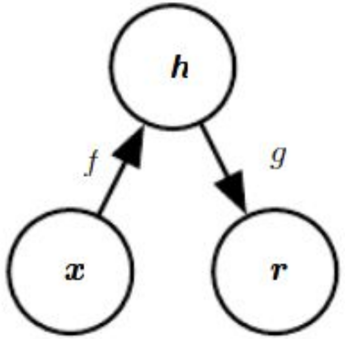
# Autoencoders

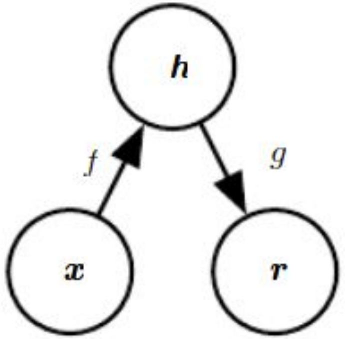
# What is an autoencoder?

- Here's one view:
  - In comes an input example **x**
  - Then the *encoder f* is applied
  - The output **h** of the encoder is called a *code* or a *latent representation/code*
  - The *decoder g* tries to reproduce **x** using **h**



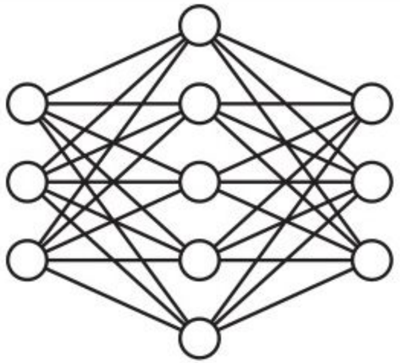
# What is an autoencoder?

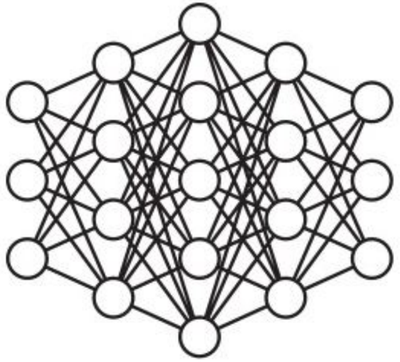
- More formally:
  $\mathbf{r} = g(\mathbf{h}) = g(f(\mathbf{x}))$
- If we're doing a good job, hopefully we'll have:
  $\mathbf{r} \approx \mathbf{x}$
- In general, we get the loss for AEs:
  $L(\mathbf{x}, g(f(\mathbf{x})) = L(\mathbf{x}, \mathbf{r})$
- Predicting black-white pixels (MNIST):
  $L(x, r) = -\frac{1}{m} \sum_{i=1}^{m} x \log r$ **Sigmoidal outputs**
- Predicting real-valued pixels (general images):
  $L(x, r) = \frac{1}{2m} \sum_{i=1}^{m} (x - r)^2$ **Linear outputs**



## What is an autoencoder?

- *Overcomplete* version
- More components in the latent code than in the input code
- Careful with this one - potentially difficult to train
  - Identity mapping: $g(f(\mathbf{x}))$ just becomes the identity function
- Solution:
  - Regularization
  - Weird tricks (later)

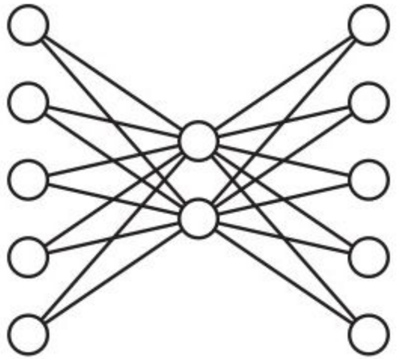

(b) Shallow overcomplete

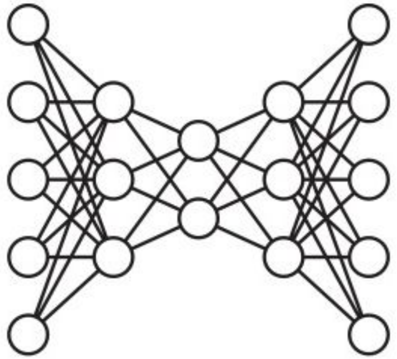

(d) Deep overcomplete

https://www.sciencedirect.com/science/article/pii/S1566253517307844

# What is an autoencoder?

- *Undercomplete* version
- In general much more interesting:
  - A compressor of data
  - A non-linear PCA machine
  - An algorithm *forced* to learn what’s “interesting” in our data

