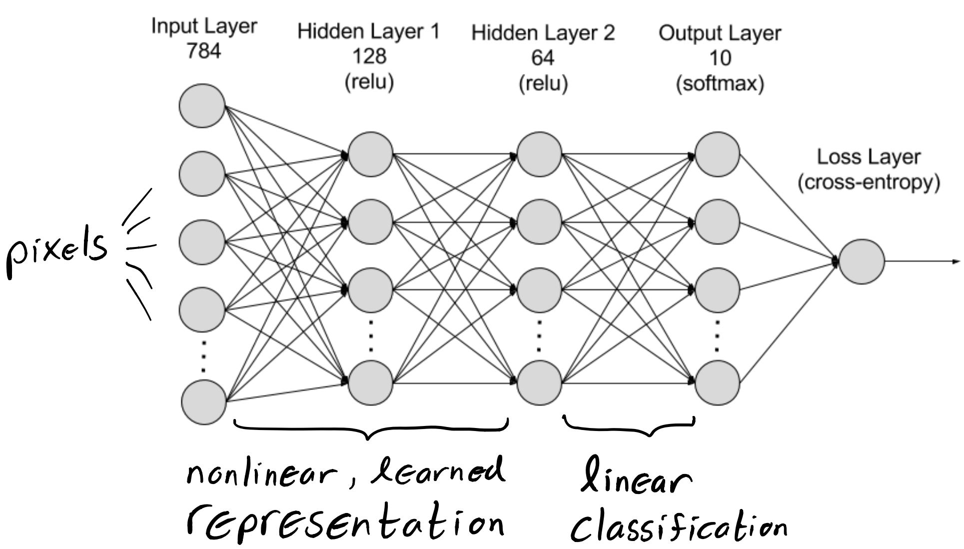
Neural Network Architectures for Images

by Paul Hand
Northeastern University

Outline

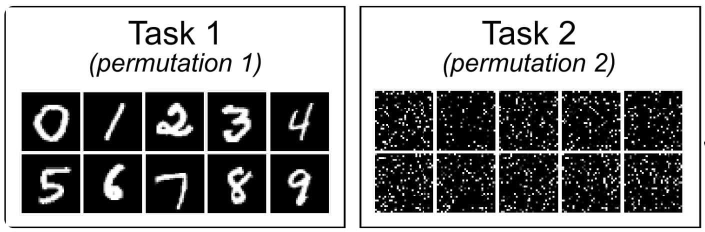
- MLPs
- CNNs
- ResNets
- Encoder-decoder nets
- Autoencoders

Multilayer Perceptrons (MLPs)



Careful:
- can't alter input size
- no geometry/locality of input is enforced
- permutation invariance

rearranges order of pixels



Let P be a random fixed perm. matrix
$D = \{(Px_i, y_i)\}$

$D = \{(x_i, y_i)\}$

permuted mnist
- choose a fixed random permutation
- apply to MNIST

Exercise: MNIST Digit classification (plain and after a fixed random permutation)

Which would you expect to perform better and why:

—MLP vs CNN on Task 1:

CNN – spatial data – leverages translational invariance of conv
small img patches present meaningful features (locality)

—MLP vs CNN on Task 2:

MLP

No edges / no local info
poorly suited w/ struct. of conv
MLP should be able to learn decision rules as they are universal approx

—Task 1 vs Task 2 using MLP:

exactly equal

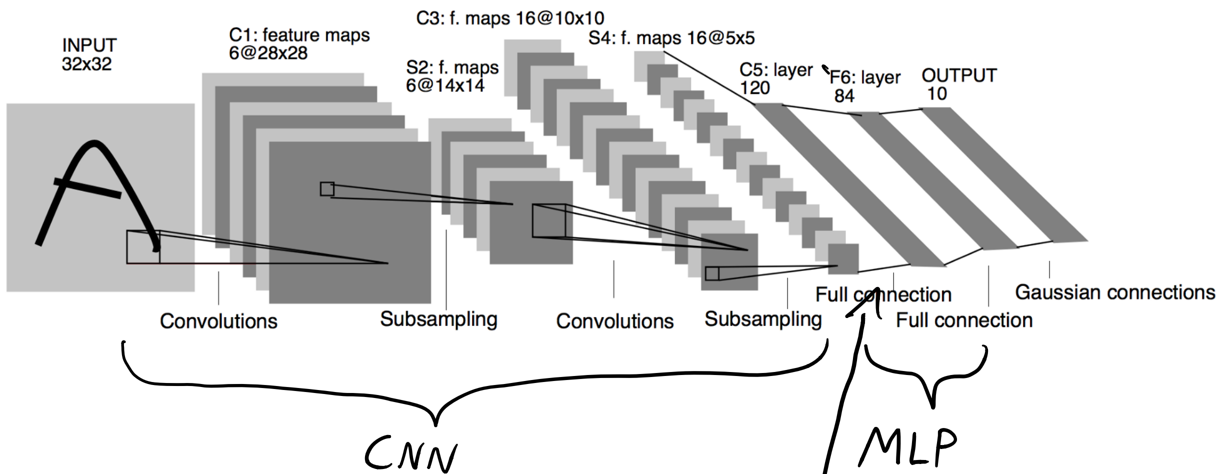
—Task 1 vs Task 2 using CNN:

