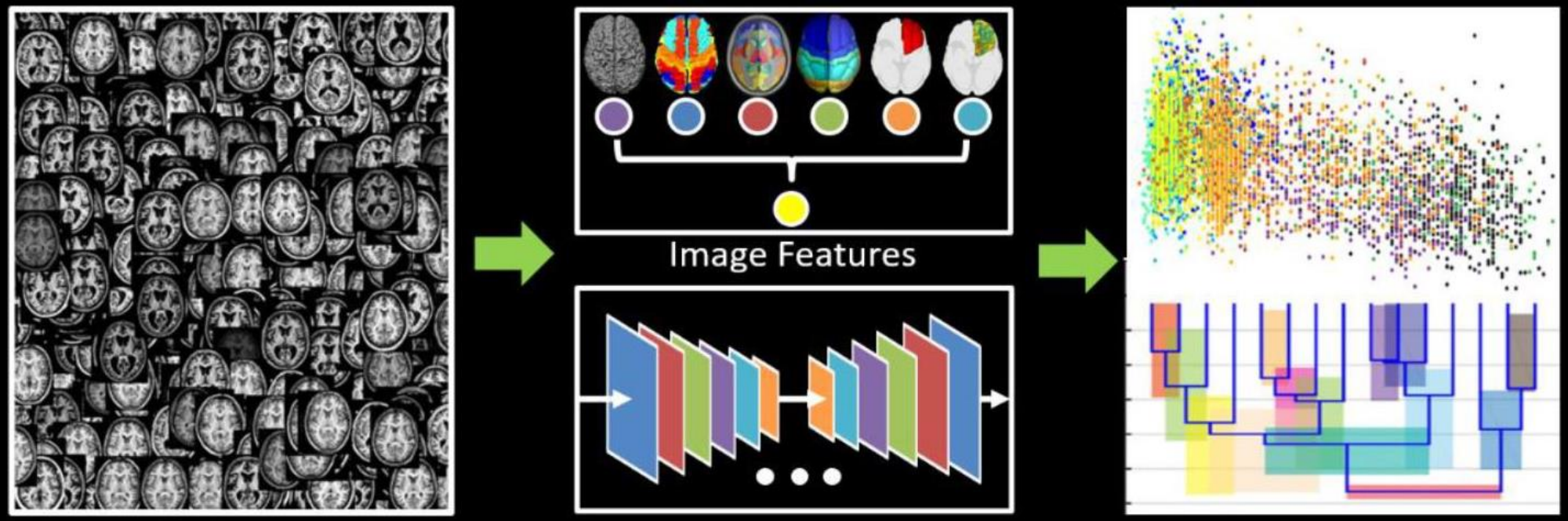


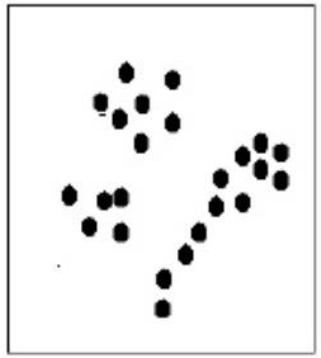
Unsupervised Learning

[Spring 2020 CS-8395 Deep Learning in Medical Image Computing]

Instructor: Yuankai Huo, Ph.D.
Department of Electrical Engineering and Computer Science
Vanderbilt University