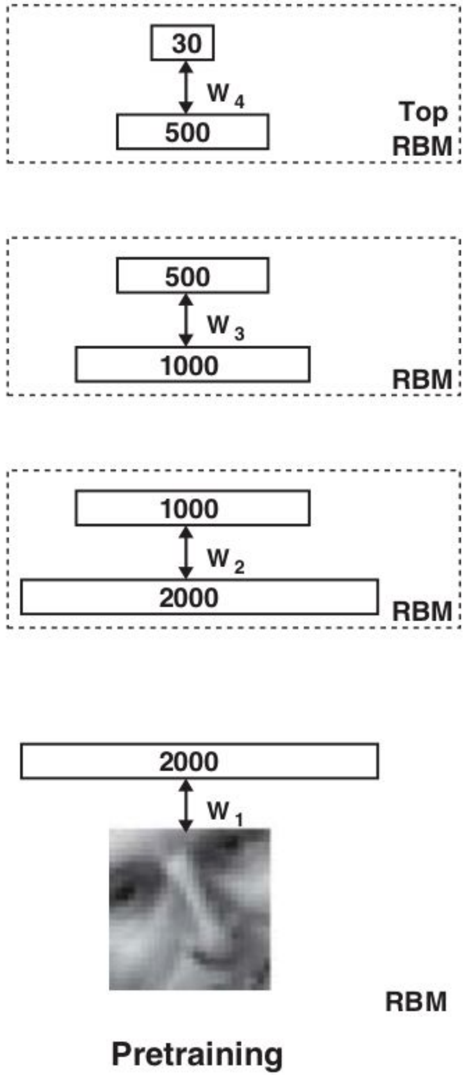

(a) Shallow undercomplete


(c) Deep undercomplete

https://www.sciencedirect.com/science/article/pii/S1566253517307844

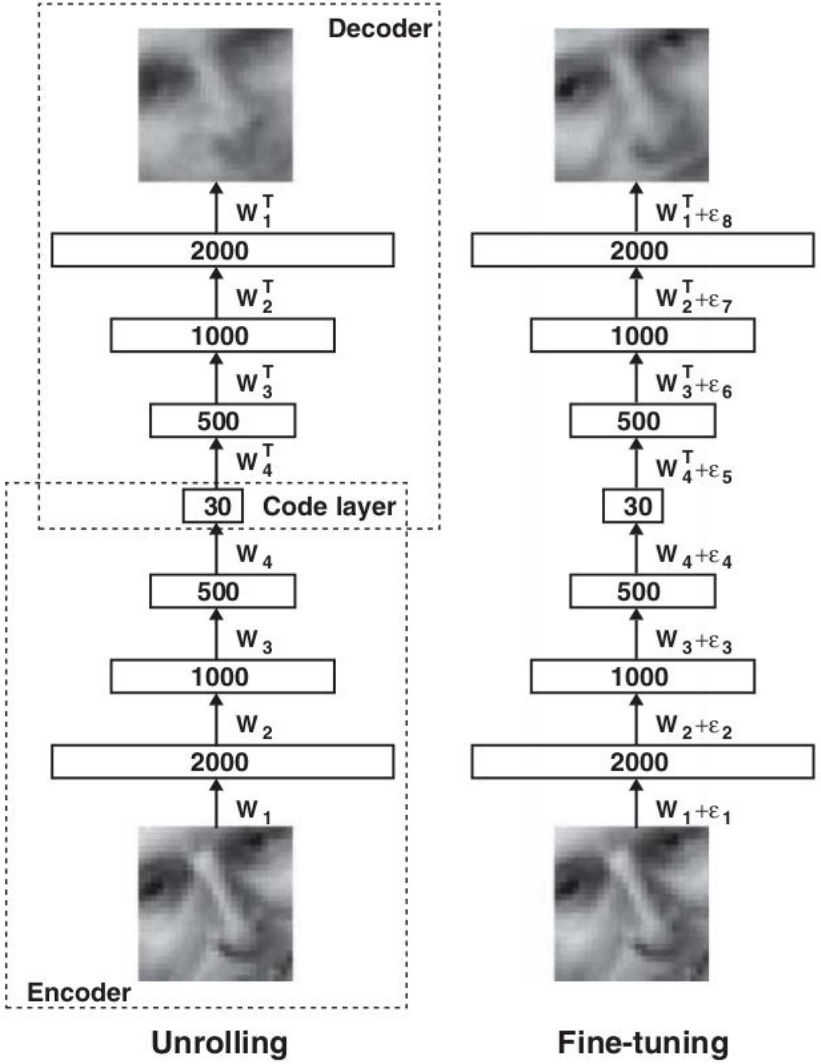
# The rise of AEs

- Before ReLU, deep nets were seen as untrainable
- CNNs were working with some depth,
  but also quite few feature maps
- Hinton & Salakhutdinov came up with a solution in 2006
- First step:
  - Greedy layer-wise pretraining
  - Each double-arrow means *f* going upwards and *g* going downwards
  - RBMs are just a special type of autoencoders



# The rise of AEs

- Upon completion of pretraing, the weights are “unrolled” and the whole net is jointly trained further (fine-tuning)
  - This made deep nets (here 8 layers) actually *work* with sigmoids
- Later, Hinton himself used ReLUs, requiring no pretraining
- The method of pretrained nets remained popular in another form: CNN backbones