1 much better than 2

Convolutional Neural Networks + MLPs

Context: Classification

(Lecun et al. 1998)



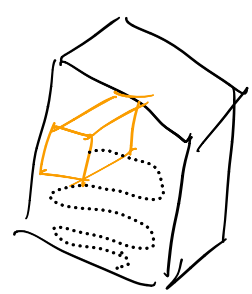
Suppose you passed in a 300x300 image of a single handwritten digit into this network. Would it still work?

CNN would still "work"
will give too many outputs
Can not use MLP

What if we rescaled the output of the CNN (right before passing it into the MLP)?

Kernel sizes are the same and hence with the larger resolution input

The field of view of any neuron will be drastically altered



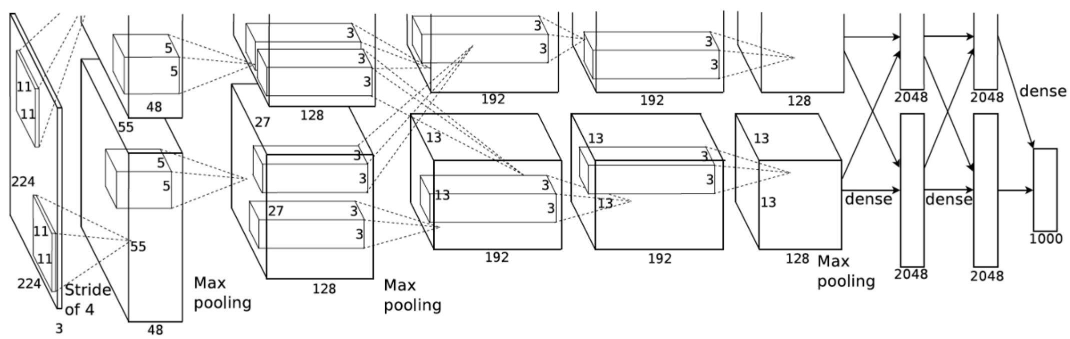
No, wouldnt work.
Wouldnt give comparable values

Convolution Layers are well suited for images

- Leverages locality / geometry of images
- Enforce translation invariance
- Fewer parameters than MLP
- Can handle images of arbitrary size
  (though following MLP can not)

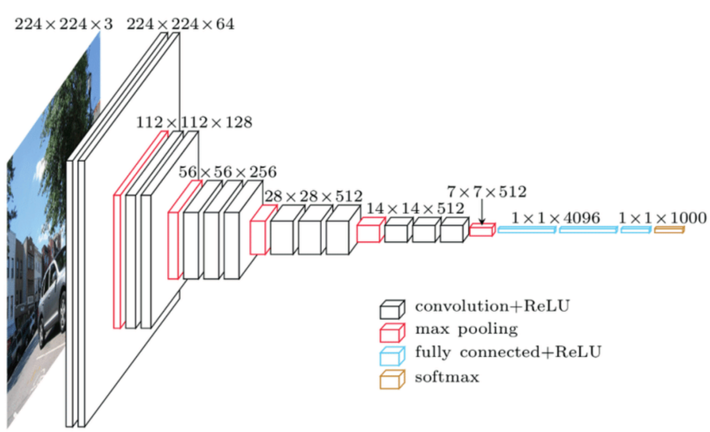
AlexNet:

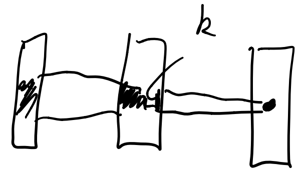
(Krizhevsky et al. 2012)



VGG Net

(Simonyan + Zisserman 2015)



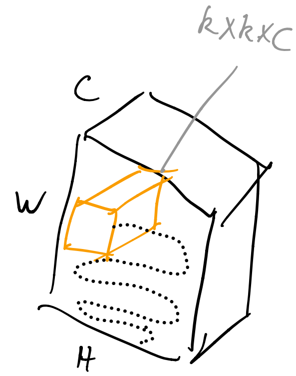


Exercise: Large filters or small filters

You have d convolutions layers with filter size k x k.

