EECS 182/282A

Reading: Prince through Ch 11 + Ch 13
Note: Little in this lecture is in the Prince textbook. (Just 9.3.7)

Today: Self-supervision
State-space models

Architecture Order In Class:

MLPs $\longrightarrow$ CNNs $\longrightarrow$ Graph NN $\longrightarrow$ $\boxed{\text{RNN/State-space}}$ $\longrightarrow$ Transformers

General Principle: Self-supervision

"I need labels to train, I don't have labels, so make my own labels from data"

Lessons from example of trying to learn what you can in the Kalman Filtering context:
1) We can learn a partial pattern that can be useful
2) Might need scaffolding parts of my NN.
3) Generic idea of "next-thing" prediction in causal sequence modeling

Step Back: Connect to unsupervised learning in classic ML.

Two approaches: 1) Dimensionality Reduction
2) Clustering

Recall Dimensionality Reduction: Think about PCA

All we have are $\{\vec{x}_i\}_{i=1}^{N}$ ← d-dim   Unlabeled Data From Interesting Distribution

Classic Recipe (Neglecting Means):
1) Construct $X = \begin{bmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix}$

2) Compute SVD $X = U \Sigma V^T = \sum_i \sigma_i \vec{u}_i \vec{v}_i^T$

3) Keep top k $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k$ singular vectors to use for dim-reduction

4) Given some other problem $\vec{x} \longrightarrow \left.\begin{bmatrix} \vec{v}_1^T \vec{x} \\ \vdots \\ \vec{v}_k^T \vec{x} \end{bmatrix}\right\}$ k-dim Features

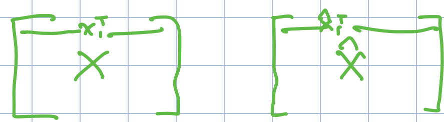
Classic Perspective has no loss, no labels, no gradients, no optimizer. No mini-batches, ...

Why was this reasonable?

Recall Eckhart-Young-Mirsky Theorem for Frobenius Norm

Given X, $\hat{X} = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T$ is the rank-k matrix that minimizes

$$\| X - \hat{X} \|_F^2$$



Let's interpret this in neural-net terms: (row by row perspective on $\hat{X}$)

$$\underset{d\text{-dim}}{\vec{x}} \to \boxed{W_1} \to \underset{k\text{-dim}}{\vec{a}} \to \boxed{W_2} \to \underset{d\text{-dim}}{\hat{\vec{x}}} \qquad \text{to minimize } \| \vec{x} - \hat{\vec{x}} \|_2^2$$

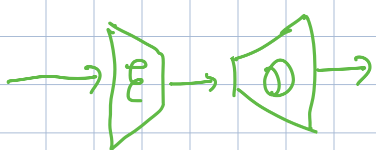
Linear   Linear

Trained over the entire data set $\{ \vec{x}_i \}_{i=1}^{N}$

If this reaches the minimizer, it has to find the same subspace to project into using W_1 as the first k s.v. $\vec{v}_1, \dots, \vec{v}_k$.

This approach has a name: <u>Auto-encoders</u>

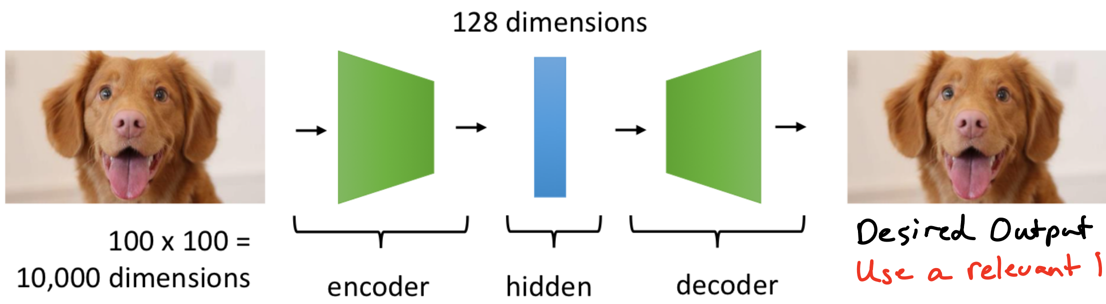
Core Ingredients: Labels are $\vec{x}_i$ itself.
Architecture has an encoder followed by a decoder
Bottleneck in the middle.



Traditional Perspective: Decoder is scaffolding.

Some Pictures...

Figures From Levine, 182 Lec 17, 2021

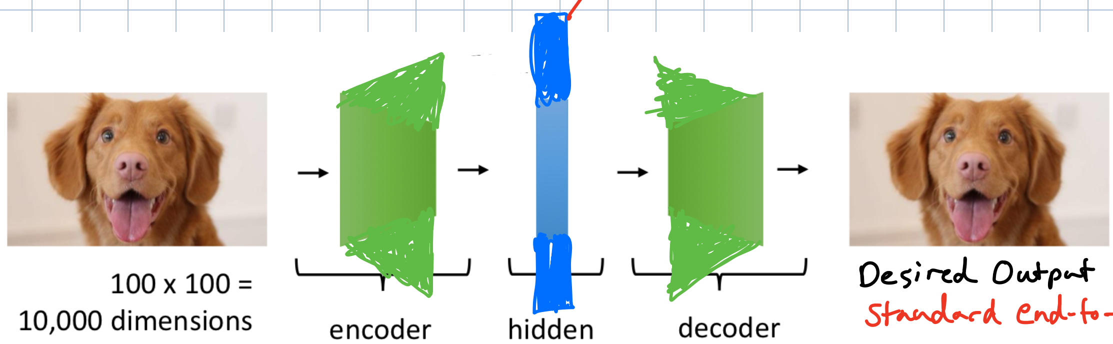


Desired Output

Use a relevant loss function to train

Classic Autoencoder... Into the encoder & decoder boxes goes a Neural Net Architecture choice tied to domain. E.g. CNN for image.

