Solving inverse problems with neural networks

CNNs and applications

EE367/CS448I: Computational Imaging

stanford.edu/class/ee367

Lecture 9

Gordon Wetzstein

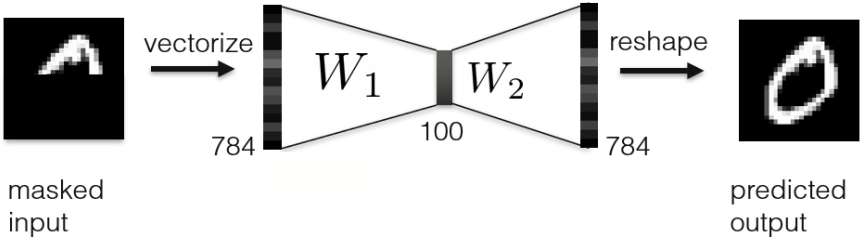
Stanford University

courtesy of David Lindell, slides adapted from Stanford CS231N

Last time...

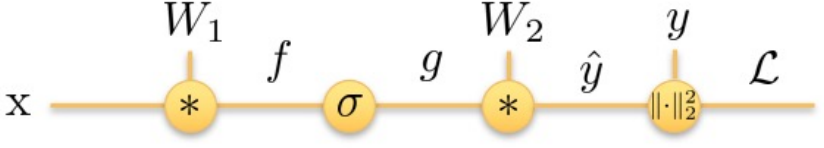
1. **Collect** training images and labels $\{x_i^{\text{tr}}\}, \{y_i^{\text{tr}}\}$
2. Define a **model** = parametric function with discretized outputs $f(x, \theta) = \hat{y}$
3. Define a **loss** = score function $\mathcal{L}(\{\hat{y}_i\}, \{y_i\})$
4. **Train** the model (i.e., optimize the weights) $\min._\theta \mathcal{L}(\{f(x_i^{\text{tr}}, \theta)\}, \{y_i^{\text{tr}}\})$
5. **Evaluate** the model on unseen images $\mathcal{L}(\{f(x_i^{\text{test}}, \theta^*)\}, \{y_i^{\text{test}}\})$

Last time...



Fully-connected networks (MLPs)

(Step 2)



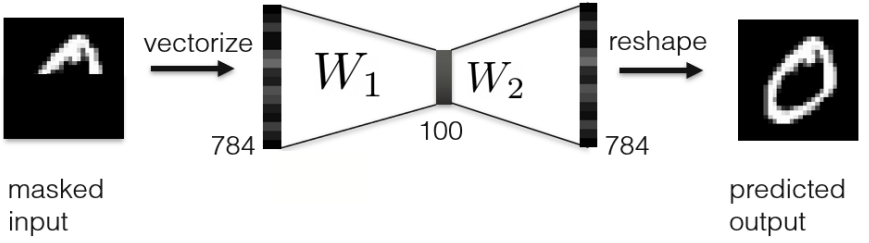
Computational graphs & backprop

(Step 4)

Today

- CNNs
- Building blocks of CNNs
- Deep networks: ResNets & U-Nets
- Deep learning applied to inverse problems

Drawbacks of fully-connected networks



Spatial structure is destroyed

Fully-connected weights do not scale

Convolutional Neural Networks

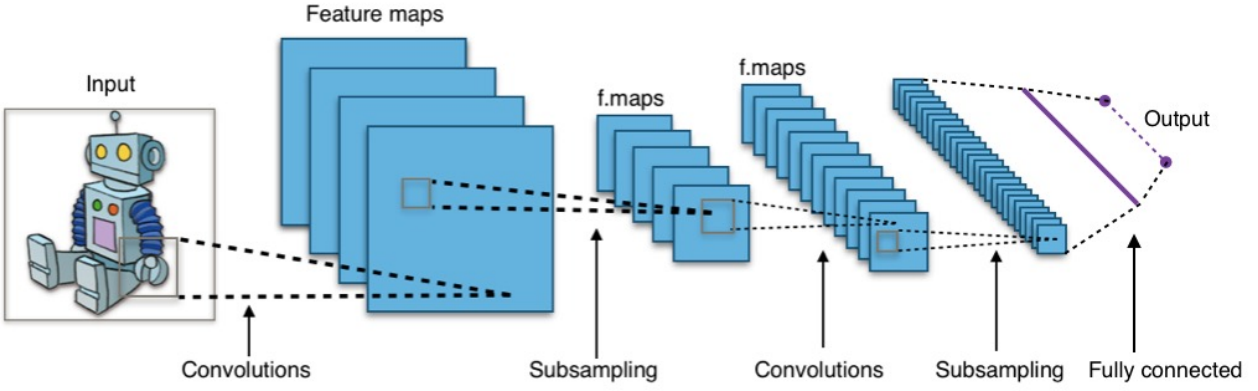


Image: CC Aphex34

Exploit spatial structure

Scale to large inputs with fewer parameters

ImageNet and AlexNet

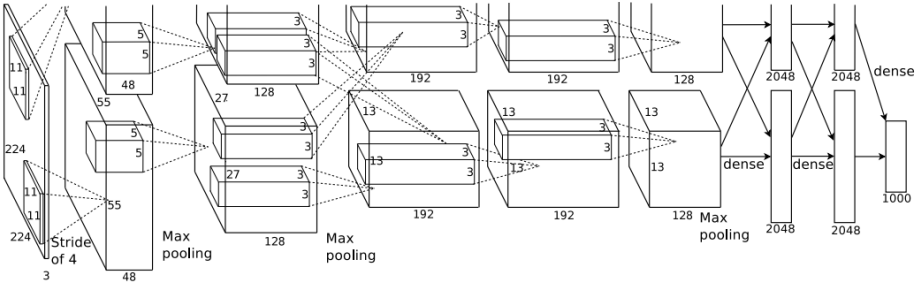
2010: ImageNet Large Scale Visual Recognition Challenge

10^7 labeled images

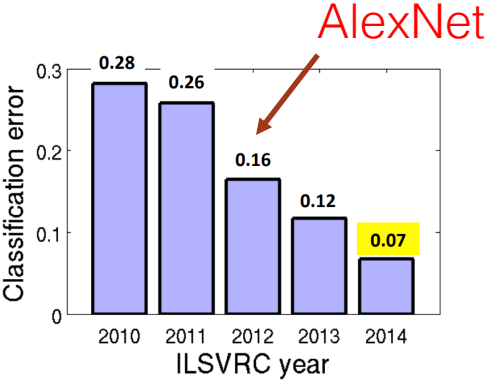
ImageNet and AlexNet

2010: ImageNet Large Scale Visual Recognition Challenge

2012: First CNN applied to ImageNet (AlexNet)



AlexNet [Krizhevsky ‘12]

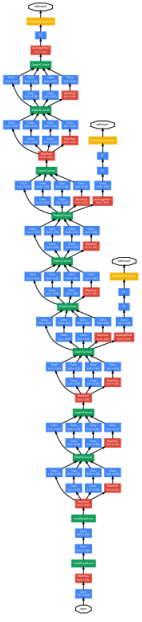


[Russakovsky ’15]

Deep Networks

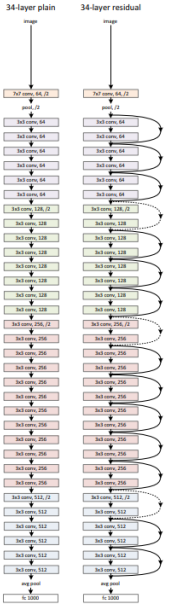
VGG

[Simonyan ‘14]



GoogLeNet

[Szegedy ‘14]



ResNet

[He ‘15]

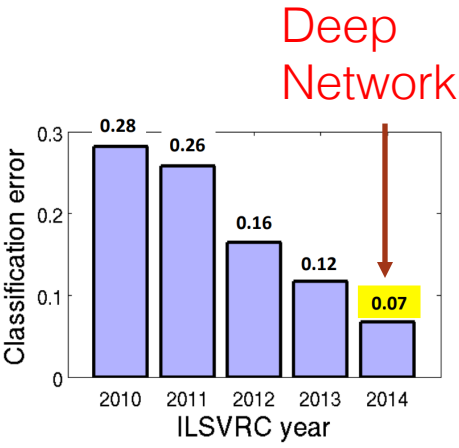
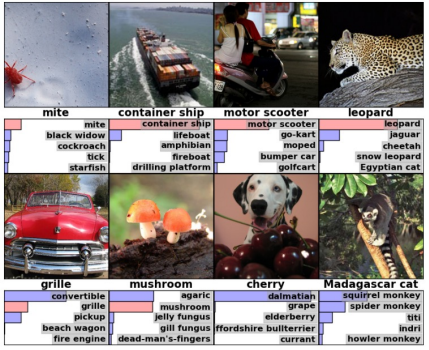


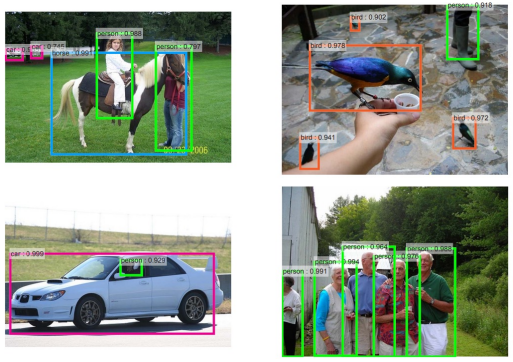
Image Classification

[Krizhevsky '12]



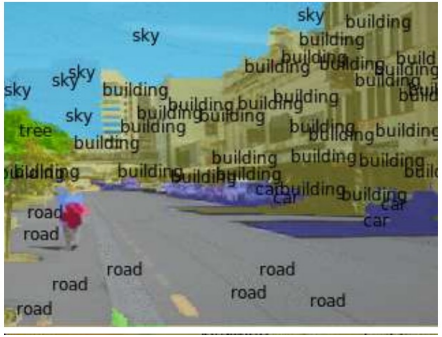
Object Detection

[Ren '16]



Segmentation

[Farabet '13]



Pose estimation

[Toshev '14]

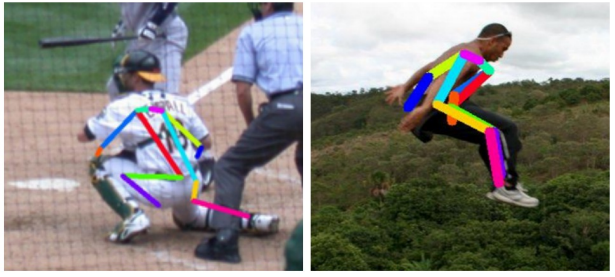


Image Denoising

[Zhang '17]

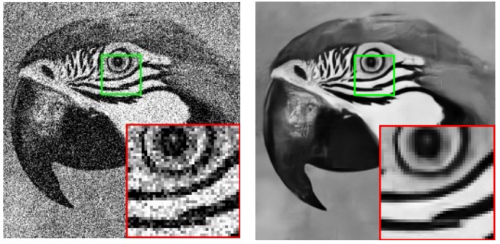


Image Deblurring

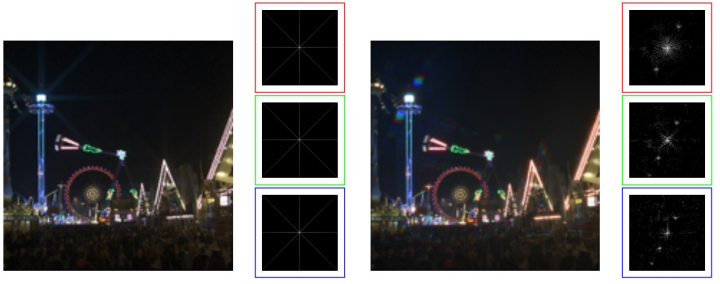
[Nah '17]

Learned ISPs

[Chen '18]

End-to-End Optimization

[Metzler '19]



Monocular Depth Estimation

[Eigen '14]

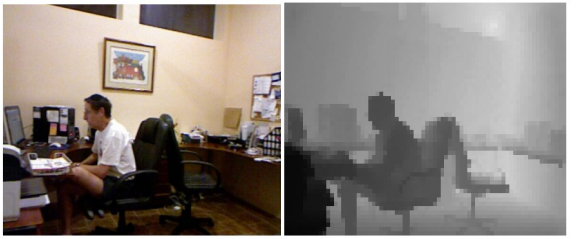
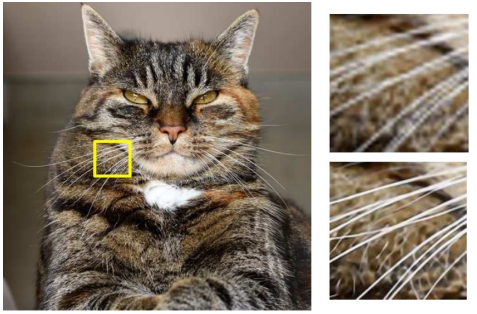


Image Relighting

[Sun '19]

Image Super-resolution

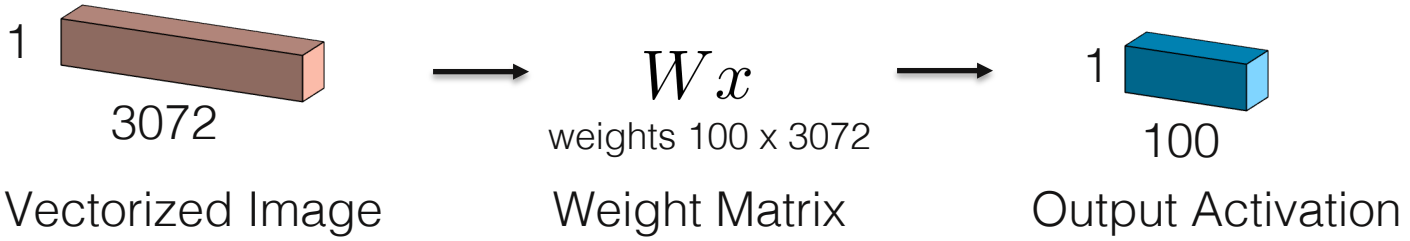
[Lim '17]