# Codes learned by AEs

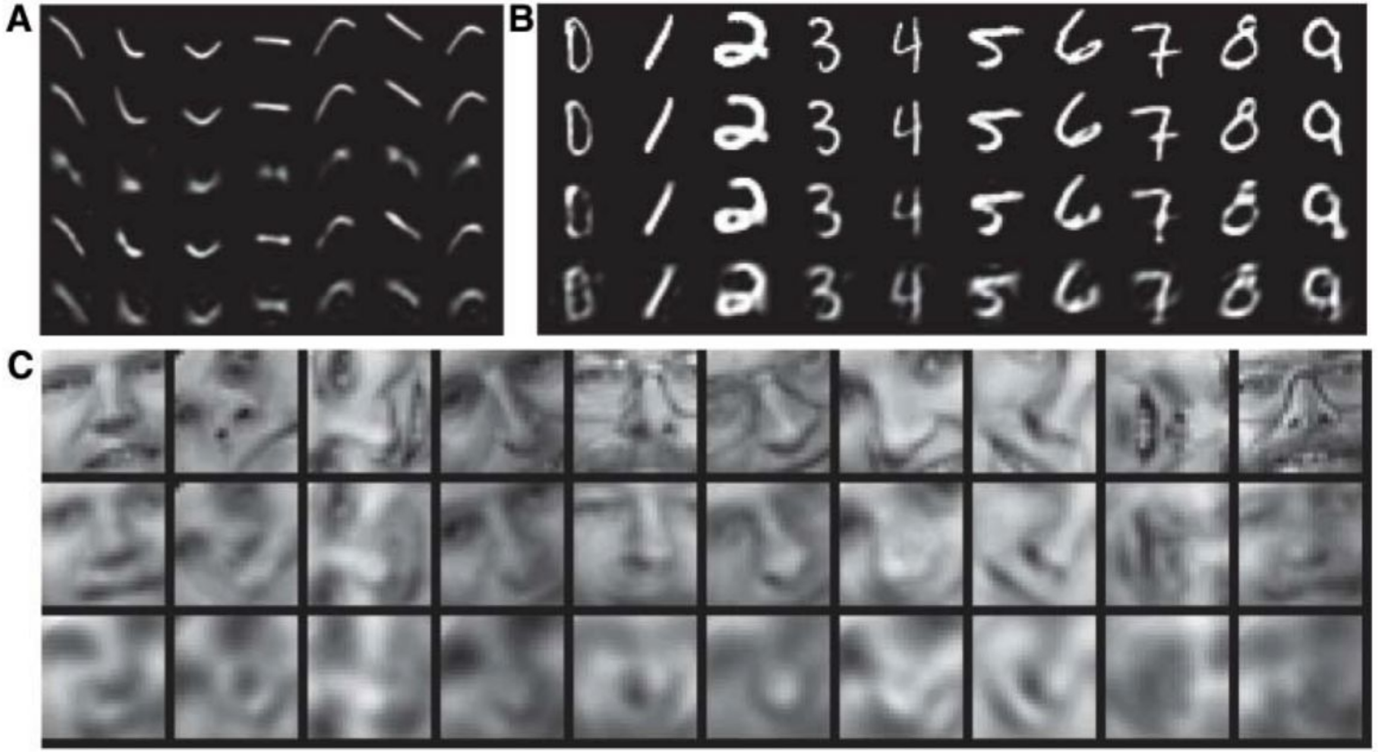
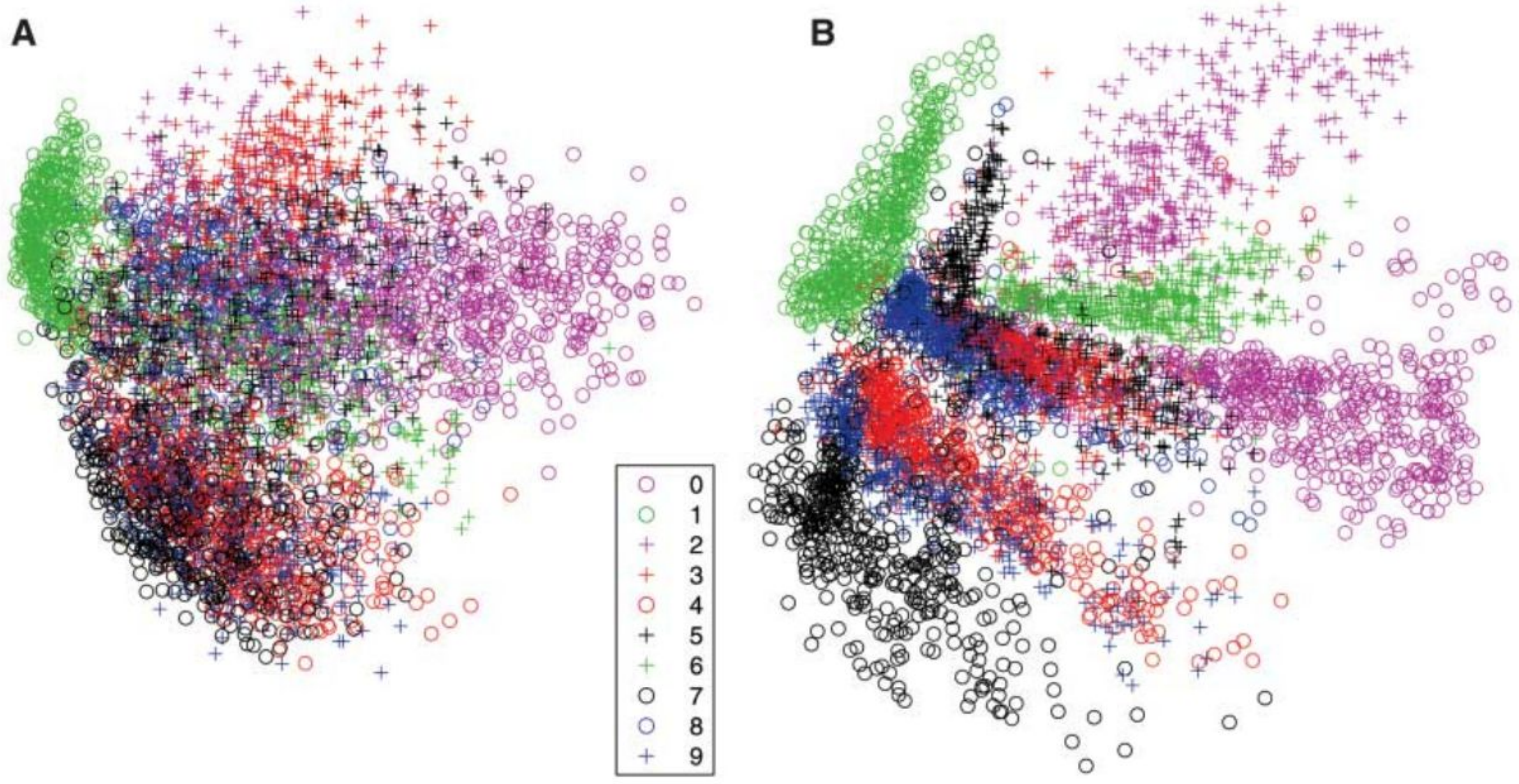


**Fig. 2.** (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error per image for the last four rows is 1.44, 7.64, 2.45, 5.90. (**B**) Top to bottom: A random test image from each class; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional logistic PCA and standard PCA. The average squared errors for the last three rows are 3.00, 8.01, and 13.87. (**C**) Top to bottom: Random samples from the test data set; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional PCA. The average squared errors are 126 and 135.