How does the width of the receptive of a neuron field depend on d and k?

$$\underbrace{F}_{\text{receptive field}} \approx d \cdot k$$



How does parameter count depend on d and k?

$$k^2 d = F \cdot k$$

Are large filters or small filters more economical in parameter count for a given receptive field?

For a fixed receptive field, smaller k uses smaller parameter count

Vanilla CNNs

Context: Image Restoration

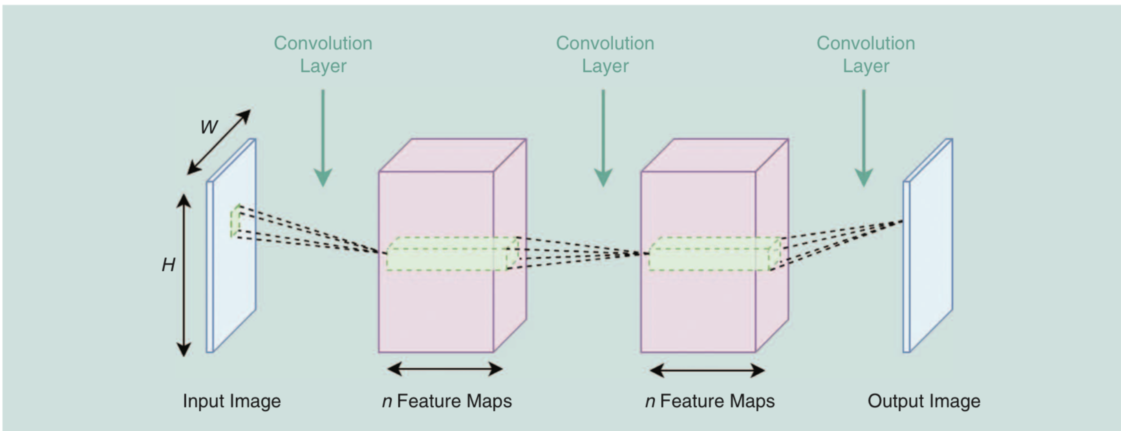


FIGURE 3. A three-layer CNN with successive convolutional layers, where the spatial dimensions of the feature maps match those of the input and output images. Following each convolution there is a nonlinearity operation, not shown here.

(Lucas et al. 2018)

Can retain image size

Can operate on image of any size

Be careful about
- receptive fields
- image scale

Dirt removal

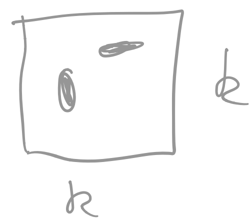
(Eigen et al. 2013)

(a) (b) (c) (d)

Consider this fully convolution network for all dirt removal.

If you passed in 10x increased resolution photograph of the same content, would you expect it to work?

No. See previous exercise



If you passed in a 10x increased field of view (at the same resolution), would you expect it to work?

Yes, the scale of the features learned from the convolutions would be the same in the new image.

Blind Deconvolution

(Hradiš et al. 2015)

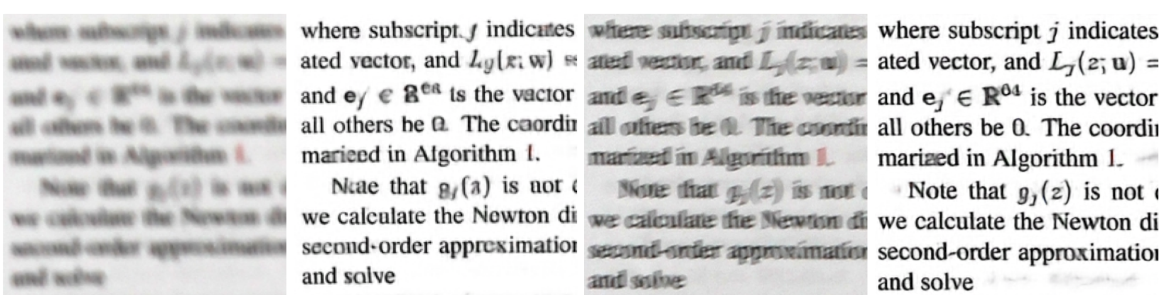


Table 1: CNN architecture – filter size and number of channels for each layer.

