Last block:

Classify an image into one of two classes by

- Stack some pooling, FC layers on an encoder
- Adjust result into a vector
- Pass to linear classifier

Learn all parameters with SGD and various losses

- get training examples right

Idea:

- exploit this machinery to improve training
- exploit this machinery to segment image

Improving a denoiser with a classifier

Idea:

if a classifier can tell the difference between a denoised image and a real one, the denoiser isn't working right

use the classifier to improve the denoiser

Easy to state, quite delicate in practice

Obvious strategy that DOESN'T WORK -I

21.1.1 The Obvious Strategy doesn't Work

Imagine we have a classifier that is good at telling whether an image has been through a particular autoencoder or not. You might use this classifier as a loss, by the following argument. Recall Section 20.2.2 interpreted $u(\mathbf{x}; \mathbf{a}, b)$ in terms of a probability with the model

$$u(\mathbf{x}; \mathbf{a}, b) = \log \left[\frac{P(\text{denoise}|\mathbf{x})}{P(\text{real}|\mathbf{x})} \right].$$

so that a data item with positive u is likely to be a denoised image, and more likely to be denoised if u is larger. In turn, one could interpret $u(\mathbf{x}; \mathbf{a}, b)$ as a loss. A more realistic set of reconstructions would have a smaller value of

$$\sum_{i \in \text{ae outputs}} u(\mathbf{x}; \mathbf{a}, b).$$

You could do this whether you use the hinge loss or the cross-entropy loss to train the classifier (**exercises**).

Obvious strategy that DOESN'T WORK -II

Strategy:

train autoencoder, fix
make a lot of denoised images
train classifier to spot denoised images
continue to train autoencoder, using loss from previous slide

Obvious strategy that DOESN'T WORK -III

Fails (easy – try it!)

Fails because:

Classifier is good at spotting some error that is characteristic of the autoencoder; if the autoencoder learns NOT TO MAKE THAT ERROR, it'll be fine – but it could make some other, new error.

But the classifier can't spot that!

More effective strategy

Repeat:

- Fix an autoencoder, then adjust a classifier slightly using the outputs of this autoencoder; that is, use the autoencoder to produce a set of outputs, label them, take some steps to minimize

$$\sum_{i \in \text{data}} \mathcal{L}_c(s_i(\theta_c))$$

  (the training loss of the classifier) as a function of θ_c.

- Fix the classifier, and adjust the autoencoder to fool the classifier; that is, take some steps to reduce

$$\sum_i \mathcal{L}(\mathcal{A}(\mathcal{N}_i; \theta_a), \mathcal{C}_i) + \lambda \sum_i -u(\mathcal{A}(\mathcal{N}_i; \theta_a); \theta_c)$$

  as a function of θ_a.

More effective strategy

Notice the classifier isn't really producing a loss

because each time you improve the autoencoder, you are using a slightly different classifier

But it does give you a gradient...

The classifier is often referred to as an adversary

The value it makes is an adversarial loss

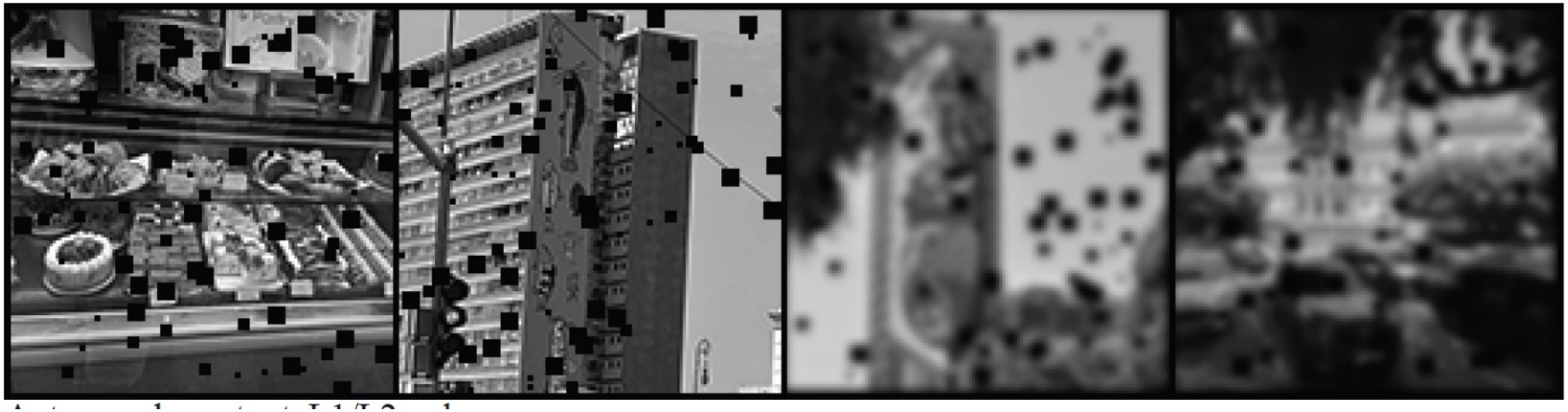
Tricky to get right, but very good when it does work