# 2D codes learned by an AE on MNIST

**Fig. 3.** (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



# Another use of AEs

- Doumanoglou et al., CVPR’16 
  ([https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Doumanoglou_Recovering_6D_Object_CVPR_2016_paper.pdf](https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Doumanoglou_Recovering_6D_Object_CVPR_2016_paper.pdf))

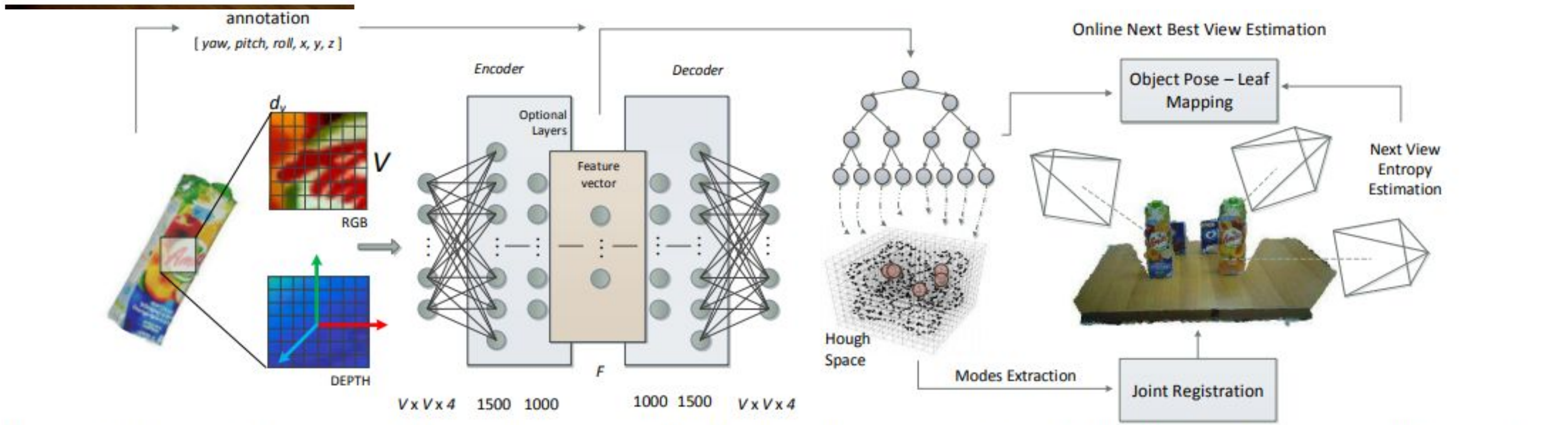


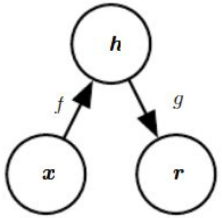
Figure 2: Framework Overview. After patch extraction, RGBD channels are given as input to the Sparse Autoencoder. The annotation along with the produced features of the middle layer are given to a Hough Forest, and the final hypotheses are generated as the modes of the Hough voting space. After refining the hypotheses using joint registration, we estimate the next-best-view using a pose-to-lead mapping learnt from the trained Hough Forest.

## SAEs

- The section on sparse AEs in the book presents an alternative view on MAP inference, when used in this context
- What it boils down to is a prior on the hiddens **h** instead of the normal prior on our weights **W**

## SAEs

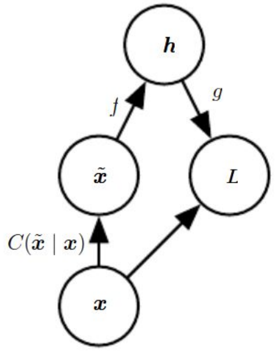
- Sparse AEs introduce a prior on the hiddens, formalized as:
  $L(x, r) + \Omega(\mathbf{h})$
- The most common use is *L*$_1$:
  $L(x, r) + \lambda \sum_i |h_i|$



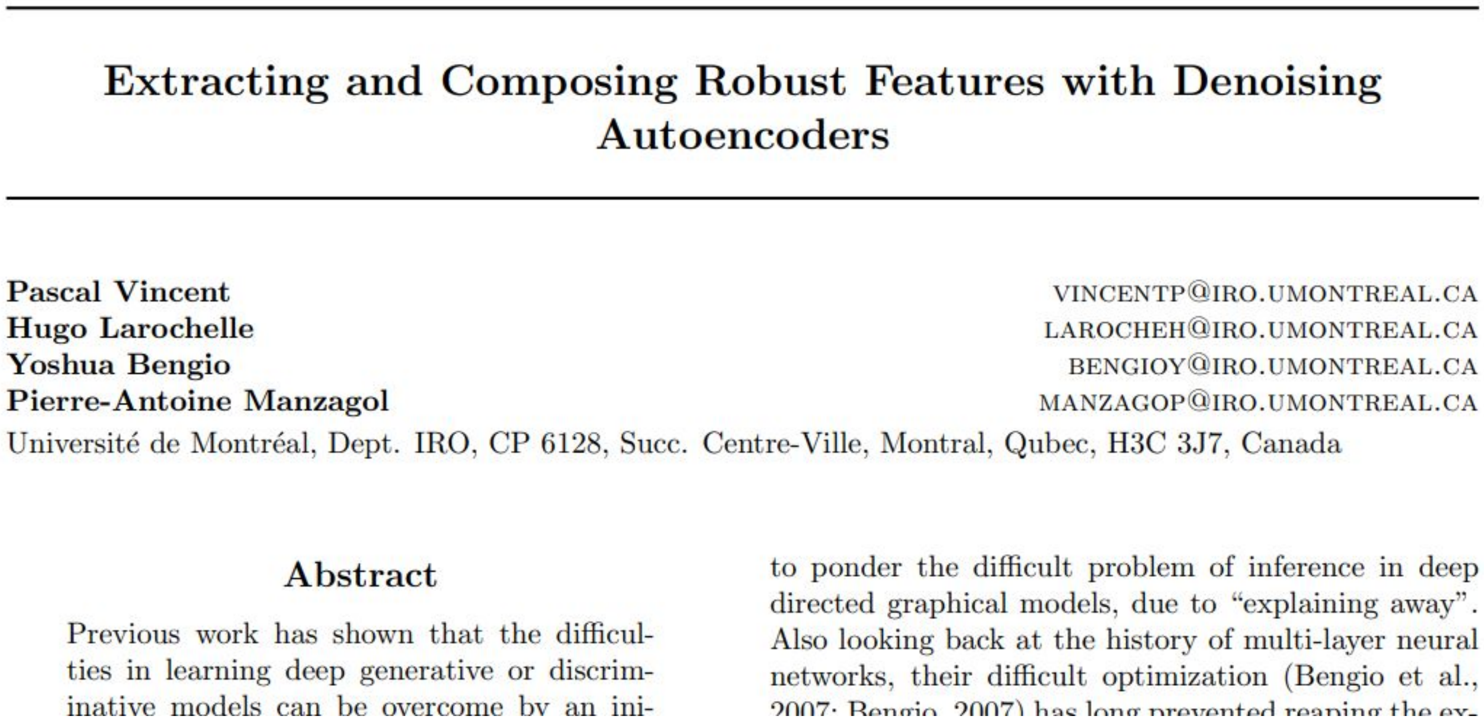
$$\begin{tikzcd} & \boldsymbol{h} \arrow[dr, "g"] & \\ \boldsymbol{x} \arrow[ur, "f"] & & \boldsymbol{r} \end{tikzcd}$$