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L15	19×19	1×1	1×1	1×1	1×1	3×3	1×1	5×5	5×5	3×3	5×5	5×5	1×1	7×7	7×7
	128	320	320	320	128	128	512	128	128	128	128	128	256	64	3

Residual Blocks + Skip Connections

(Lucas et al. 2018)

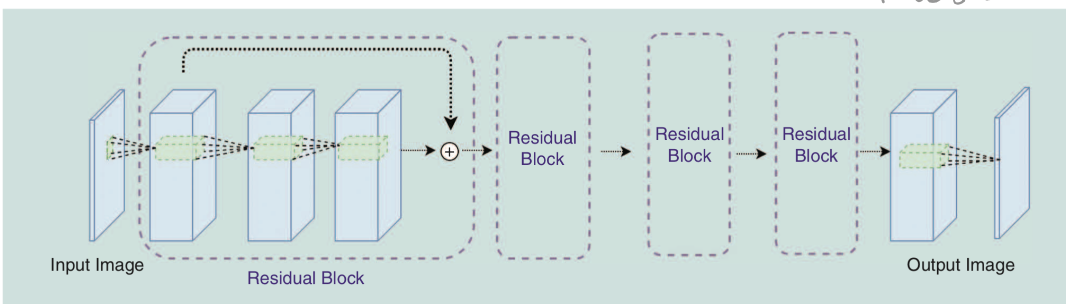
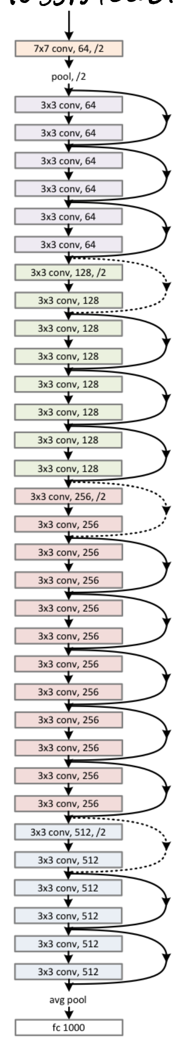
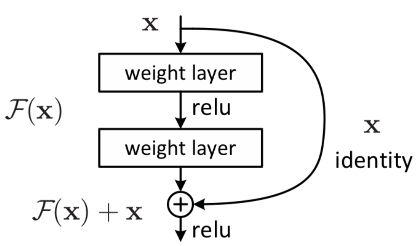


FIGURE 4. An example of a deep residual CNN. Each residual block, consisting here of three convolutions, learns a residual between its input and its output.

ResNet (for classification):

(He et al. 2015)





Ensure that enough computation is performed before adding back

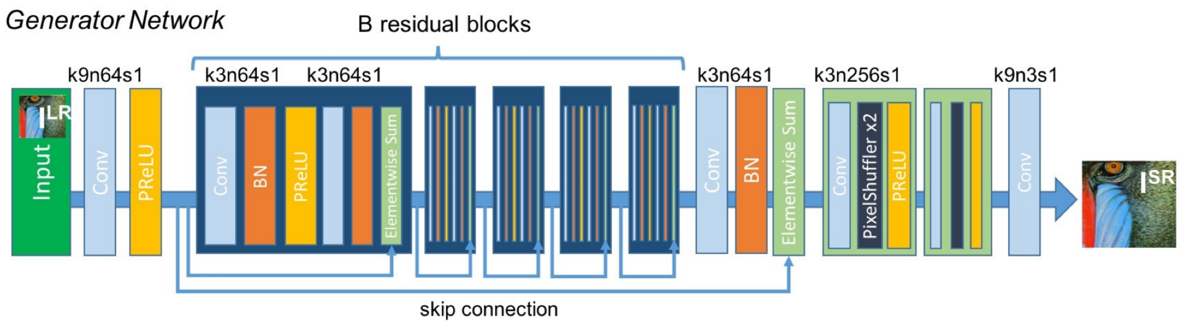
Figure 2. Residual learning: a building block.

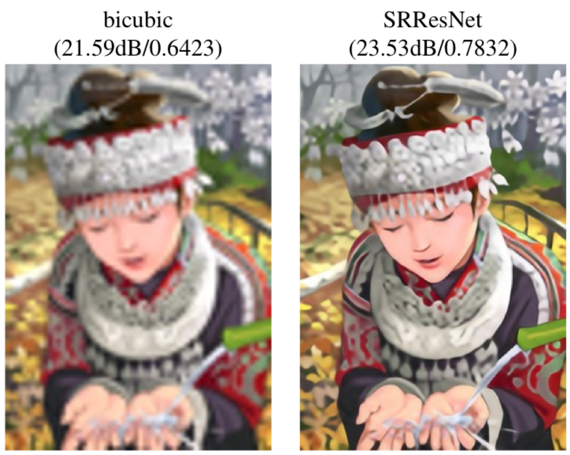
Residual blocks allow increased depth

Increased depth allows
- large receptive fields
- Smaller filter sizes (Savings in parameter count)

ResNet (for Superresolution):

(Ledig et al. 2017)