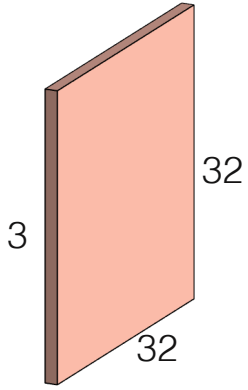
Synthetic Depth-of-Field

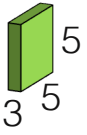
ai.googleblog.com

Fully-Connected Layer



Convolutional Layer



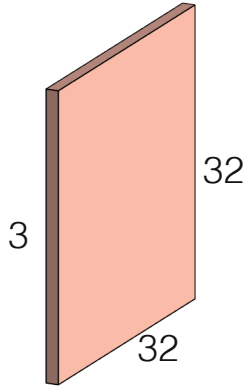


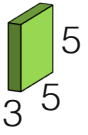
weights 5x5x3

Input Image

Filter

Convolutional Layer





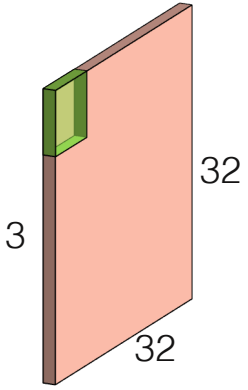
weights 5x5x3

"Channel" dimension

Input Image

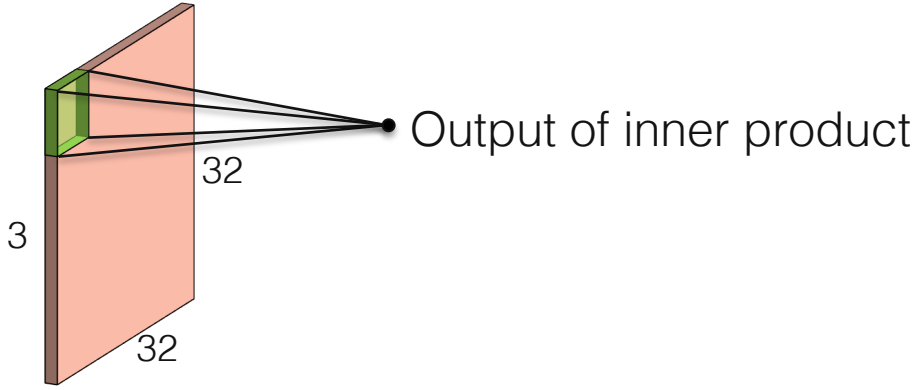
Filter

Convolutional Layer



Input Image

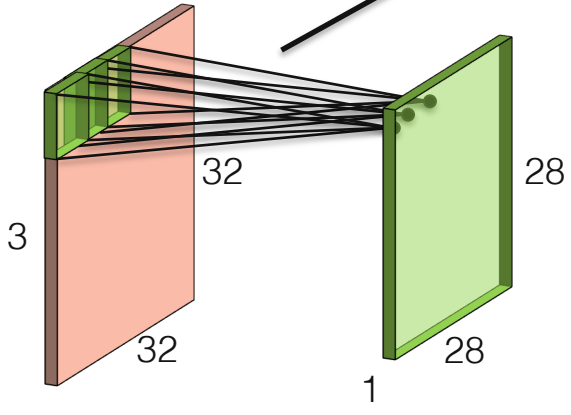
Convolutional Layer



Input Image

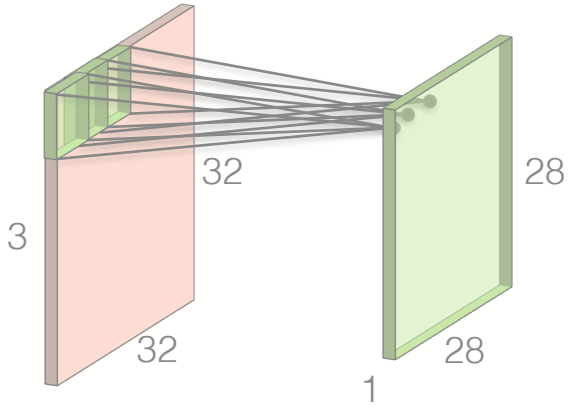
Convolutional Layer

Convolution = sliding window + inner product

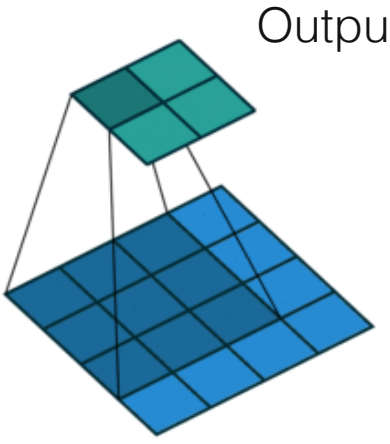


Input Image Activations

Convolutional Layer

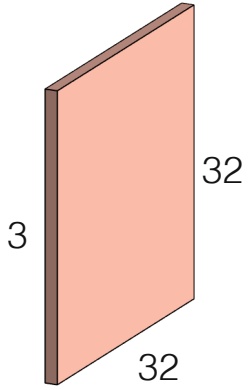


Input Image Activations

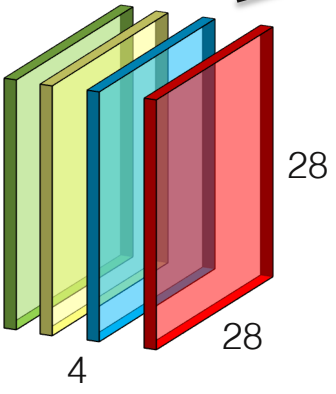


https://github.com/vdumoulin/conv_arithmetic

Convolutional Layer



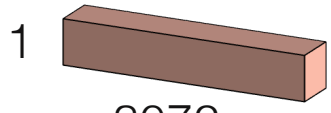
Input Image



Activations

Multiple output channels
using multiple filters

Fully-Connected Layer

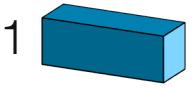


Vectorized Image

Wx

weights 100 x 3072

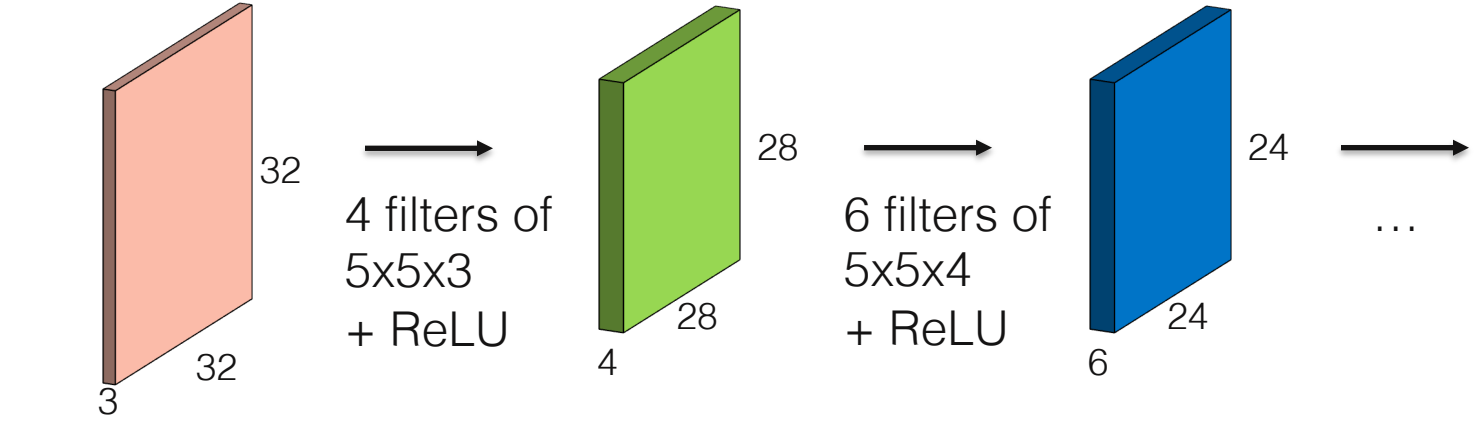
Weight Matrix



Output Activation

Special case of convolutional layer when filter size = input size

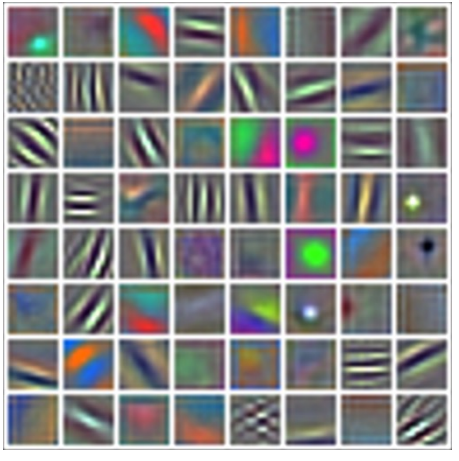
Convolutional Neural Network



Case Study: AlexNet

Input Image

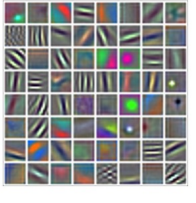
First-layer Filters

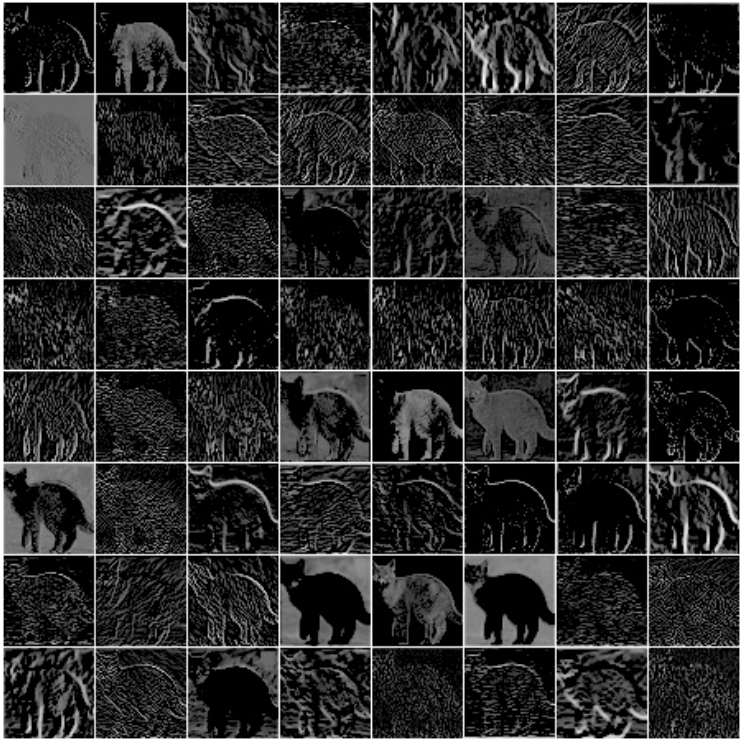


Case Study: AlexNet

Input Image

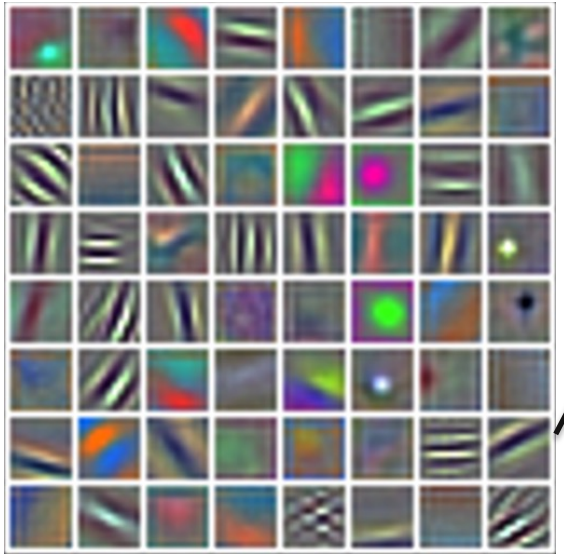
First-layer Filters





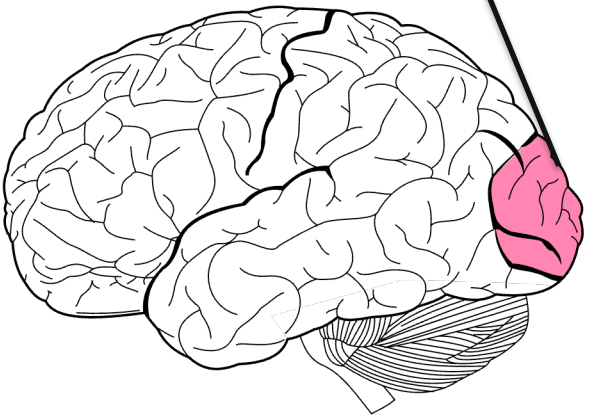
Activations

Case Study: AlexNet



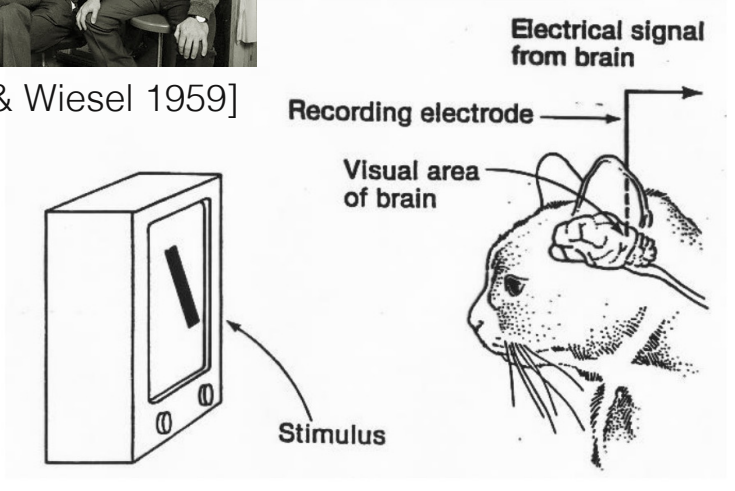
First-layer Filters

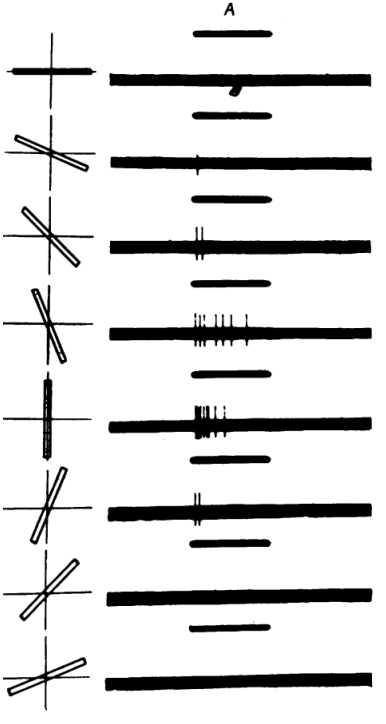
Similar to simple cells
in visual cortex!
- Edge detectors