# DAEs

- Denoising AEs are trained by corrupting the input
  - Denote the corrupted input $\tilde{\mathbf{x}}$
- The output is kept “clean”
- The loss looks like this:
  $L(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$



# DAEs



## Extracting and Composing Robust Features with Denoising Autoencoders

**Pascal Vincent** VINCENTP@IRO.UMONTREAL.CA
**Hugo Larochelle** LAROCHEH@IRO.UMONTREAL.CA
**Yoshua Bengio** BENGIOY@IRO.UMONTREAL.CA
**Pierre-Antoine Manzagol** MANZAGOP@IRO.UMONTREAL.CA
Université de Montréal, Dept. IRO, CP 6128, Succ. Centre-Ville, Montral, Qubec, H3C 3J7, Canada

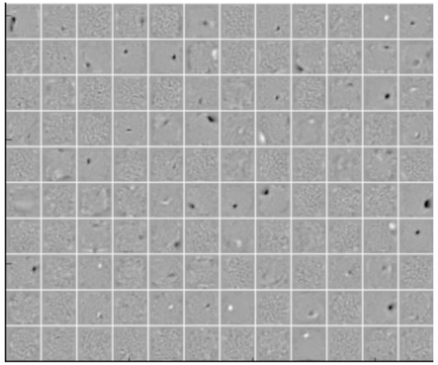
### Abstract

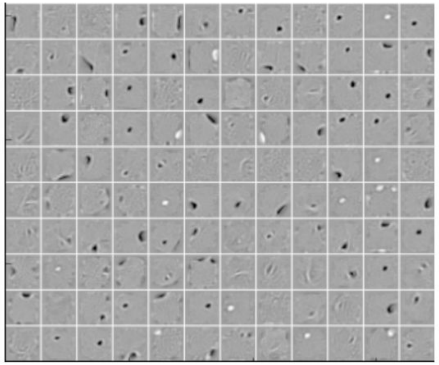
Previous work has shown that the difficulties in learning deep generative or discriminative models can be overcome by an ini-

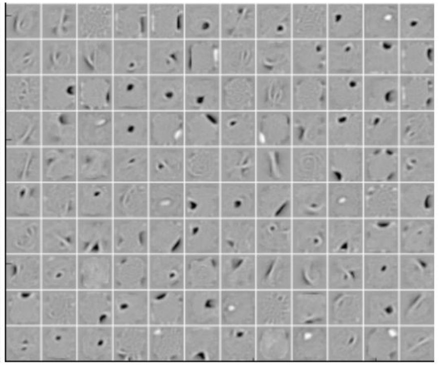
to ponder the difficult problem of inference in deep directed graphical models, due to "explaining away". Also looking back at the history of multi-layer neural networks, their difficult optimization (Bengio et al., 2007; Bengio, 2007) has long prevented reaping the ex-

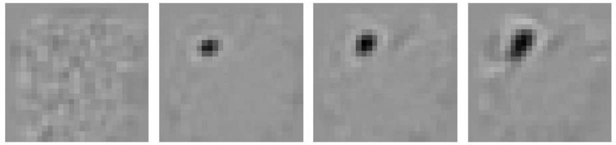
## DAEs

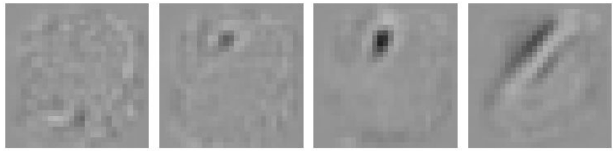
- Noise regularizes the net and produces more smooth filters


(a) No destroyed inputs


(b) 25% destruction


(c) 50% destruction


(d) Neuron A (0%, 10%, 20%, 50% destruction)


(e) Neuron B (0%, 10%, 20%, 50% destruction)

## DAEs

- DAEs are better at learning the structure or the *manifold* of $p_{\text{data}}$