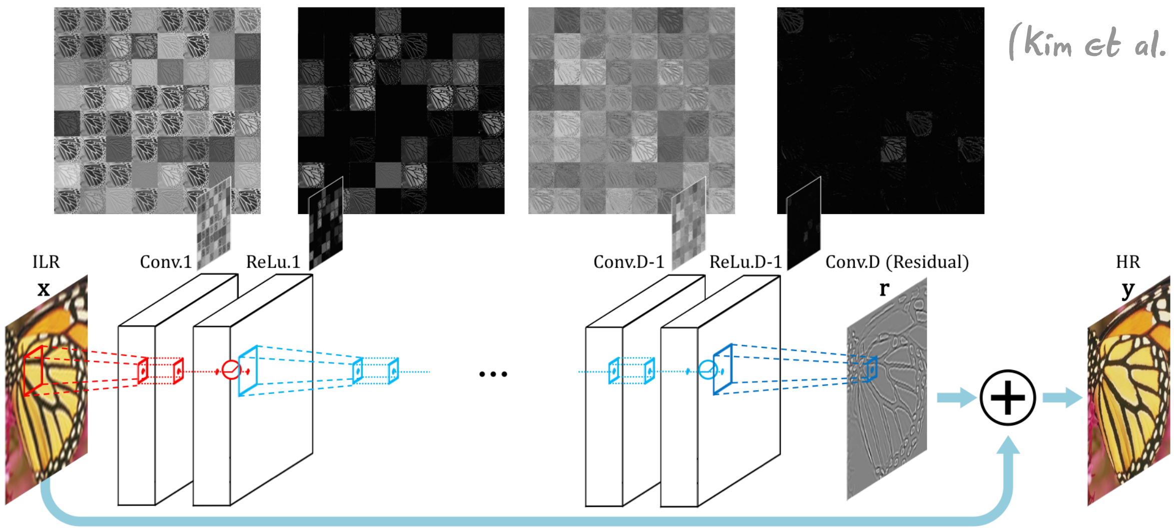


ResNets are good at

- Successive refinement of images

Idea: Do superresolution by only learning residual

(Kim et al. 2016)



Learning a residual may be easier than learning a full mapping

analogy: fixing a painting

Encoder-decoder Nets

(Lucas et al. 2018)

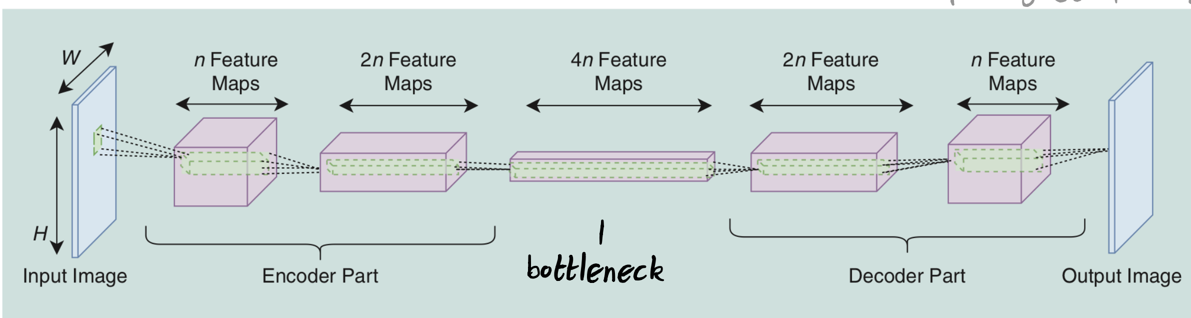


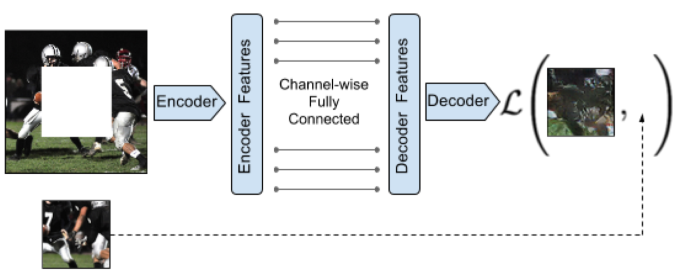
FIGURE 5. In an encoder-decoder CNN, the feature maps are spatially compressed by an encoder network, then increased back to the size of the output image by a decoder network.