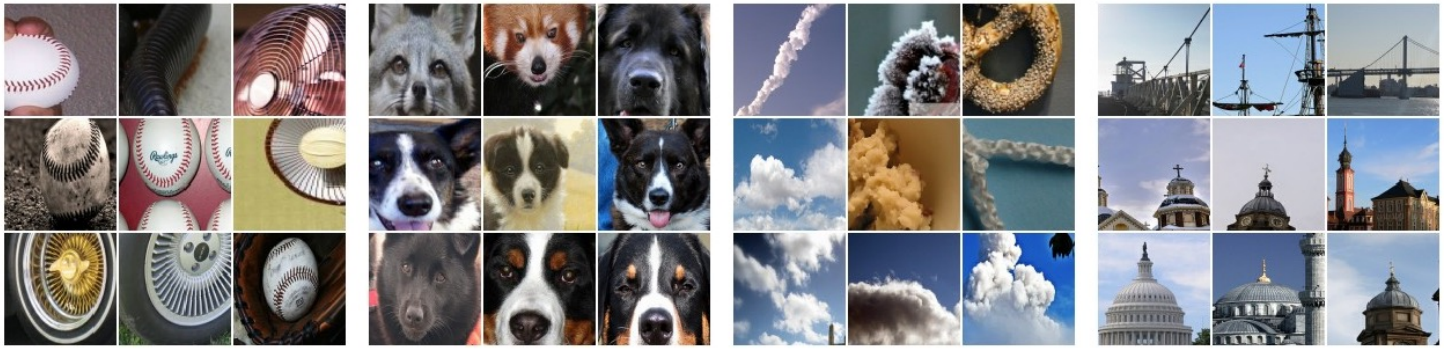
Case Study: AlexNet

[Hubel & Wiesel 1959]





CNN higher layer filters



[Olah ‘17]

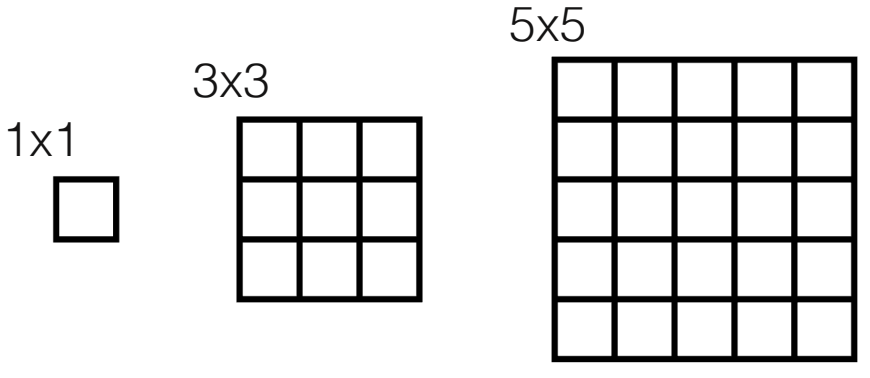
Dataset examples that maximize neuron outputs