Issues:

- Goldilocks problem:
  - if the classifier is “too stupid”, the gradient is no use
    - autoencoder beats classifier
  - if the classifier is ”too smart”, the gradient is no use
    - classifier beats autoencoder

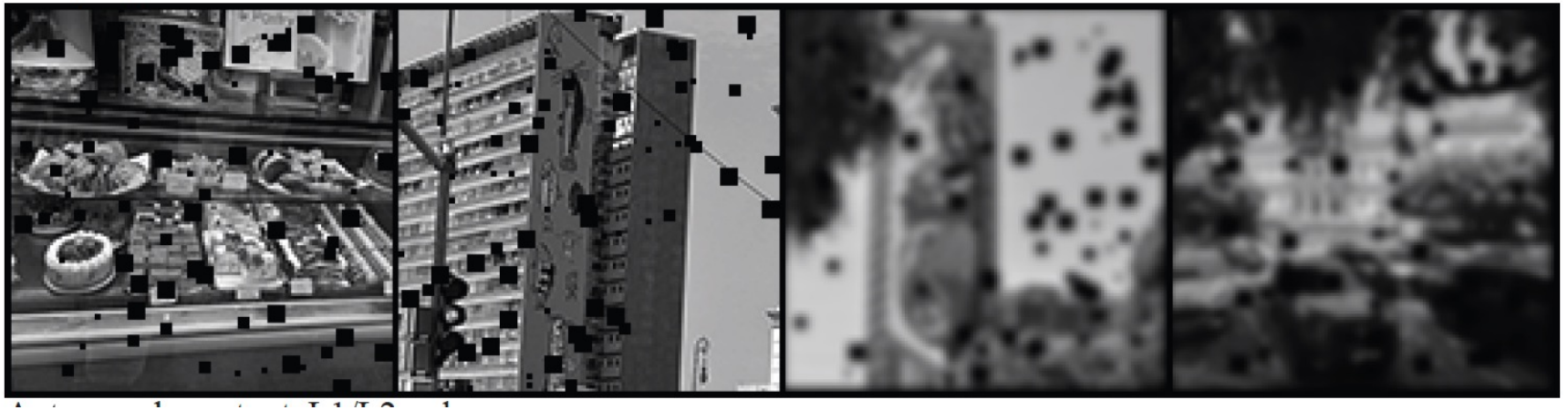
- How much training should classifier/autoencoder get?