Encoder converts image to a set of latent variables (a code)
Decoder generates image from this code

Bottleneck forces net to gain semantic understanding of image

Encoder-decoder net for inpainting:

(Pathak et al. 2016)



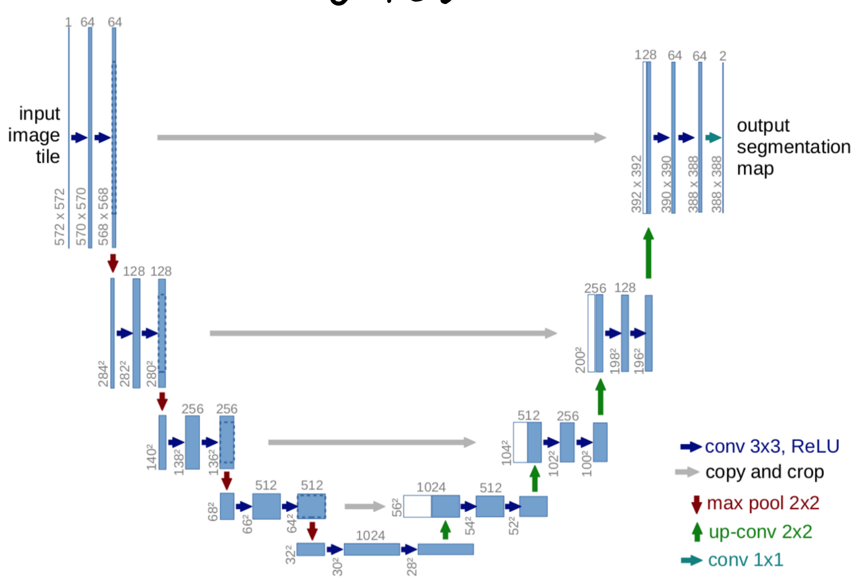
Combining encoder-decoder nets w/ skip connections

The compressive nature of an encoder-decoder bottleneck may result in loss of detail in decoded images.

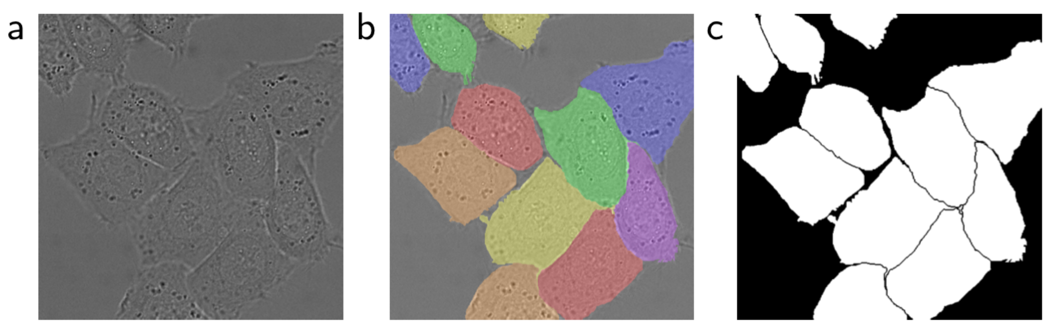
Idea: fix with skip connections

U-net

(Ronneberger et al. 2015)



Segmentation (biomedical images)



Strength of U-nets:

- semantic understanding by bottleneck
- skip connections preserve detail

What sort of architecture would be reasonable to select for the following tasks. Vanilla CNN, ResNet, Encoder-Decoder, U-net

A) Remove some of the motion blur due to someone holding a cell phone camera

Resnet, VCNN

B) You are given a black and white image and you want to colorize it

Unet

C) You are given an image where 1% of pixels are missing. You want to inpaint those pixels

Depends on which pixels are missing.

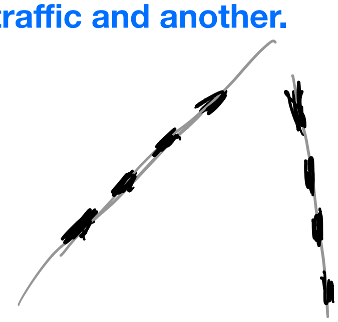
low semantic

If pixels are missing at random, I would try vanilla CNN / ResNet
If solid block of pixels is missing, I might also consider encoder-decoder

highly semantic

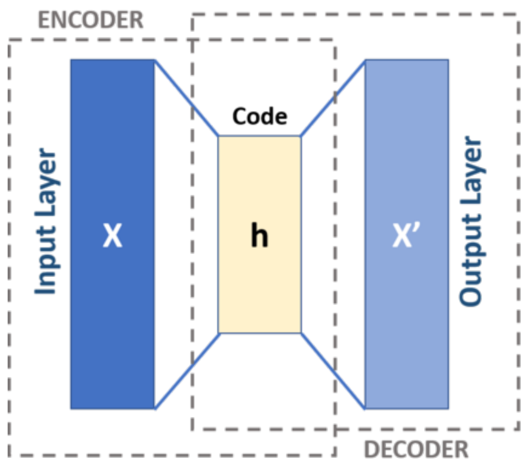
D) You have photographs of a road and you want to decide which pixels correspond to the boundary between one lane of traffic and another.