Convolutional Layers – Design Choices

- Filter size
- Number of filters
- Padding
- Stride

Convolutional Layers – Design Choices

Filter size



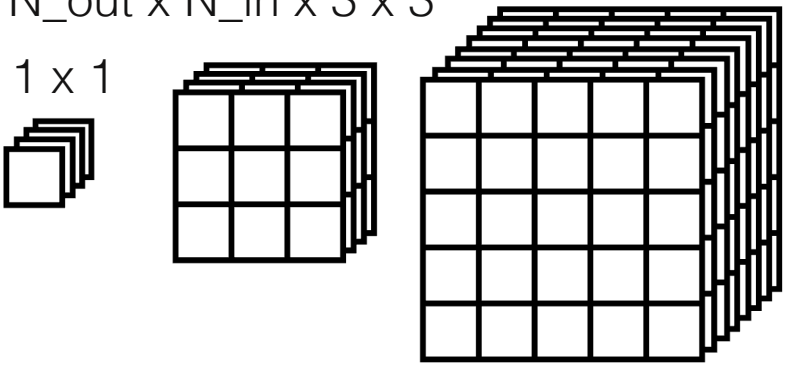
Convolutional Layers – Design Choices

Number of channels

N_out x N_in x 5 x 5

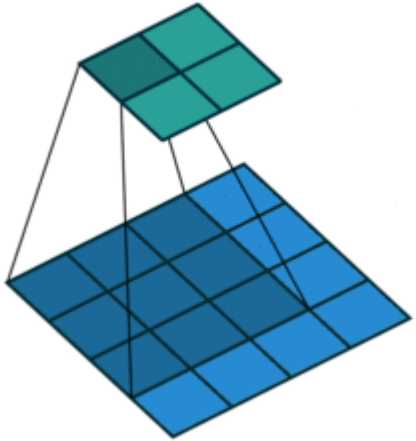
N_out x N_in x 3 x 3

N_out x N_in x 1 x 1

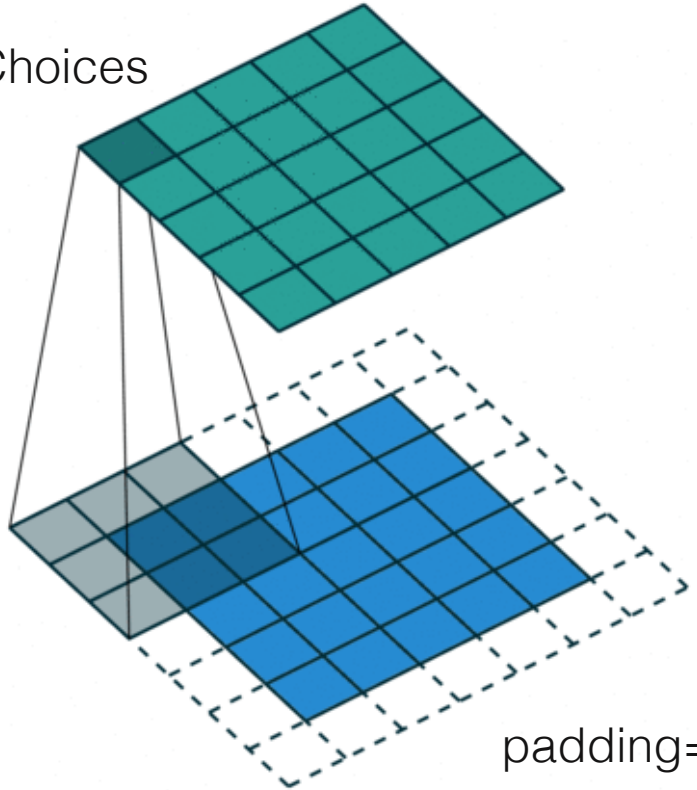


Convolutional Layers – Design Choices

padding



No padding

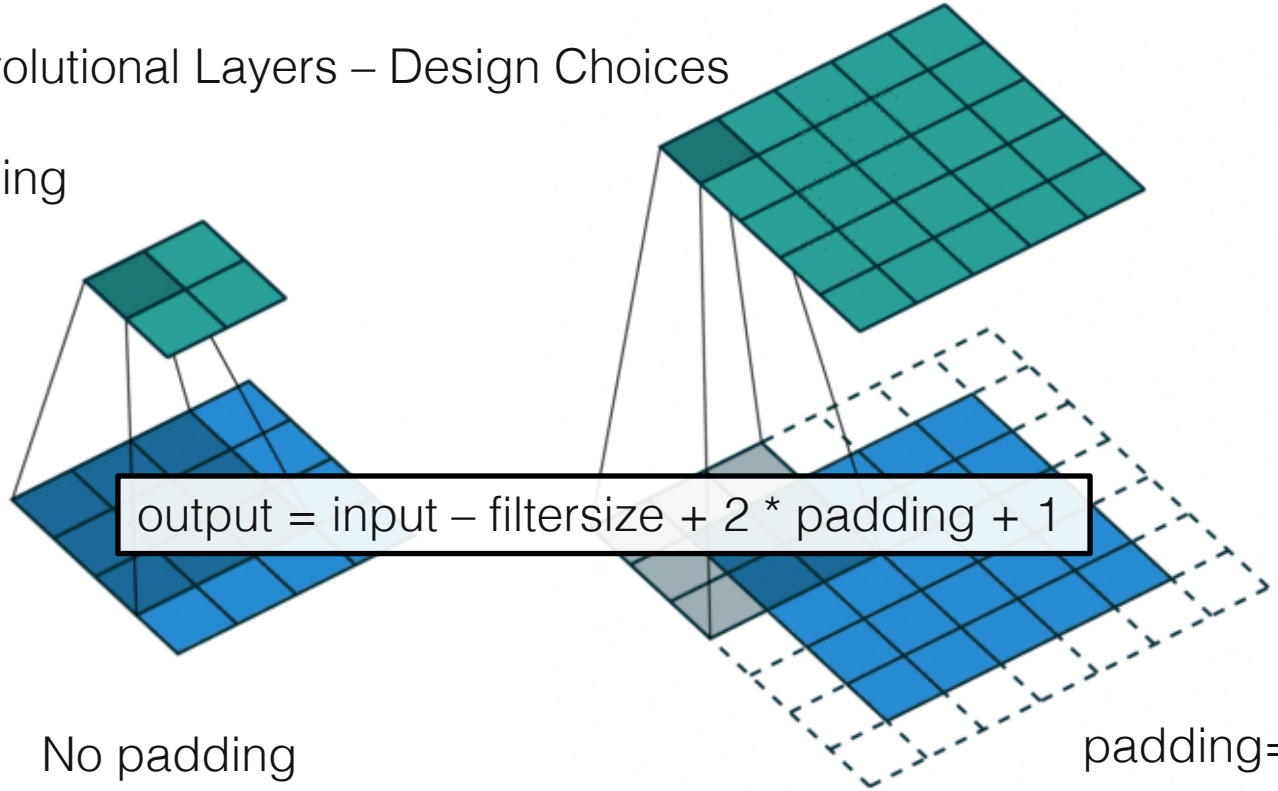


padding=1

https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers – Design Choices

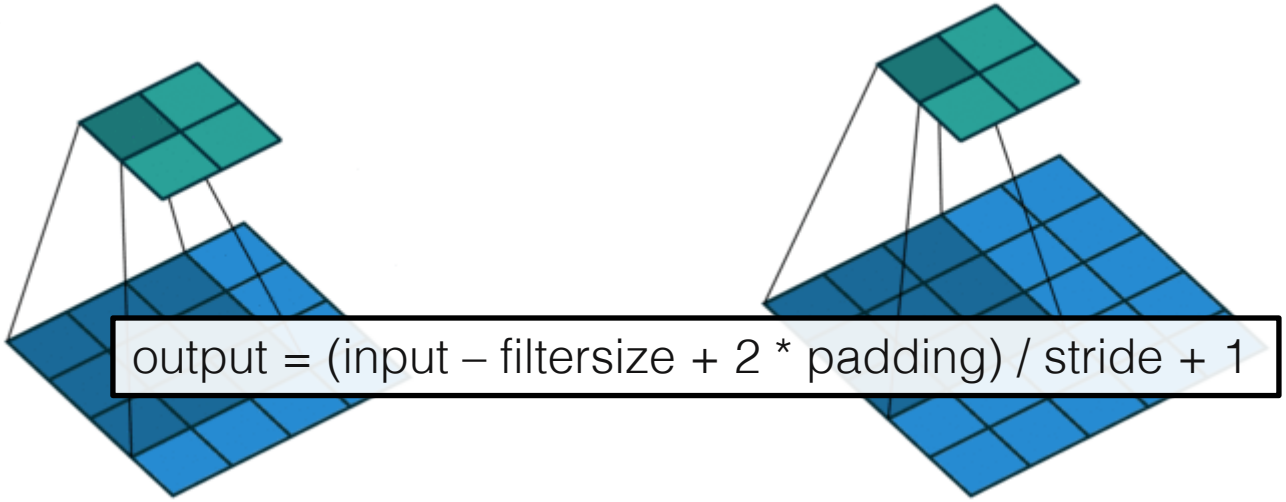
padding



https://github.com/vdumoulin/conv_arithmetic

Convolutional Layers – Design Choices

stride



stride = 1

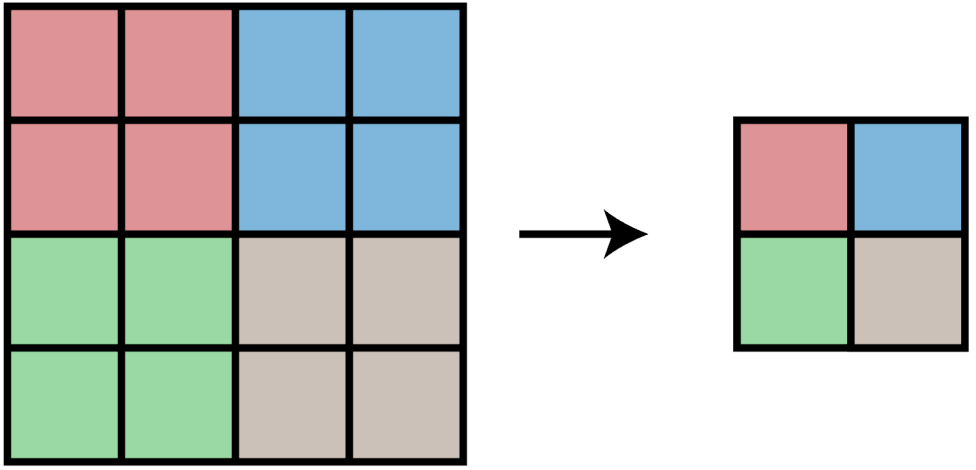
stride = 2

https://github.com/vdumoulin/conv_arithmetic

CNNs – Other Layers

- Pooling layers
- Transposed convolution
- Upsampling layers

Pooling

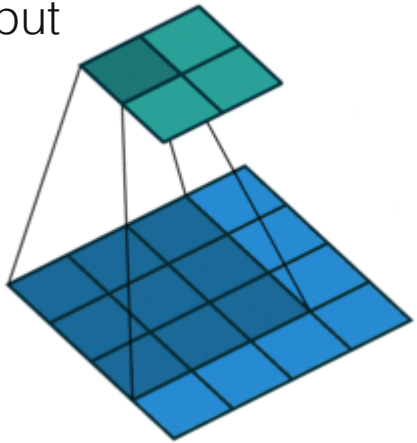


e.g., max pool size=2, stride=2

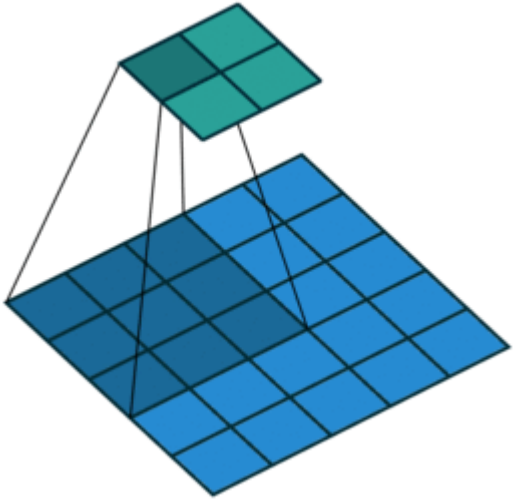
Transposed Convolution

input

output



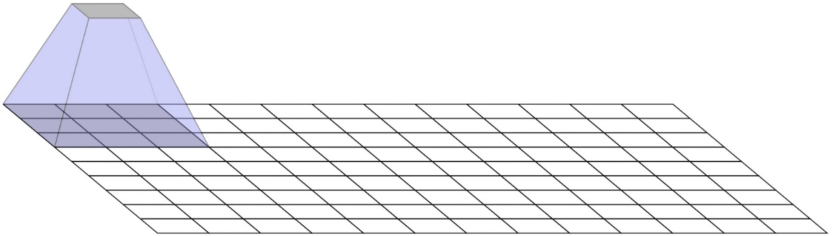
stride=1



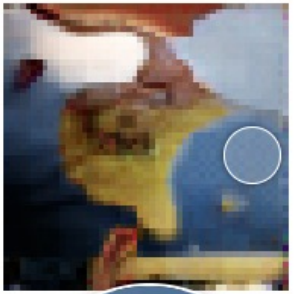
stride=2

https://github.com/vdumoulin/conv_arithmetic

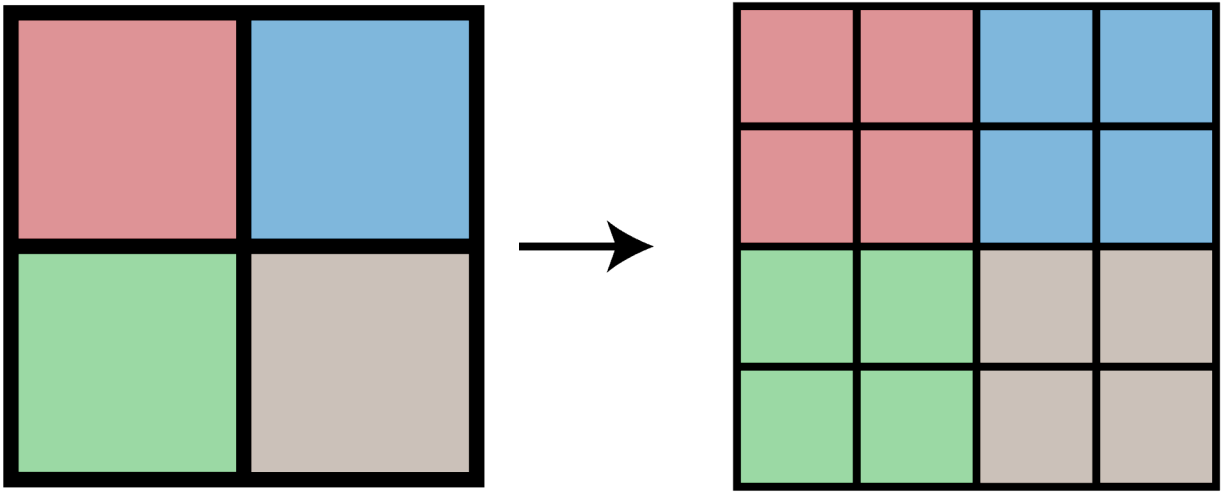
Transpose Convolution (checkerboard artifacts)



[Odena ‘16]

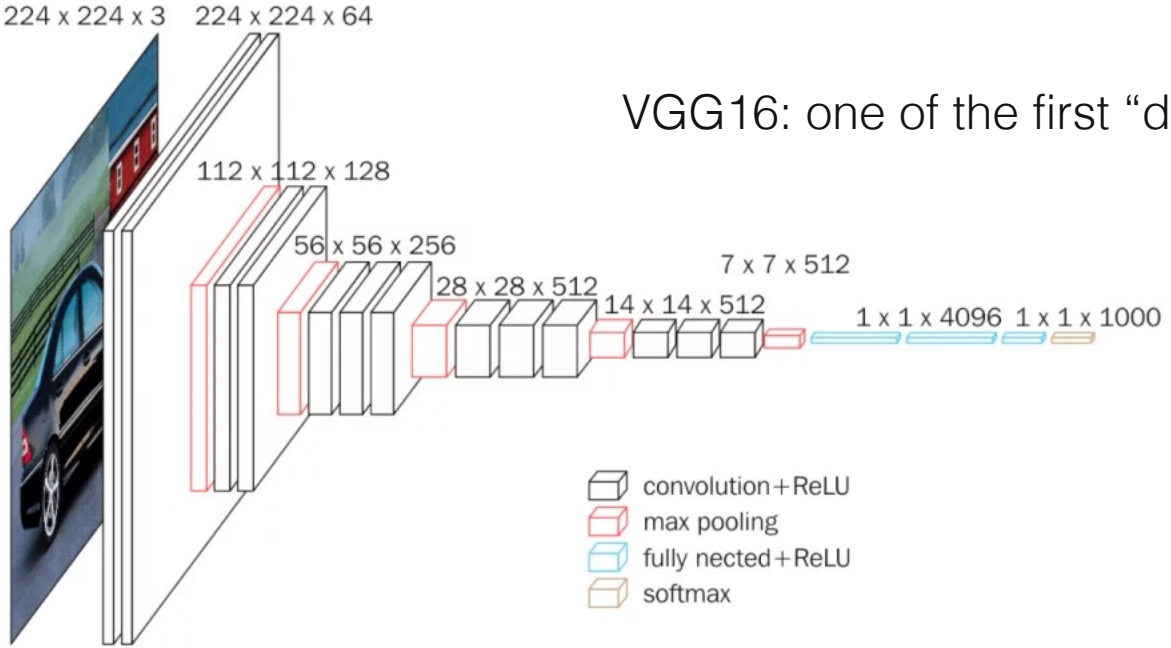


Upsampling layers



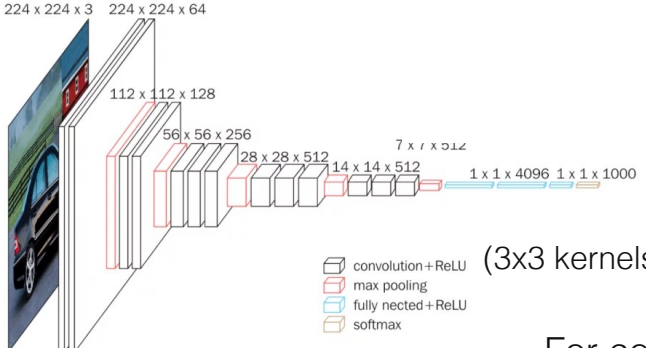
e.g., bilinear upsampling, nearest neighbor upsampling

Common Network Architectures



VGG16: one of the first “deep” CNNs

Common Network Architectures



(3x3 kernels)