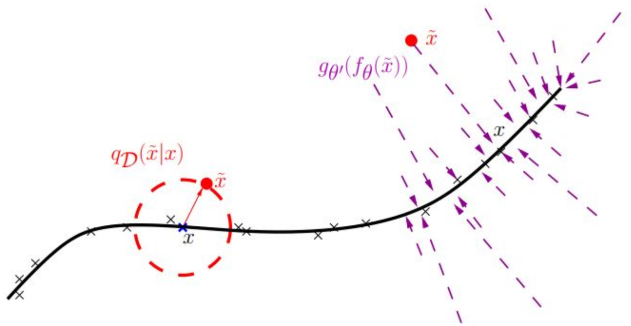

$g_{\theta'}(f_\theta(\tilde{x}))$
$\tilde{x}$
$q_{\mathcal{D}}(\tilde{x}|x)$
$\tilde{x}$
$x$
$x$

*Figure 2.* **Manifold learning perspective.** Suppose training data ($\times$) concentrate near a low-dimensional manifold. Corrupted examples (●) obtained by applying corruption process $q_{\mathcal{D}}(\widetilde{X}|X)$ will lie farther from the manifold. The model learns with $p(X|\widetilde{X})$ to "project them back" onto the manifold. Intermediate representation $Y$ can be interpreted as a coordinate system for points on the manifold.

## DAEs

- Let’s look at an example: https://colab.research.google.com/drive/1jXQaleSqksOrWhDPtze6YnieuuAc_mAe?usp=sharing

## CAEs

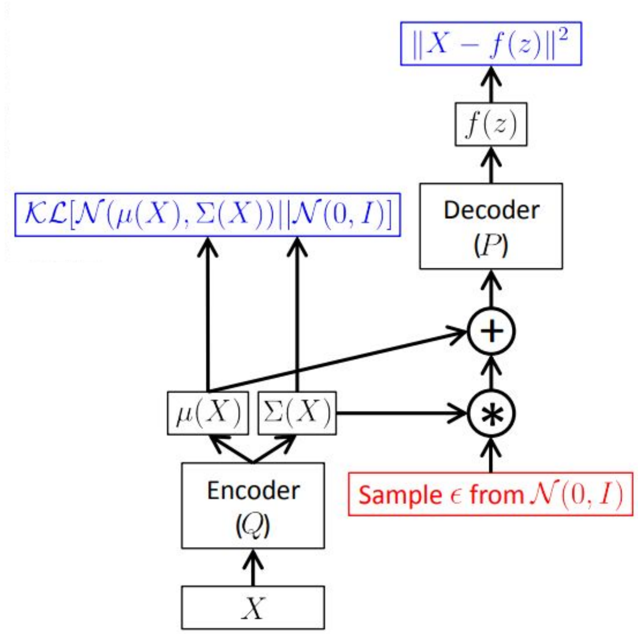
- Contractive AEs try to achieve insensitivity to input perturbations directly on the hidden code
    - Remember that DAEs tried to achieve this on the reconstruction directly
- The term is now defined as a penalty on the "speed" of change of **h**:

$$\Omega(\mathbf{h}) = \lambda \left\| \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right\|_F^2 = \lambda \left\| \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2$$

- This has the effect of "contracting" nearby input examples to nearby hidden codes

## VAEs

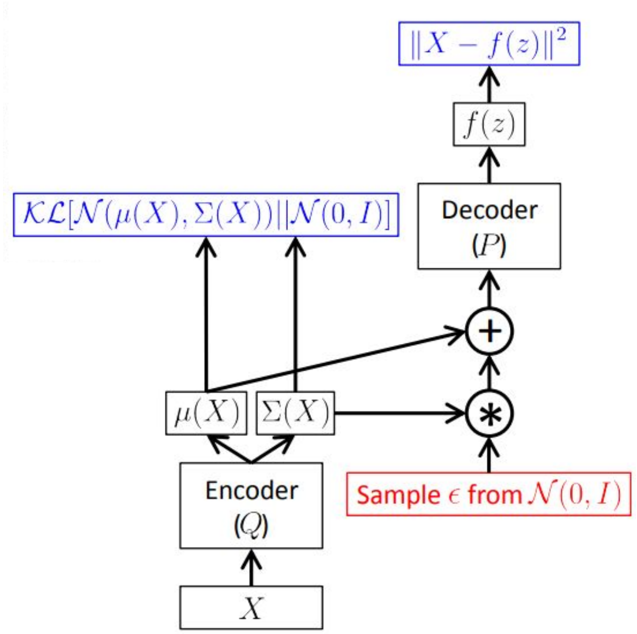
- Variational AEs are trained to produce a latent mean and variance


$\|X - f(z)\|^2$
$f(z)$
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, I)]$
Decoder ($P$)
$+$
$*$
$\mu(X)$
$\Sigma(X)$
Encoder ($Q$)
Sample $\epsilon$ from $\mathcal{N}(0, I)$
$X$

Doersch, 2016

# VAEs

- VAEs then use a statistical *divergence* to make the latent code Gaussian

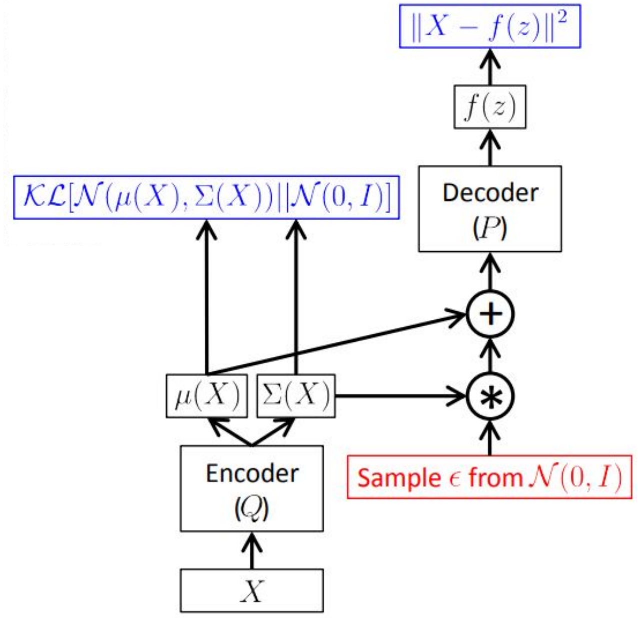


$\|X - f(z)\|^2$
$f(z)$
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) || \mathcal{N}(0, I)]$
Decoder ($P$)
$+$
$\mu(X)$
$\Sigma(X)$
$*$
Encoder ($Q$)
Sample $\epsilon$ from $\mathcal{N}(0, I)$
$X$