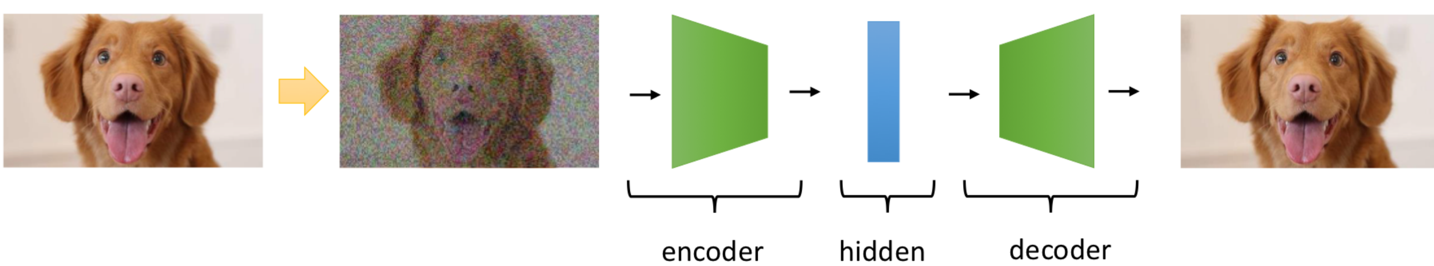
Aux Loss that is sparsity-seeking eg $\|\cdot\|_1$ penalty



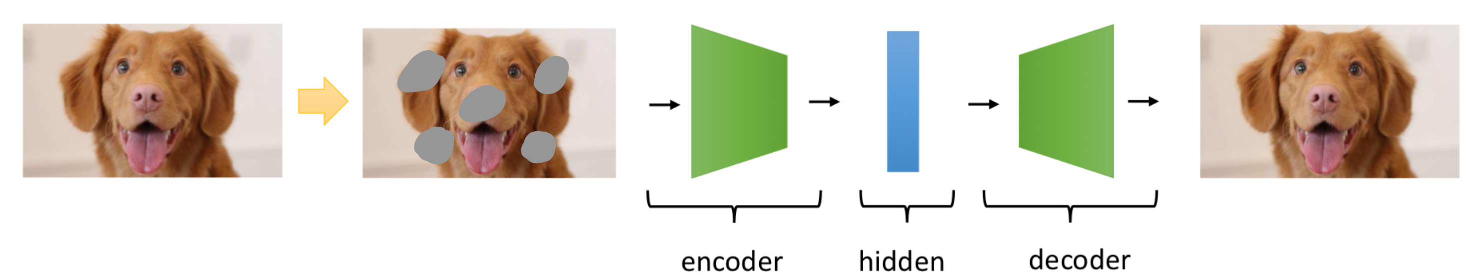
Desired Output

Standard end-to-end loss

Sparse Autoencoder: Can make hidden "bottleneck" even bigger than the original but impose an auxilliary loss.



Denoising Autoencoder: Instead of regularizing with a bottleneck or aux loss, use data augmentation (e.g. adding noise)



Masked Autoencoders: Same spirit as denoising autoencoders, but the data augmentation is random masking

Note: Can involve learned ? vectors if this is appropriate.

Note from O.H. In our default perspective, reconstruction loss is on everything. But there's a variant where it is only on masked places.

Recall Perspective on Traditional Unsupervised learning in standard ML

Two approaches: 1) Dimensionality Reduction ⟺ Autoencoder-style Self-Supervision

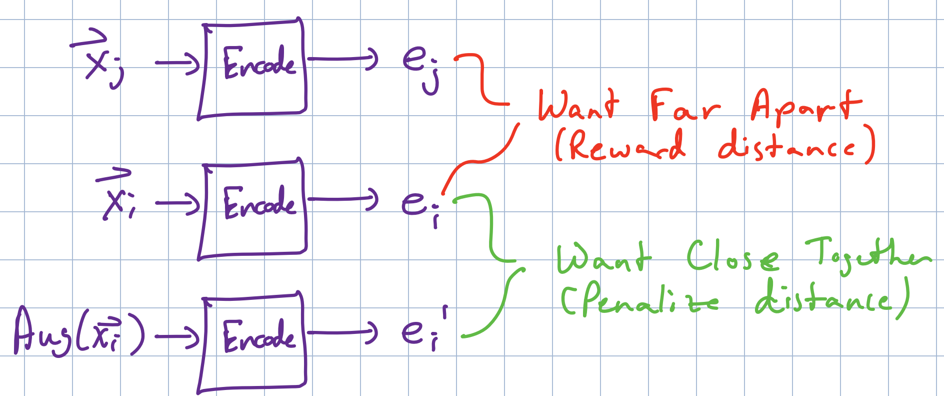
2) Clustering ⟺ ???

Contrastive Self-supervision

Core Idea: An example and its own augmentations should be in the same cluster, While fundamentally different examples are in different clusters.

(First formalized in "Provable Guarantees for Self-Supervised Deep Learning with Spectral Contrastive Loss" by HaoChen, et. al. in 2021 but builds on latent intuition in "A theoretical analysis of Contrastive Unsupervised Representation Learning" by Arora, et. al. in 2019)

Most basic structure: Embed/Encode 3 things



Fun going beyond.... It's possible to avoid negative examples entirely, but this requires other tricks to prevent collapse...