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 64)      1792
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 224, 224, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 64)      0
 ...
=================================================================
Total params: 134,268,738
Trainable params: 134,268,738
Non-trainable params: 0
```

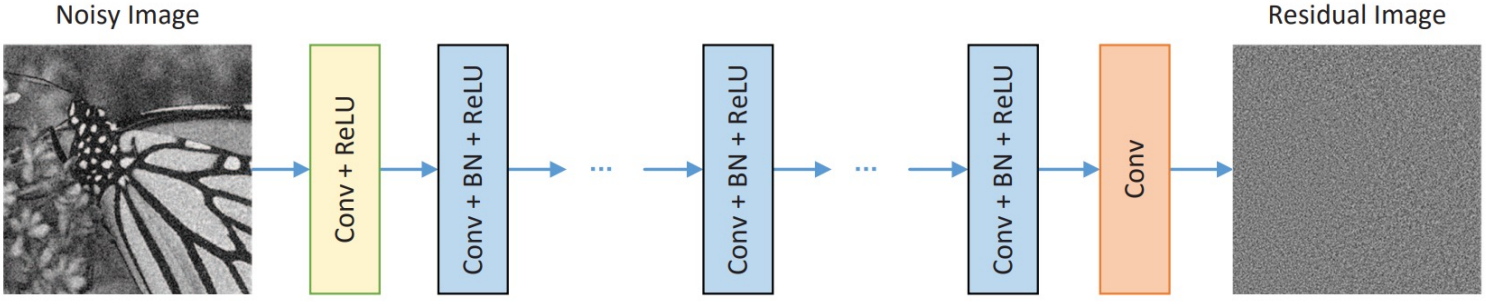
Not suitable for image processing…

For convolutional layers, the number of parameter is:

$$\underbrace{C_{\text{out}} \times C_{\text{in}} \times k \times k}_{\text{filters}} + \underbrace{C_{\text{out}}}_{\text{bias}}$$

Screenshot: Rohit Thakur – from builtin.com

Image denoising with DnCNN



[Zhang ‘16]

Key ideas: residual learning & batch normalization

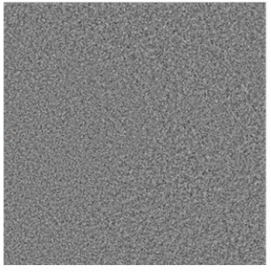
Residual Learning

Clean image

=

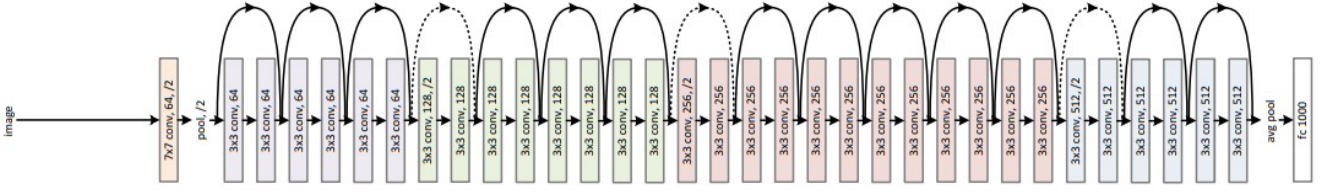
noisy image

-



estimated noise

Residual Learning



[He ‘15]

Popularized by residual nets “ResNets” for image classification

Batch Normalization

- Normalize the data along the “batch” dimension
- Can speed up and stabilize training

BATCHNORM2D

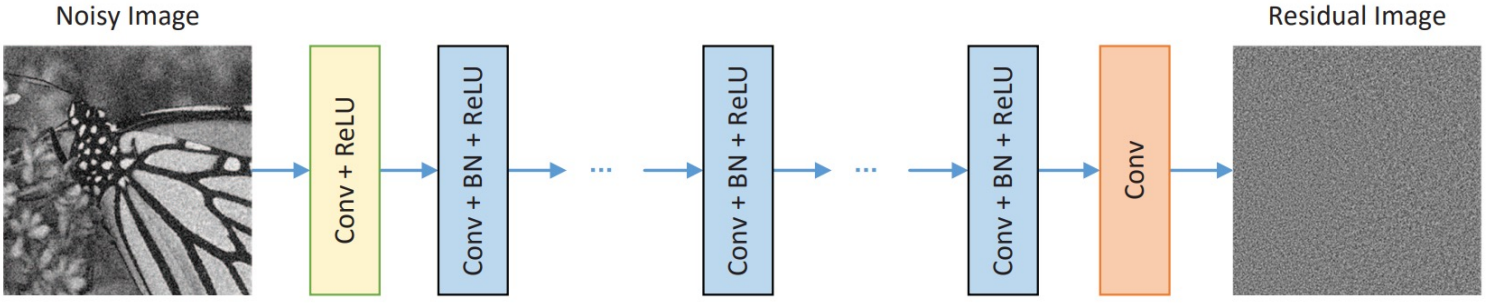
CLASS `torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True, device=None, dtype=None)` [SOURCE]

Applies Batch Normalization over a 4D input (a mini-batch of 2D inputs with additional channel dimension) as described in the paper Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift .

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

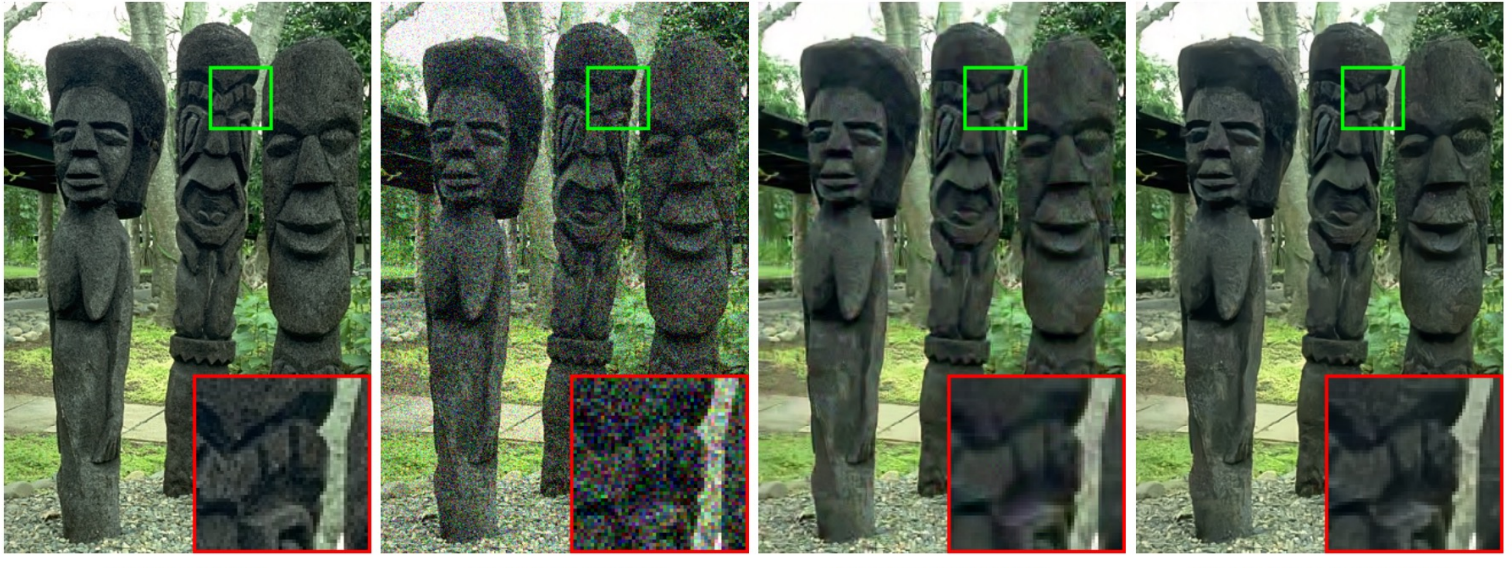
https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html

Image denoising with DnCNN



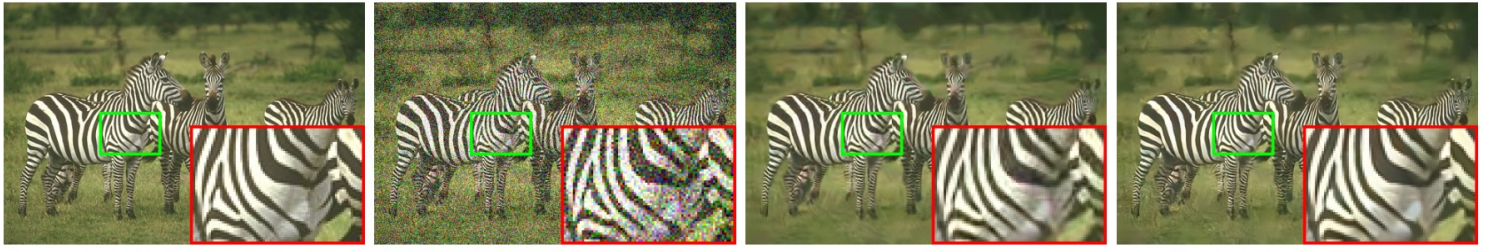
[Zhang ‘16]

No fully connected layers – can be applied to any input size



(a) Ground-truth (b) Noisy / 17.25dB (c) CBM3D / 25.93dB (d) CDnCNN-B / 26.58dB

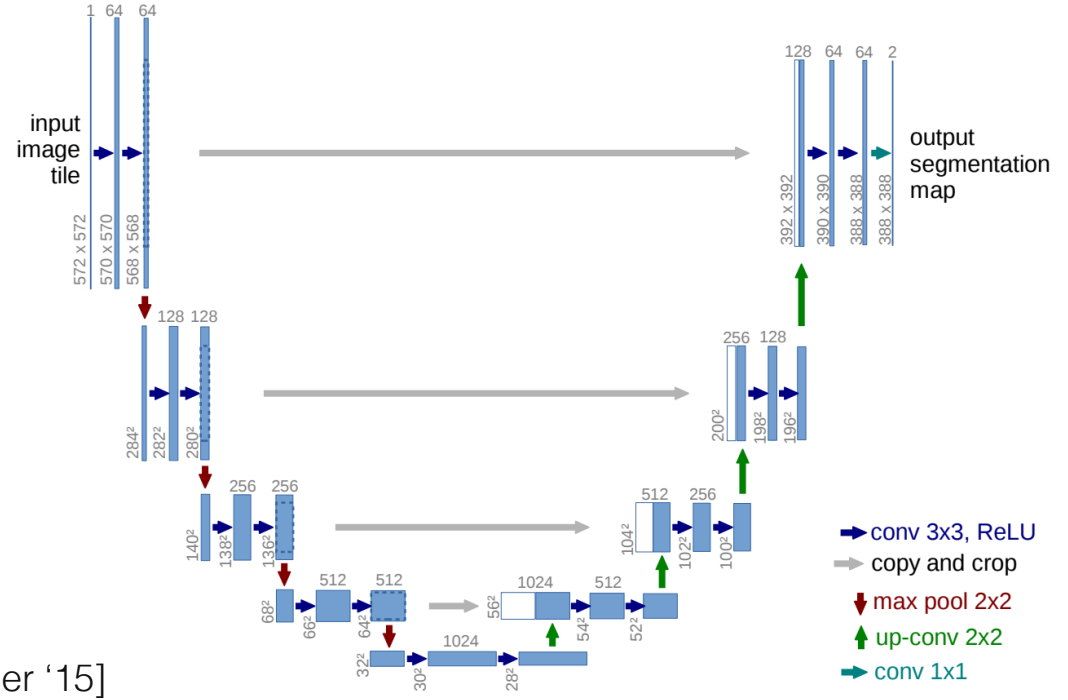
[Zhang ‘16]



(a) Ground-truth (b) Noisy / 15.07dB (c) CBM3D / 26.97dB (d) CDnCNN-B / 27.87dB

[Zhang ‘16]

U-Net: General-purpose architecture

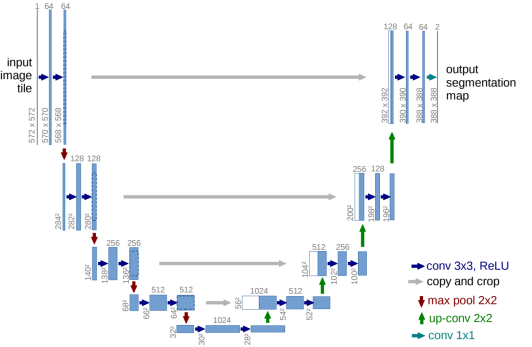


[Ronneberger ‘15]

U-Net: General-purpose architecture

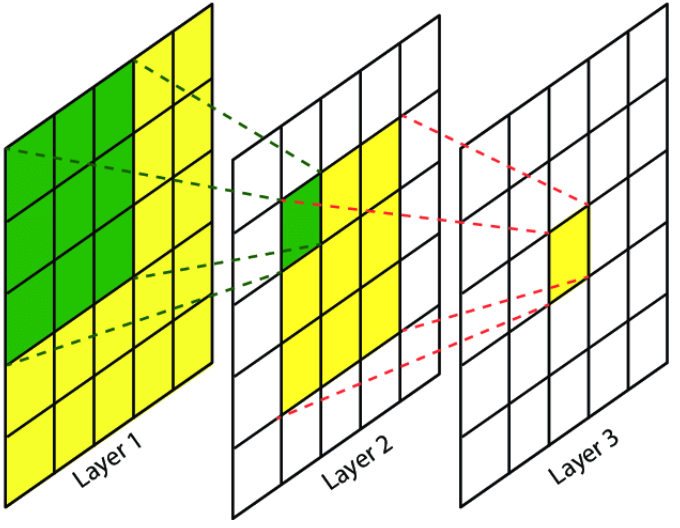
Introduced for biomedical image segmentation

- Uses residual connections
- Multi-scale processing (captures details at different scales)
- Large receptive field



U-Net: General-purpose architecture

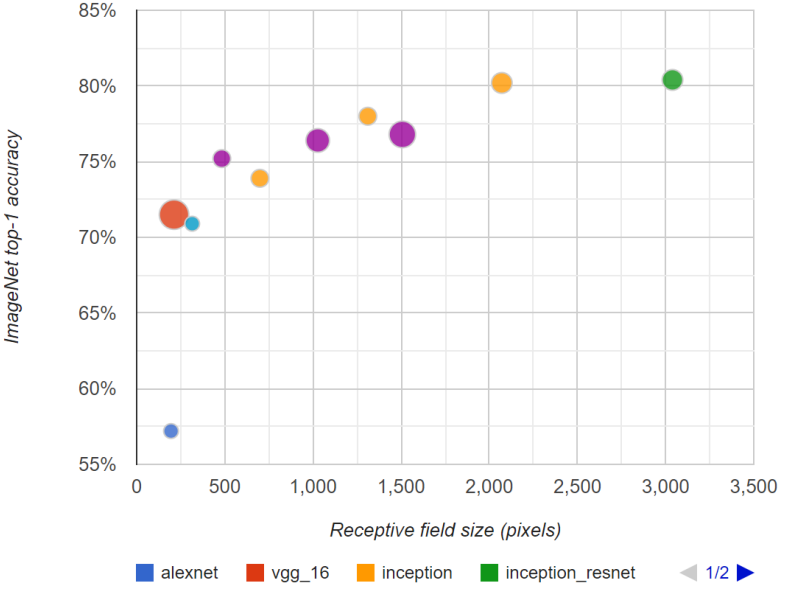
Receptive field: size of the input that contributes to the activation/output value



[Lin ‘17]

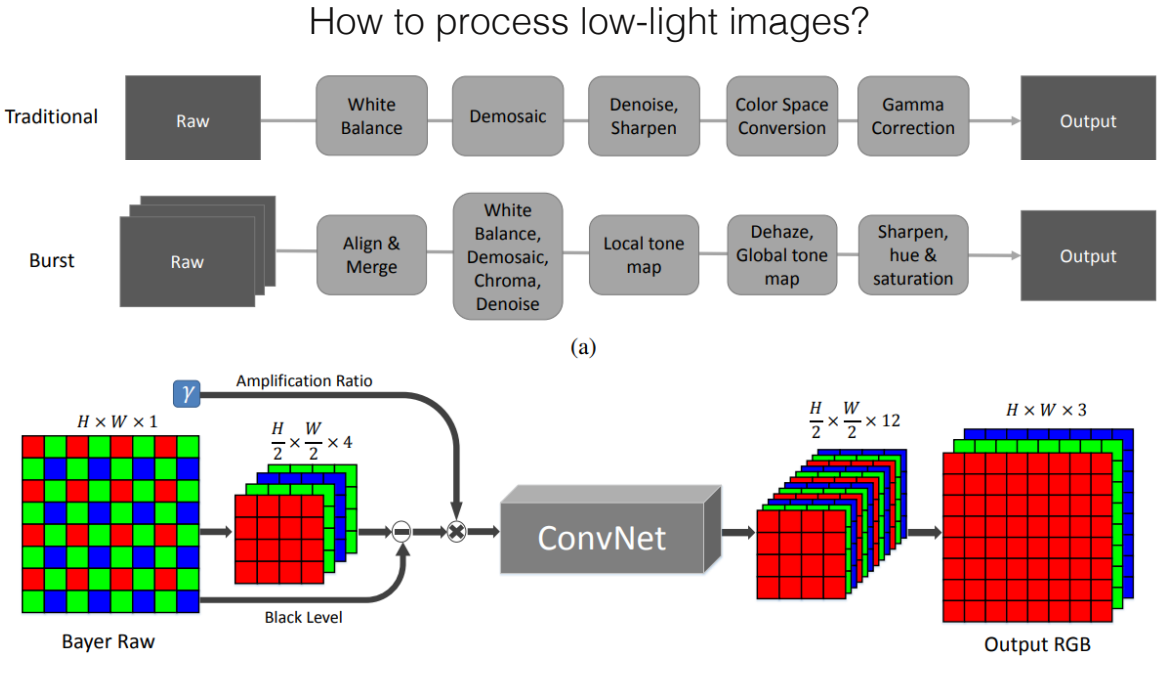
U-Net: General-purpose architecture

Large receptive field is important for high-level vision tasks and semantic understanding



[Araujo ‘19]

“Learning to see in the dark”



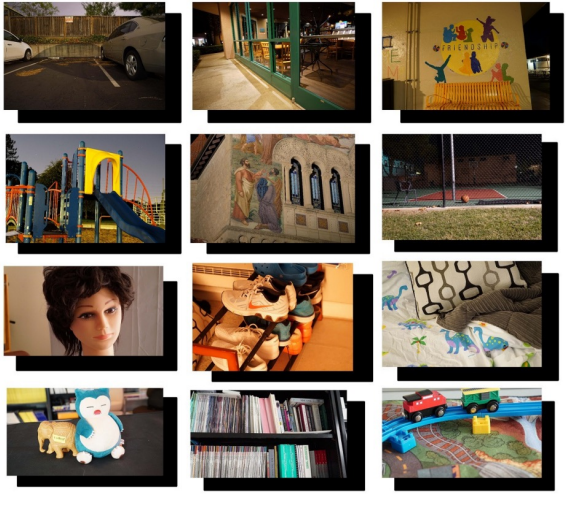
[Chen ‘18]