Noisy input



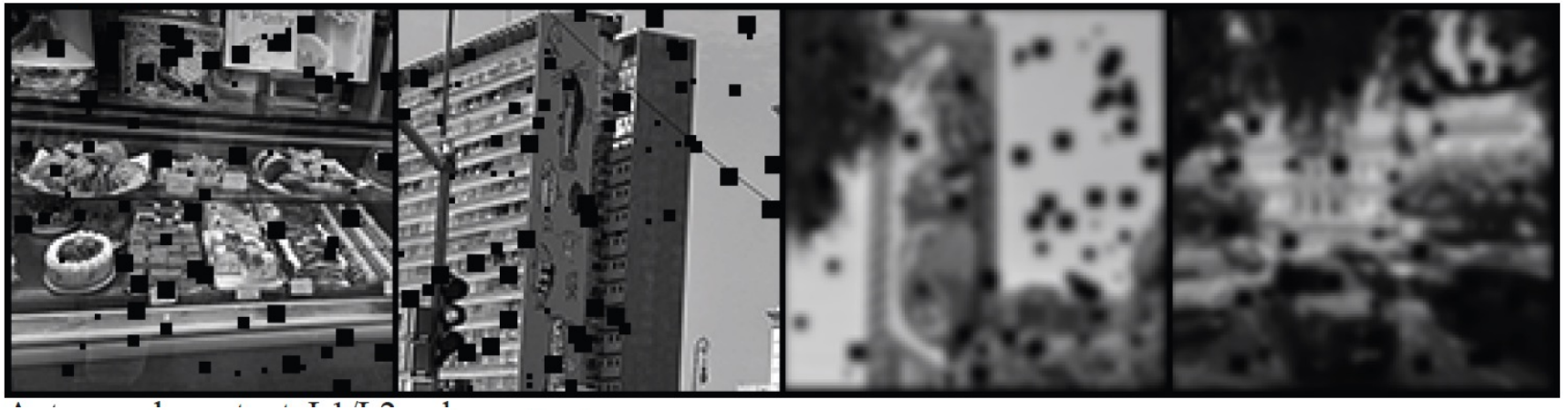
Autoencoder output, L1/L2 only

Noisy input

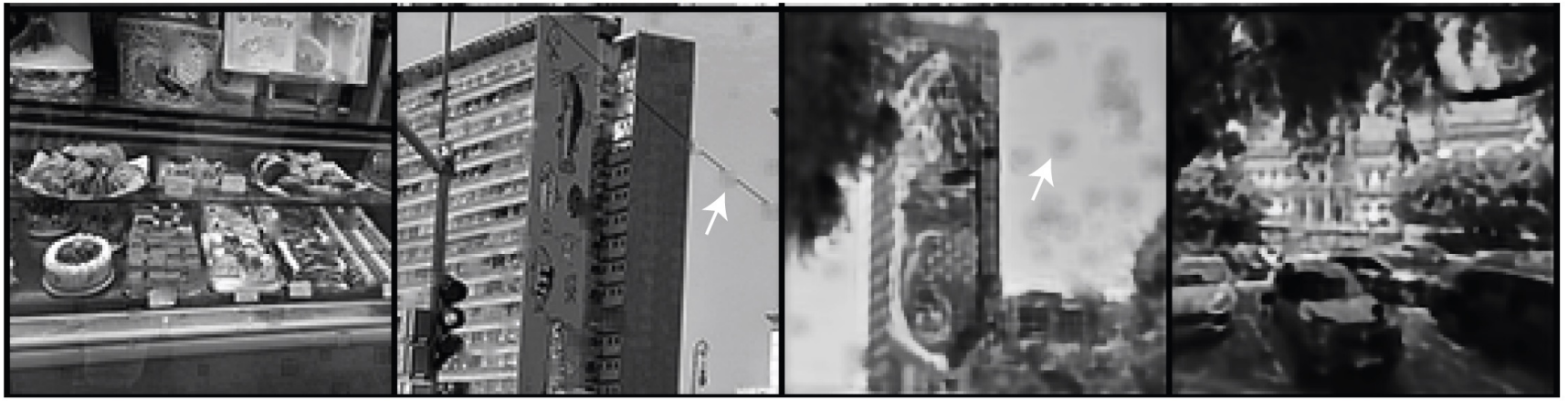


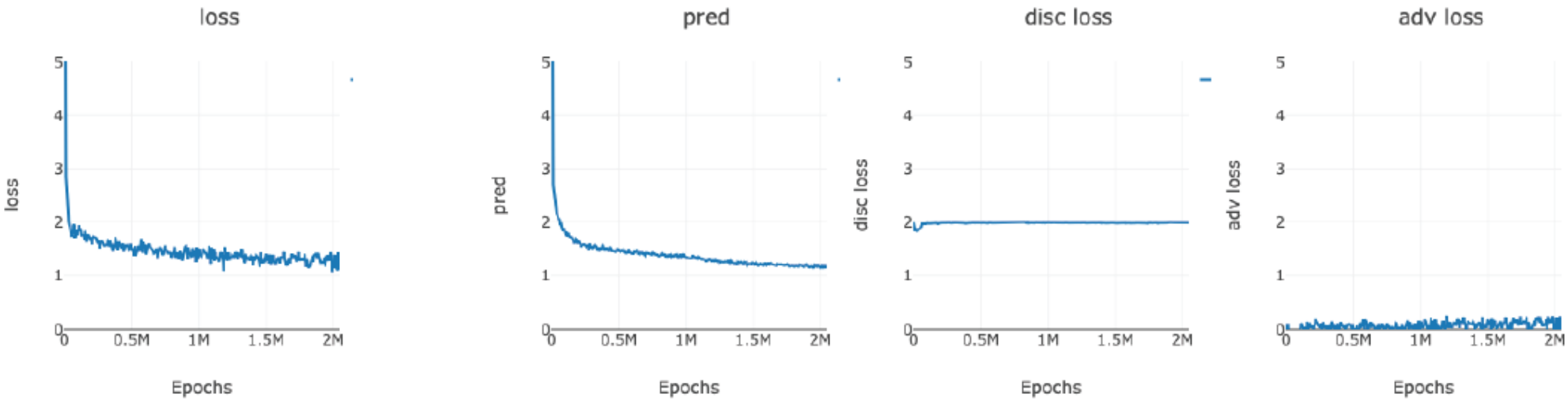
Autoencoder output, L1/L2 and big adversary

Noisy input



Autoencoder output, L1/L2 and small adversary




loss
pred
disc loss
adv loss
loss
pred
disc loss
adv loss
0
1
2
3
4
5
0
0.5M
1M
1.5M
2M
Epochs