Unet

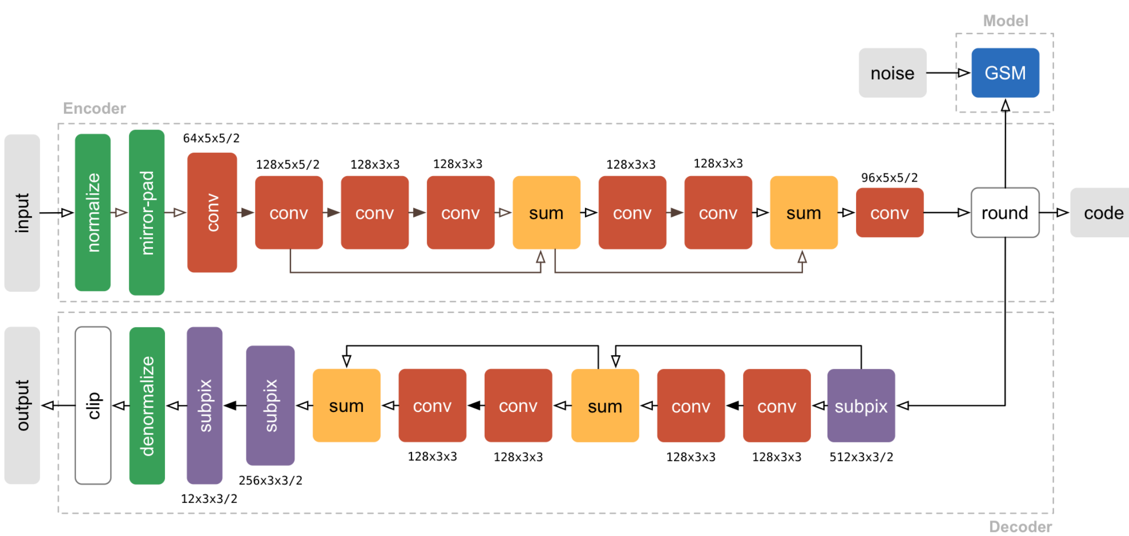


Auto encoders

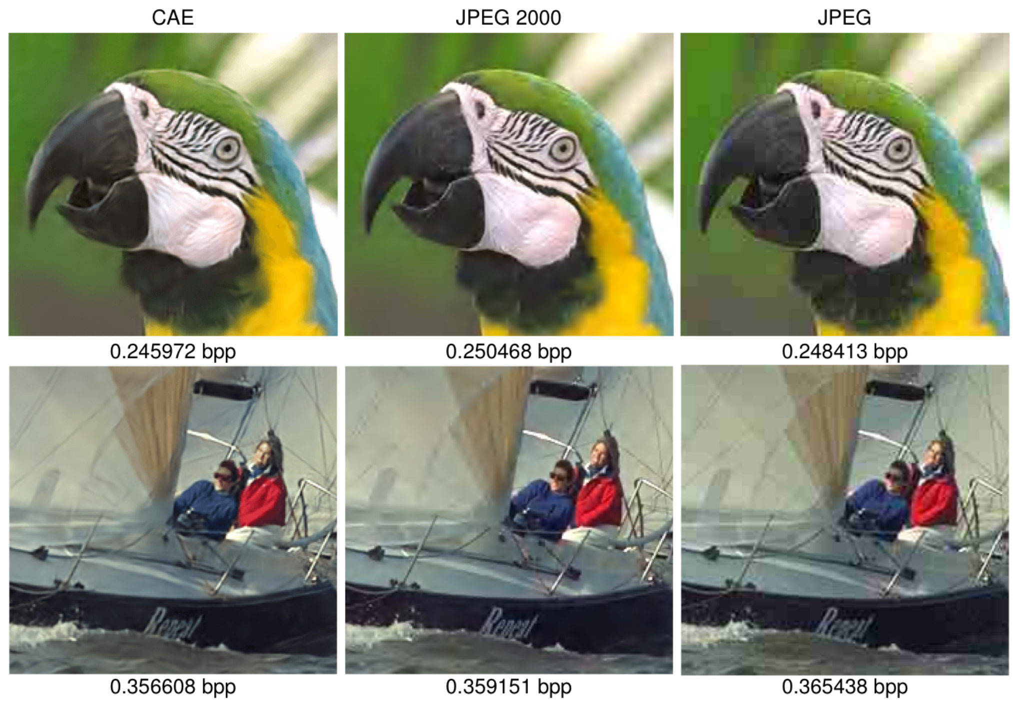
Context: unsupervised representation learning