“Learning to see in the dark”

fully-connected layers that require processing small image patches and reassembling them [26]. Our default architecture is the U-net [35].

[Chen ‘18]

“Learning to see in the dark”



Trained on short-exposure (noisy) / long-exposure image pairs

[Chen ‘18]

“Learning to see in the dark”

(a) Traditional pipeline (b) Our result

[Chen ‘18]

Deep optics for HDR imaging

Reference HDR

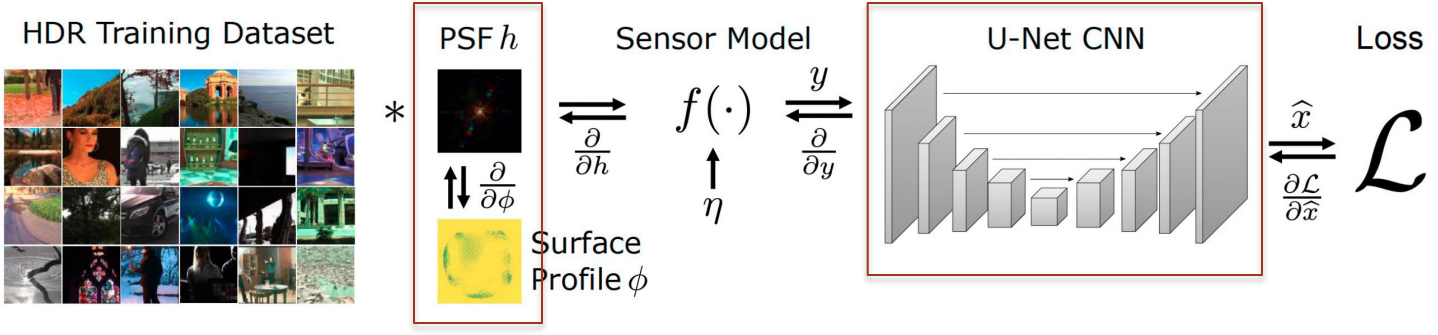
LDR

some information is lost by the sensor...

Can we optimally ”encode” the image with a tailored PSF?

[Metzler ‘20]

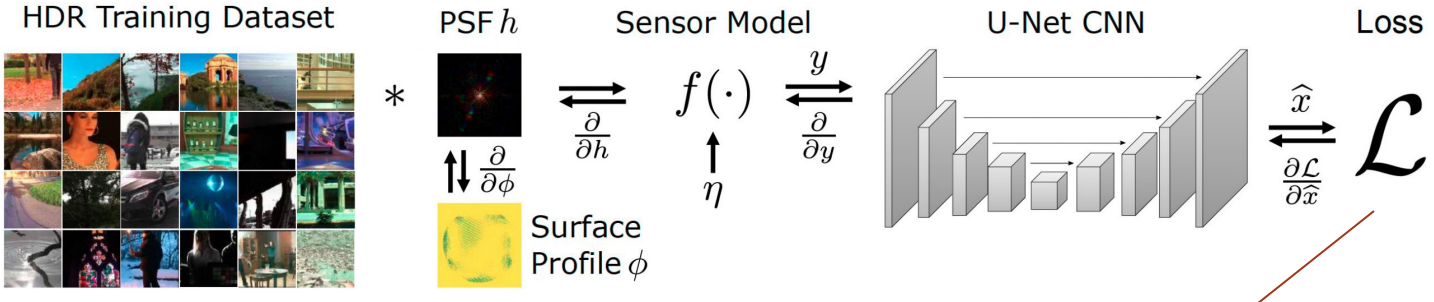
Deep optics for HDR imaging



The “encoder” (PSF) and the “decoder” (U-Net) are **jointly** optimized!

[Metzler ‘20]

Deep optics for HDR imaging

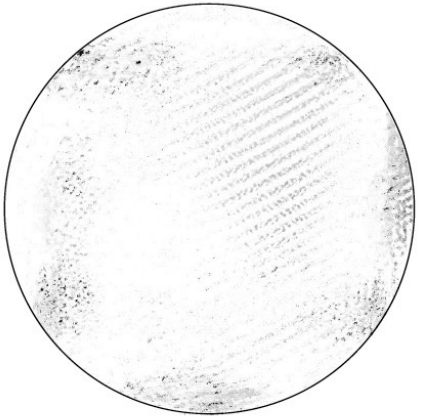


Training:
minimize the difference between reconstruction and tone-mapped GT images

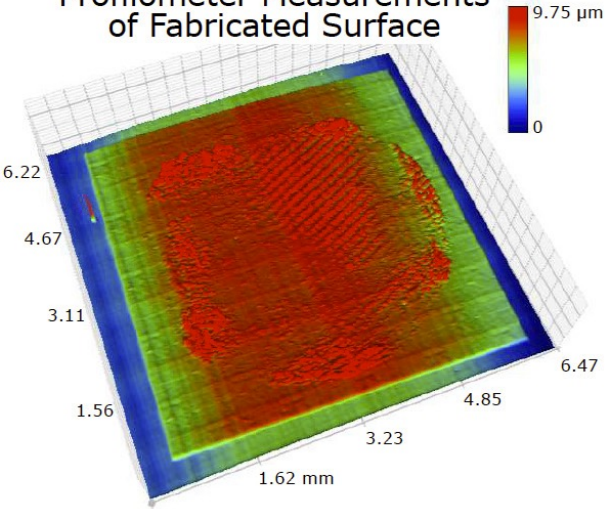
[Metzler ‘20]

Deep optics for HDR imaging

Optimized Height Profile

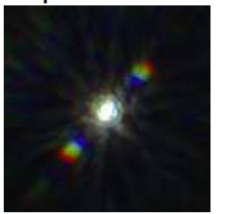


Profilometer Measurements of Fabricated Surface

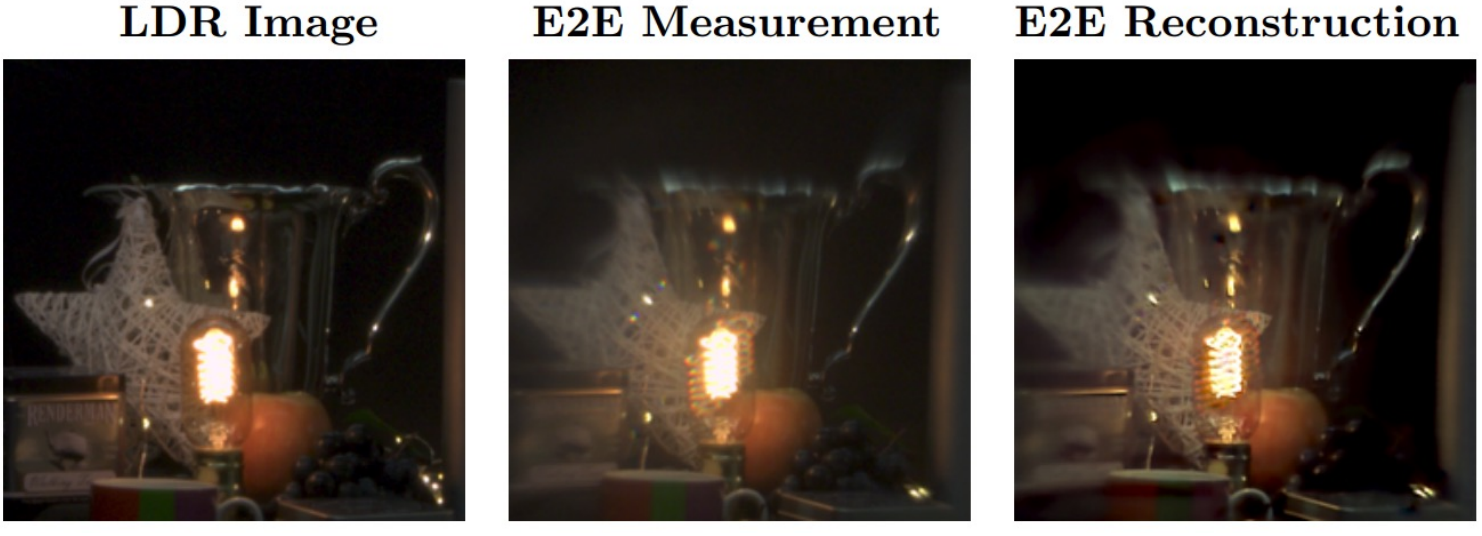


Simulated PSF

Captured PSF

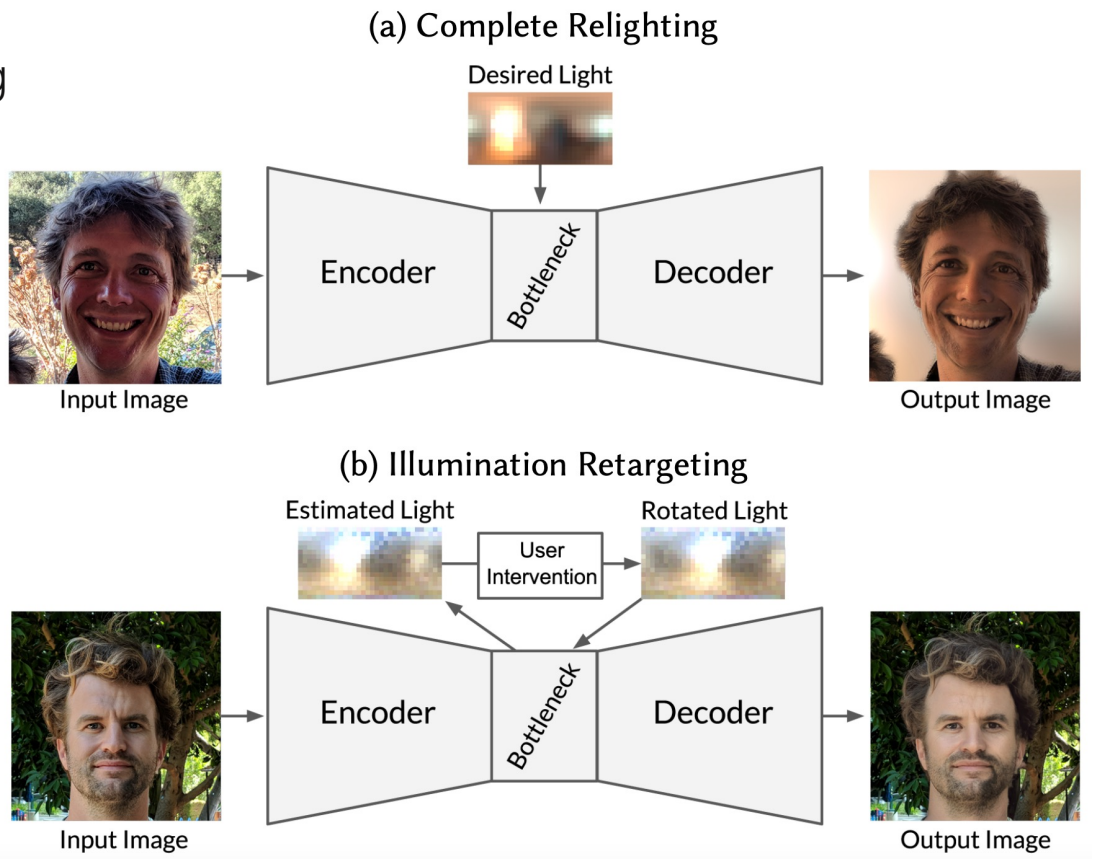


[Metzler ‘20]



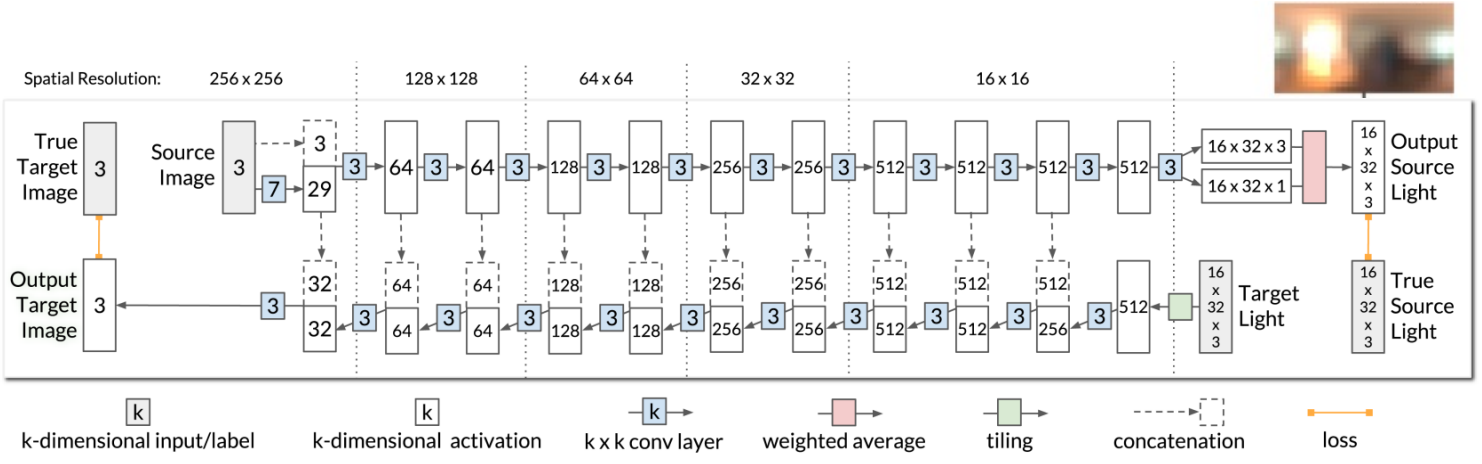
[Metzler ‘20]

Image Relighting



[Sun ‘19]

Image Relighting



How did they get the training data?

[Sun ‘19]

Image Relighting

Light-stage dataset capture (Google)

[Sun ‘19]

Image Relighting

OLAT photos (columns)

$$b = Ax$$

Re-rendered image

Environment map

(a) OLAT images (7 cameras).

(b) Ground-truth renderings.

[Sun ‘19]

Image Relighting

(a) Input image and estimated lighting

(b) Rendered images from our method under three novel illuminations

[Sun ‘19]

Data-Driven Approach

1. **Collect** training images and labels $\{x_i^{\text{tr}}\}, \{y_i^{\text{tr}}\}$

2. Define a **model** = parametric function with discretized outputs $f(x, \theta) = \hat{y}$

What if we do not have access to clean data?

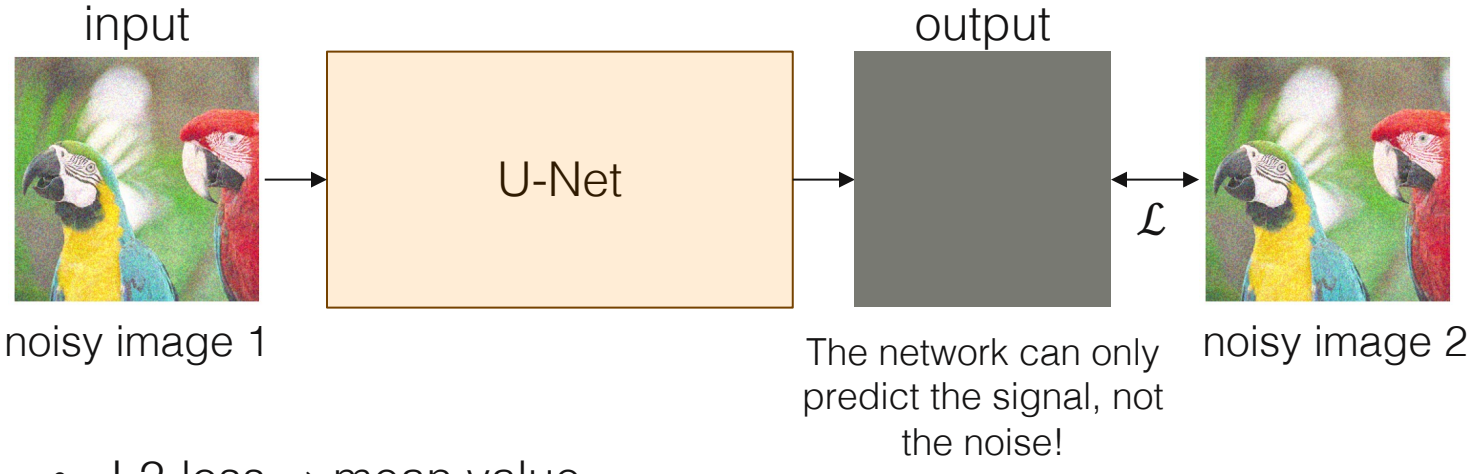
3. Define a **loss** = score function $\mathcal{L}(\{\hat{y}_i\}, \{y_i\})$

(e.g. for denoising)

4. **Train** the model (i.e., optimize the weights) $\min_\theta \mathcal{L}(\{f(x_i^{\text{tr}}, \theta)\}, \{y_i^{\text{tr}}\})$

5. **Evaluate** the model on unseen images $\mathcal{L}(\{f(x_i^{\text{test}}, \theta^*)\}, \{y_i^{\text{test}}\})$

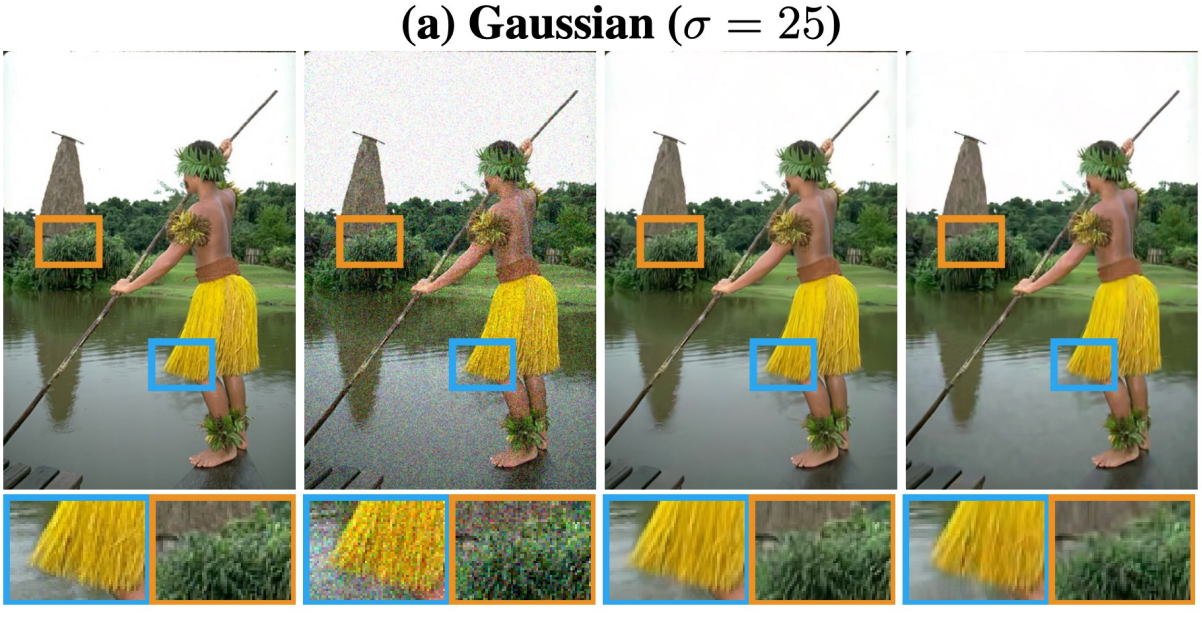
Noise2Noise



- L2-loss → mean value
- L1-loss → median value

[Lehtinen ’18]

Noise2Noise



[Lehtinen ’18]

Noise2Noise

Input ($p \approx 0.25$)	L_2	L_1
17.12 dB	26.89 dB	35.75 dB

[Lehtinen ’18]

What if we only have one corrupted image?

Maximum-A-Posteriori (MAP) Problems

I have:

- a measurement (observation): y
- a likelihood model: $\mathbf{Likelihood}(y|x)$