Doersch, 2016

## VAEs

- When properly trained:
  - The reconstruction is good at reproducing the input
  - The latent mean/variance are Gaussian
- **This makes sampling easy!**

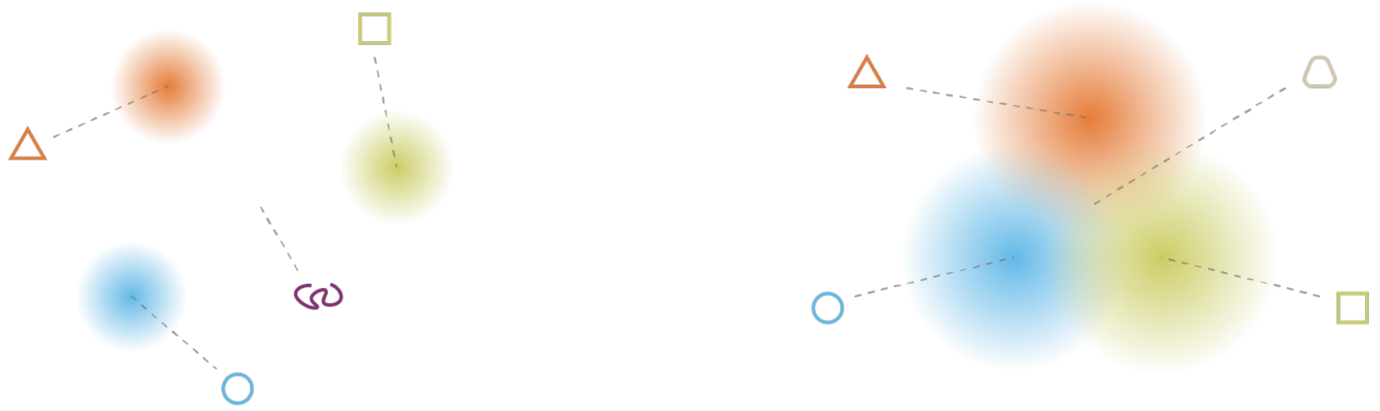


$\|X - f(z)\|^2$
$f(z)$
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) || \mathcal{N}(0, I)]$
Decoder (P)
+
$\mu(X)$
$\Sigma(X)$
*
Encoder (Q)
Sample $\epsilon$ from $\mathcal{N}(0, I)$
$X$

Doersch, 2016

# VAEs

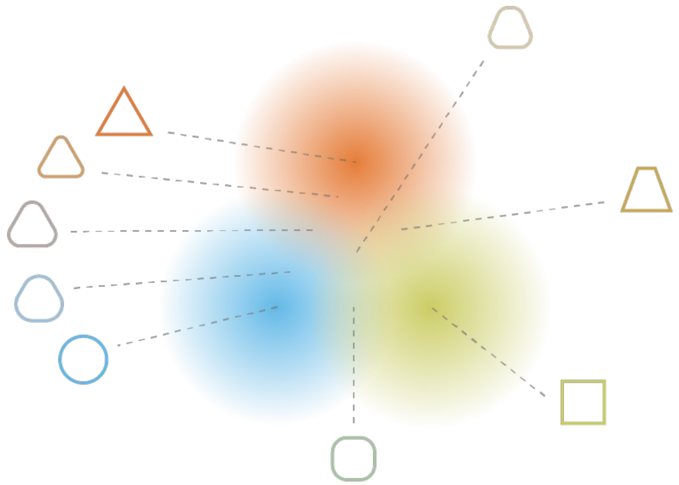
- Here’s the principle of the added KL-based regularization:



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

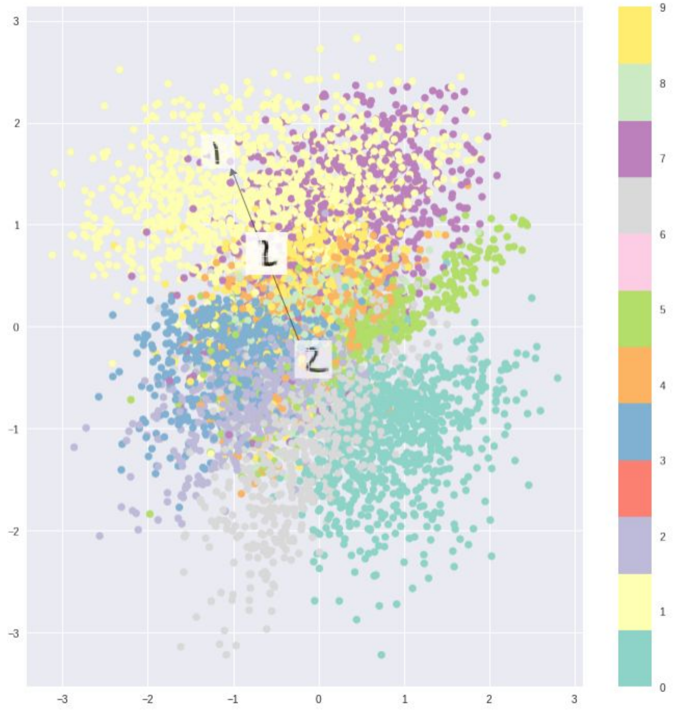
## VAEs

- This allows for easy sampling and interpolation:

