative networks* do not directly apply to classification
- Shortage of Labels:
  - self-supervision can be used to improve encoder
  - domain adaptation, transfer knowledge
  - weakly-supervised or semi-supervised training
- General NN limitations:

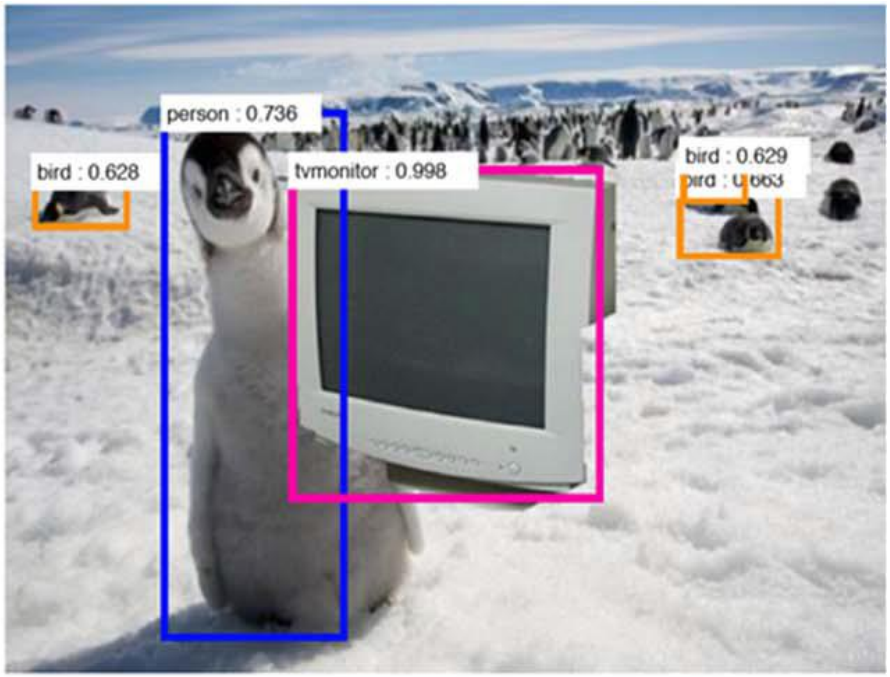
  - We started from *perceptron* motivated by *neuron*, but CNNs are clearly **not how brain works**. Why?
  - great pattern matching/classification (learned hierarchical non-linear filters) but **no real intelligence** (yet) – easy to fool, creativity?

Challenges for image understanding

Deep Nets: What have they ever done for Vision?”

Alan Yuille, 2019 https://neuralarchitects.org/slides/yuille-slides.pdf





- great pattern matching/classification (learned hierarchical non-linear filters)

but **no real intelligence** (yet) – easy to fool, creativity?

no training data is enough for **combinatorially complex world**

Computer Vision

is a great interdisciplinary research area

lots of open problems