- a prior model: $\mathbf{Prior}(x)$

} I want to find the best model x^*:

$$
\begin{aligned}
x^\star &= \text{argmax}_x \; p(x|y) \\
&= \text{argmax}_x \; p(y|x)p(x) \\
&= \text{argmin}_x \; -\log p(y|x) - \log p(x) \\
&\qquad\qquad\qquad \text{data term} \qquad \text{regularizer}
\end{aligned}
$$

Denoising is a MAP Problem

$$x_0 = x + \text{noise}$$

Gaussian

$$x^\star = \text{argmin}_x \; \|x - x_0\|^2 - \alpha \log p(x)$$

You will learn how to solve this optimization problem efficiently in the next lecture...

We need an explicit prior

We only have one image...

Deep Image Prior

Neural networks convey an **implicit prior**!

$$x = f_\theta(z)$$

image

neural network

random input (noise)

$$\theta^* = \text{argmin}_\theta \{\|f_\theta(z) - x_0\|^2\} \qquad x^* = f_{\theta^*}(z)$$

The denoised imaged is implicitly represented by the weights of a NN

Deep Image Prior

Neural networks convey an **implicit prior**!

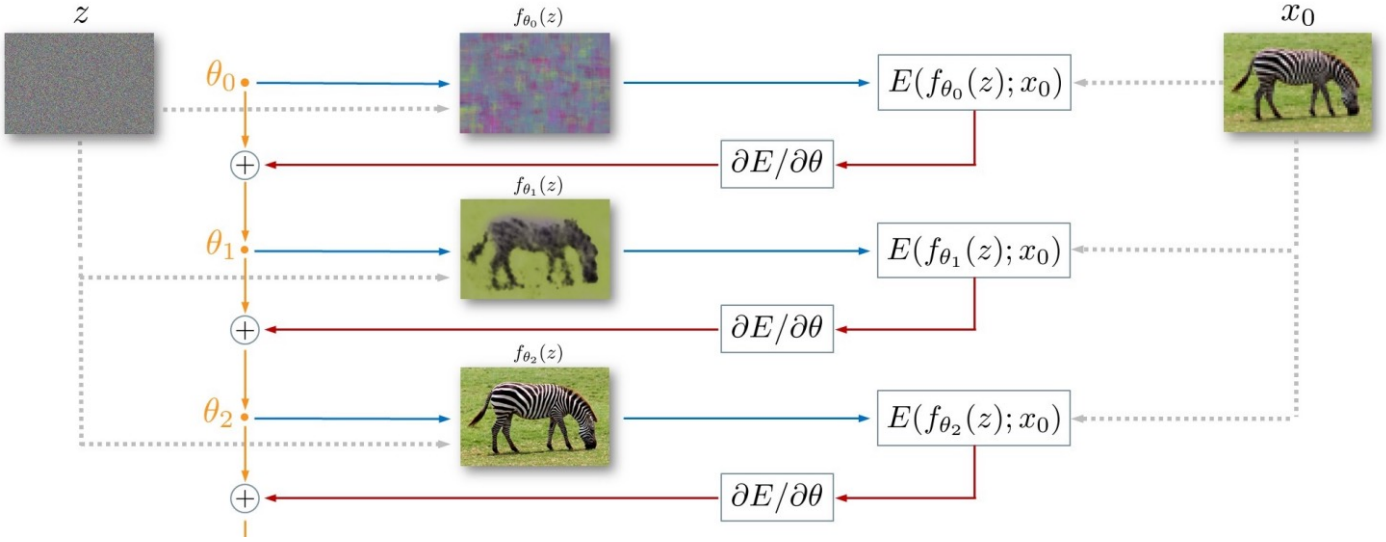
$$\theta^* = \text{argmin}_\theta \{\|f_\theta(z) - x_0\|^2\} \qquad x^* = f_{\theta^*}(z)$$

Why $x^* \neq x_0$?

- The neural network cannot represent **any** image
- The minimization step can be stopped **before convergence**

[Ulyanov '18]

Deep image prior

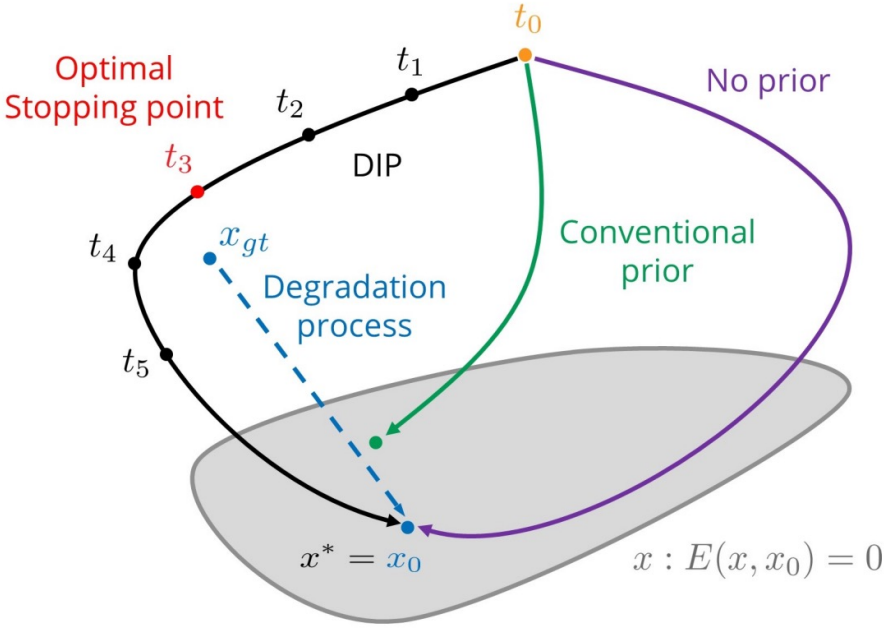


No need to collect clean images! (no training step here)

Must be optimized from scratch every time

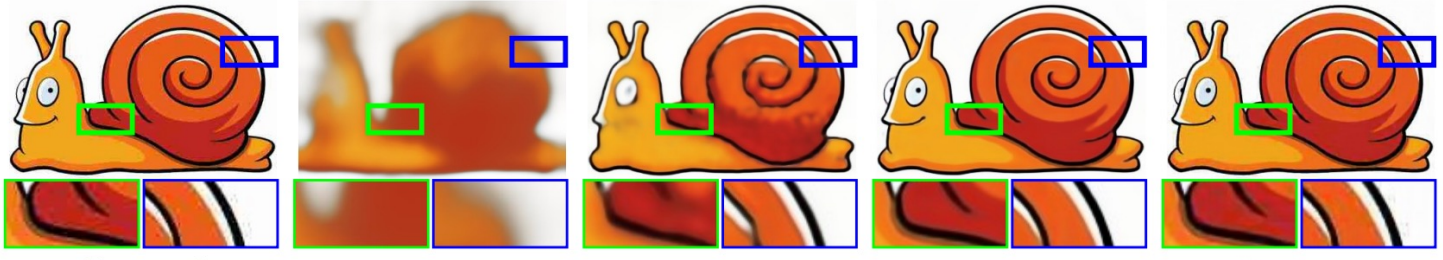
[Ulyanov '18]

Deep Image Prior



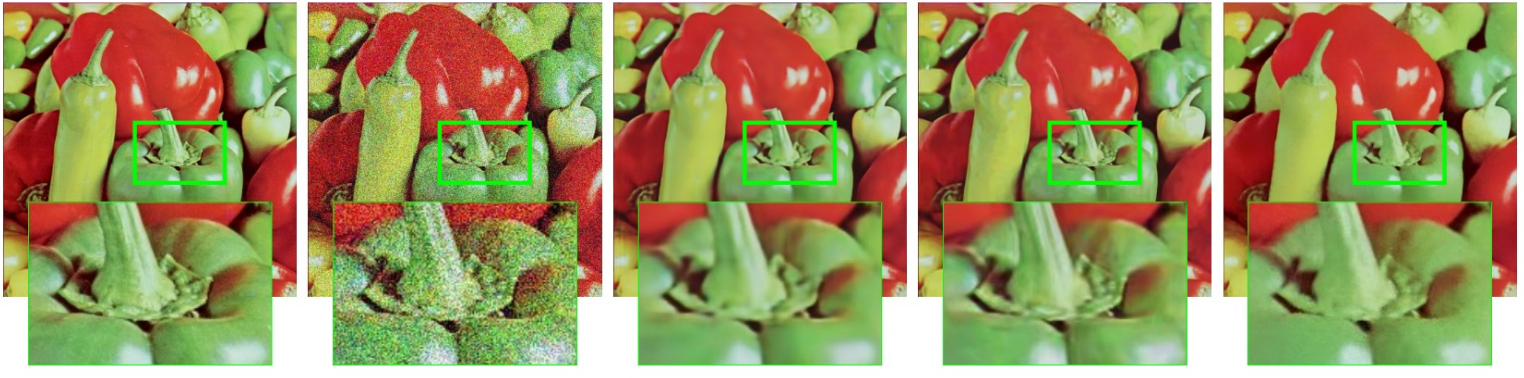
[Ulyanov '18]

Deep image prior



[Ulyanov ‘18]

Deep image prior



(a) GT (b) Input (c) Ours (d) CBM3D (e) NLM

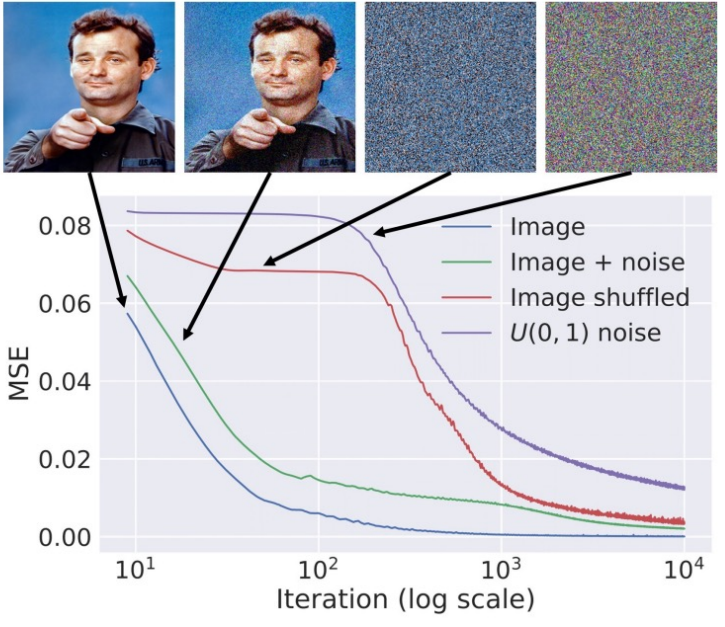
[Ulyanov ‘18]

Deep image prior

As we show in the experiments, the choice of architecture does have an impact on the results. In particular, most of our experiments are performed using a U-Net-like "hourglass" architecture with skip connections, where z and x have the same spatial dimensions and the network has several millions of parameters. Furthermore, while it is also possible to optimize over the code z, in our experiments we do not do so. Thus, unless noted otherwise, z is a fixed randomly-initialized $3D$ tensor.

[Ulyanov '18]

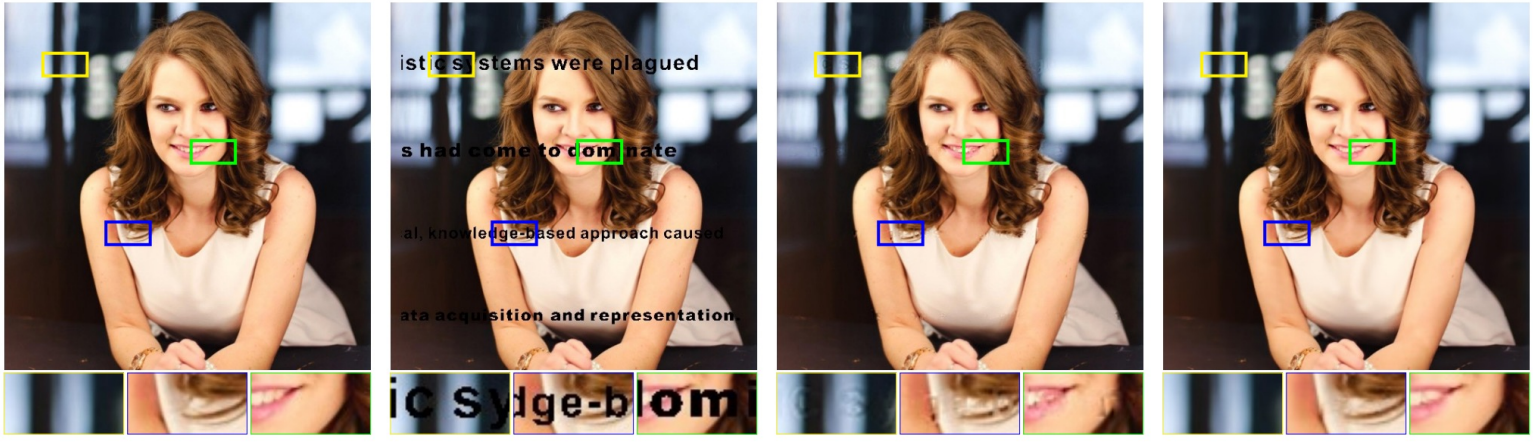
Deep image prior



The network fits real images faster than noise

[Ulyanov ‘18]

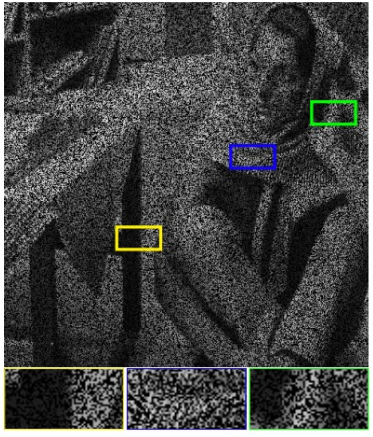
Deep image prior



[Ulyanov ‘18]

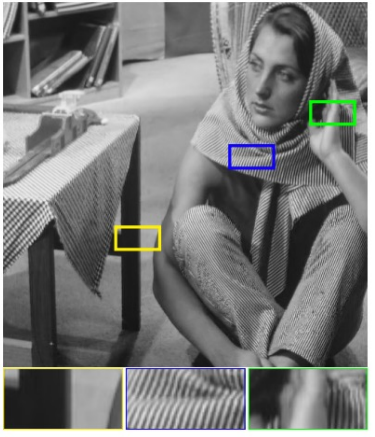
Deep image prior

GT



Corrupted

CSC

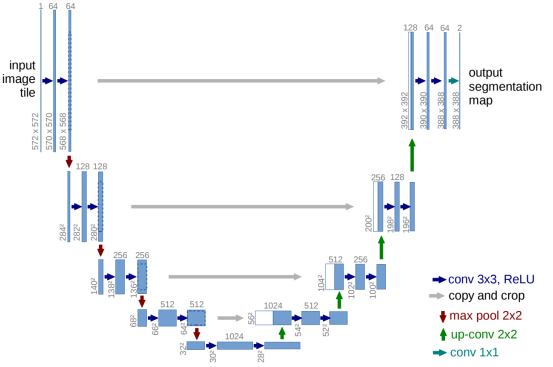


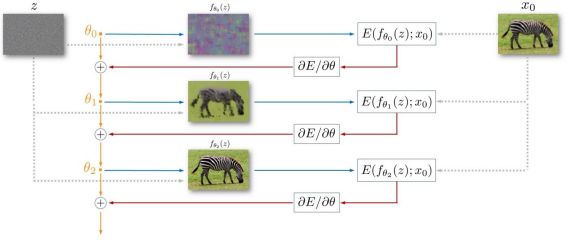
DIP

[Ulyanov ‘18]

Summary

- CNNs
- Building blocks of CNNs
- Deep networks: ResNets & U-Nets
- Application:
  - Learning to see in the dark
  - Deep optics for HDR imaging
  - Relighting
  - Noise2Noise
  - Deep Image Prior