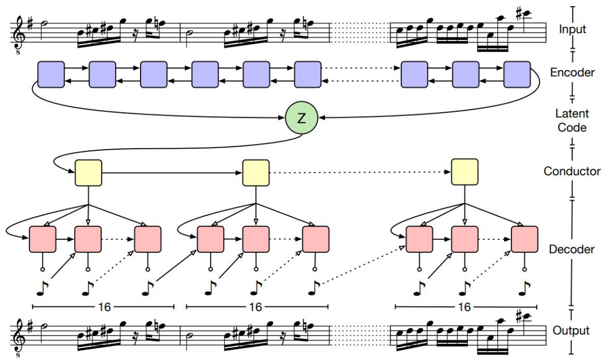


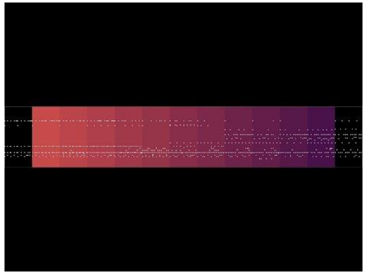
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

# VAEs



Roberts et al., 2016

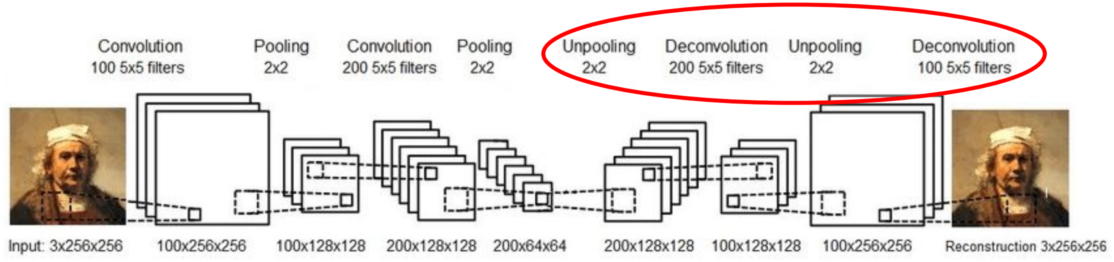




https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

## Convolutional AEs

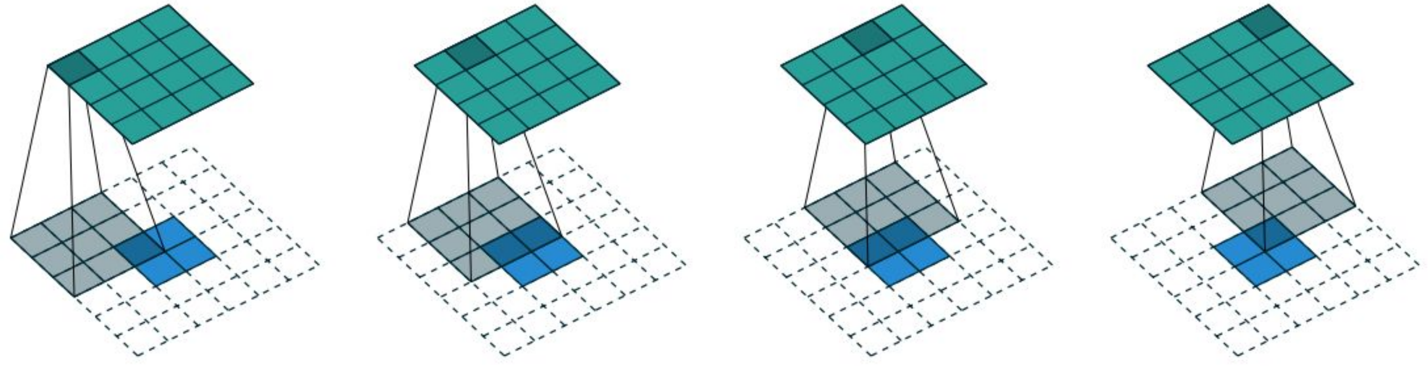
- Finally, you can of course use CNNs for doing better autoencoding of images
- The key ingredient is *transposed convolution*
  - Early literature misused the term *deconvolution* for the same principle



https://www.researchgate.net/figure/Illustration-of-convolutional-autoencoder-In-this-example-the-CAE-comprises-two_fig3_306081538

## Transposed convolution

- Available in pytorch:
  https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html
- An excellent guide on this topic is here:
  https://arxiv.org/abs/1603.07285v1



# Challenge

Anders created a 50k/10k train/test split of the FaceScrub dataset. See next slide for download instructions. (full dataset here: http://vintage.winklerbros.net/facescrub.html)

- Denoising AutoEncoder (DAE) on FaceScrub:
    - Implement a convolutional AE to reconstruct face images
    - Now do the same but corrupt the images and teach the model to reconstruct the original
    - (Might be easier to train an over complete DAE)
- Bonus: Train a Variational AE (VAE) with a 2D latent space on MNIST
    - Plot the latent embeddings of the mnist images and compare with Figure 3B from the science paper: https://www.cs.toronto.edu/~hinton/science.pdf
    - Plot the corresponding grid of reconstructions

## FaceScrub

Drive: https://drive.google.com/file/d/1Uzgc9c0MSYP4y9ia7mvwCGggjhTDEmrk/view?usp=sharing

Download through python:

```
import numpy as np
from google_drive_downloader import GoogleDriveDownloader as gdd

gdd.download_file_from_google_drive('1Uzgc9c0MSYP4y9ia7mvwCGggjhTDEmrk', '~/img_align_celeba_50k.npz')
blob = np.load('~/img_align_celeba_50k.npz')
x_train, x_test = blob['x_train'], blob['x_test']
```