With compressive autoencoder, one can compress images

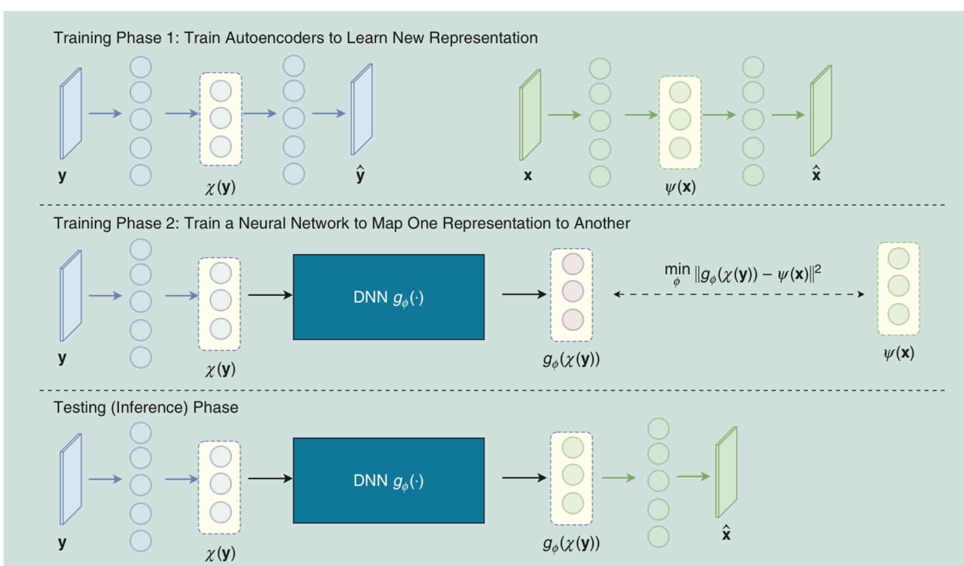


(Theis et al. 2017)



(Theis et al. 2017)

Learned autoencoder representations can be used for other tasks



(Zeng et al. 2017)

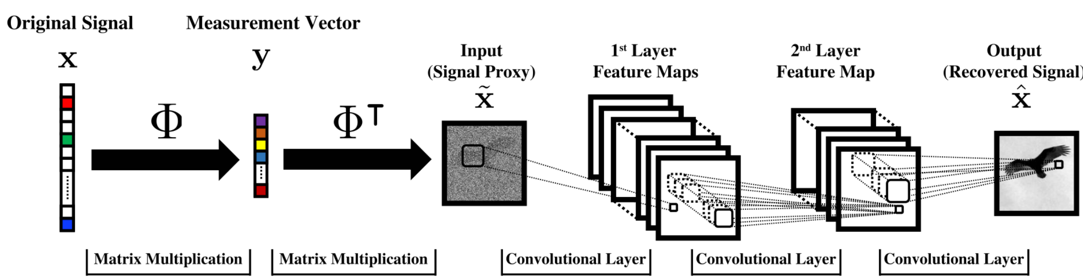
FIGURE 6. An example of an approach in which new representations for images are learned, prior to solving the reconstruction problem in a supervised way. In Zeng et al.'s work [37], autoencoders first learn new features for the LR and HR patches (training phase 1). An MLP is then trained to map the representation of the observed LR patch to that of the HR patch (training phase 2). The final HR patch can be obtained with the second half of the autoencoder trained to reconstruct HR images (testing phase).

One last idea: Spare nets from learning things you know

(fully end-to-end trained nets may not be the best approach to solving a problem)

Compressed Sensing:

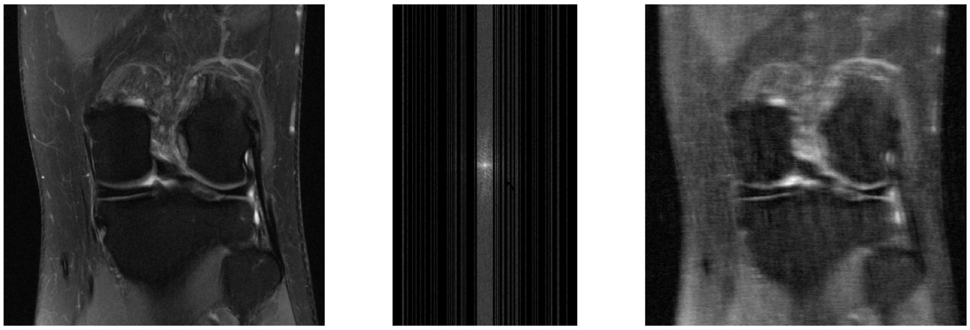
(Mousavi + Baraniuk 2017)



What is compressed sensing?

MRI reconstruction:

(Zbontar et al. 2019)



(a) Cropped and vertically flipped reconstruction from fully sampled k-space data

(b) Rectangular masked k-space

(c) Reconstruction via zero-filled IFFT

Would it make sense to use a convolutional neural network to solve end-to-end compressed sensing with generic measurement matrix Phi?