References and Further Reading

slides adapted from Stanford CS231N: http://cs231n.stanford.edu/slides/

Araujo, André, Wade Norris, and Jack Sim. "Computing receptive fields of convolutional neural networks." *Distill* 4.11 (2019)

Chen, Chen, et al. "Learning to see in the dark." *Proc. CVPR*. 2018.

Eigen, David, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network." *Proc. NeurIPS*. (2014).

Farabet, Clement, et al. "Learning hierarchical features for scene labeling." *IEEE TPAMI*. 35.8 (2012): 1915-1929.

He, Kaiming, et al. "Deep residual learning for image recognition." Proc. CVPR. 2016.

Hubel, David H., and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." *The Journal of physiology* 148.3 (1959): 574-591.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Proc. NeurIPS* 25 (2012): 1097-1105.

Lehtinen, Jaakko, et al. "Noise2Noise: Learning image restoration without clean data." arXiv preprint arXiv:1803.04189 (2018).

Lim, Bee, et al. "Enhanced deep residual networks for single image super-resolution." *Proc. CVPR Workshops*. 2017.

Lin, Haoning, Zhenwei Shi, and Zhengxia Zou. "Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network." *Remote sens.* 9.5 (2017): 480.

Metzler, Christopher A., et al. "Deep optics for single-shot high-dynamic-range imaging." Proc. CVPR. 2020.

Nah, Seungjun, Tae Hyun Kim, and Kyoung Mu Lee. "Deep multi-scale convolutional neural network for dynamic scene deblurring." *Proc. CVPR*. 2017.

Odena, Augustus, Vincent Dumoulin, and Chris Olah. "Deconvolution and checkerboard artifacts." *Distill* 1.10 (2016)

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert. "Feature visualization." *Distill* 2.11 (2017)

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. 2015.

Ren, Shaoqing, et al. "Faster R-CNN: towards real-time object detection with region proposal networks." *IEEE TPAMI*. 39.6 (2016): 1137-1149.

Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *IJCV* 115.3 (2015): 211-252.

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *Proc. ICLR* (2014).

Sun, Tiancheng, et al. "Single image portrait relighting." *ACM Trans. Graph.* 38.4 (2019): 79-1.

Toshev, Alexander, and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks." *Proc. CVPR*. 2014.

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." *Proc. CVPR*. 2018.

Zhang, Kai, et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising." *IEEE Trans. Imag. Proc.* 26.7 (2017): 3142-3155.

We just scratched the surface!

- Image/video generation
    - Generative adversarial networks (GANs)
    - Diffusion models & flow-based models
    - Autoregressive models (image + text)
- 3D generation/reconstruction
    - Neural radiance fields
    - Gaussian mixture models
    - …

To learn more about these : CS229, CS231N, CS236, CS348I, CS228