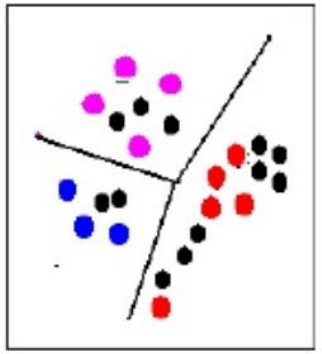
Learn from big unlabeled images

Unsupervised Learning



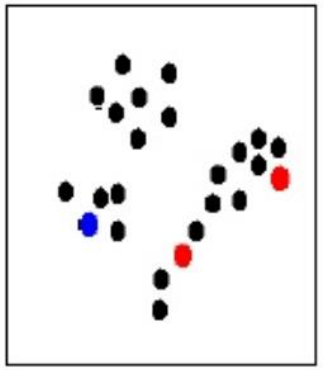
We know nothing about data sructure

Supervised Learning



We know well data sructure

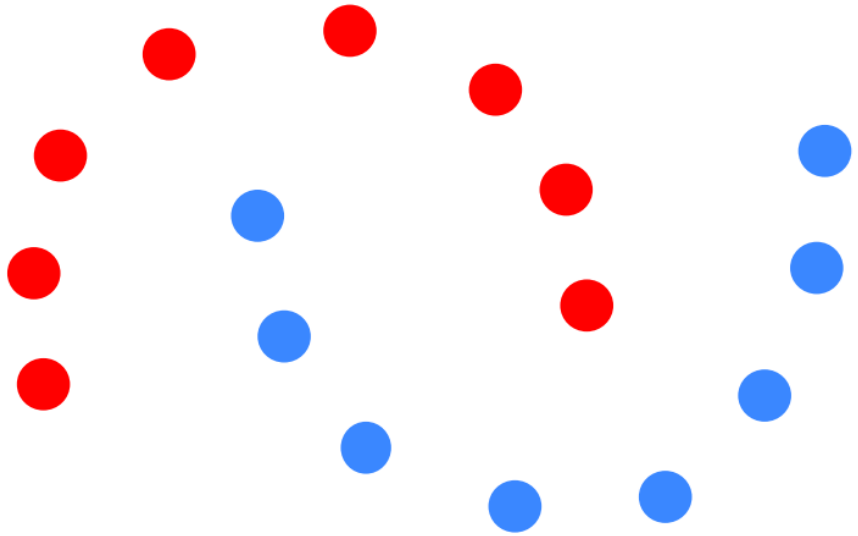
Semi-Supervised Learning



We know something about data sructure

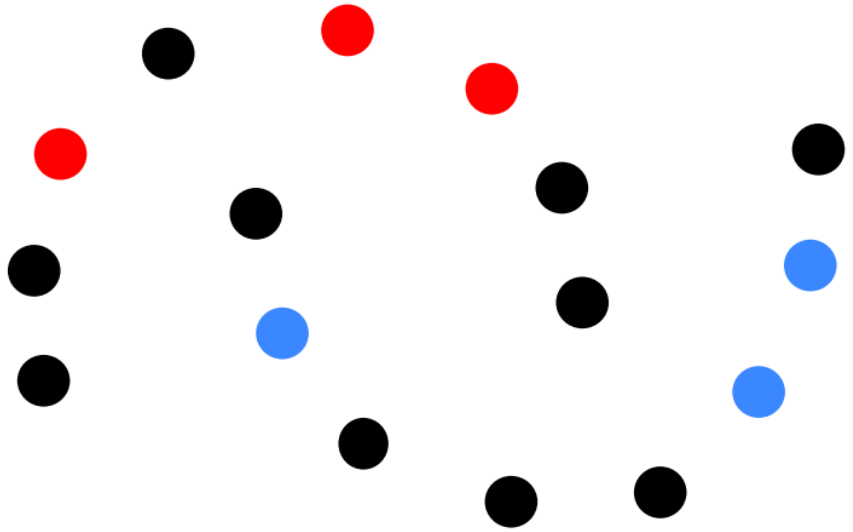
https://www.slideshare.net/Tatiana.lando/clustering-presentation

Two Moons Dataset



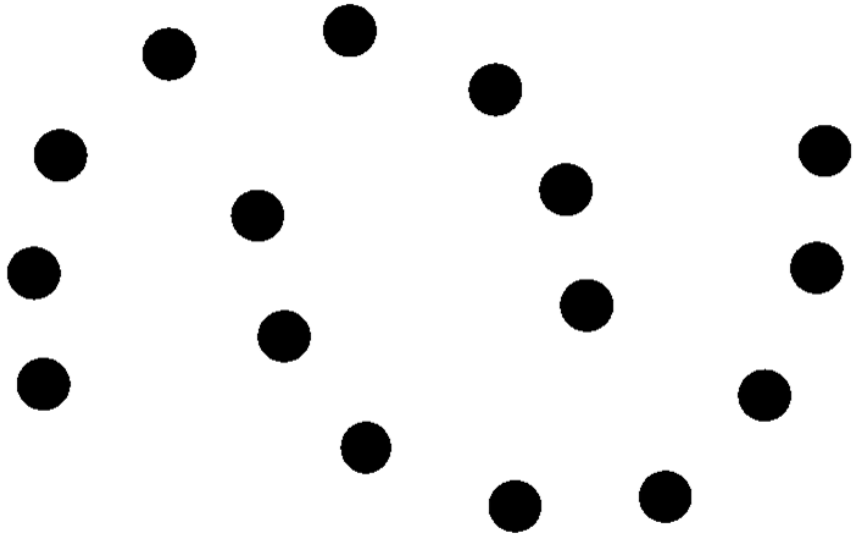
https://web.stanford.edu/class/cs224n/lectures/lecture17.pdf

Don’t have all labels



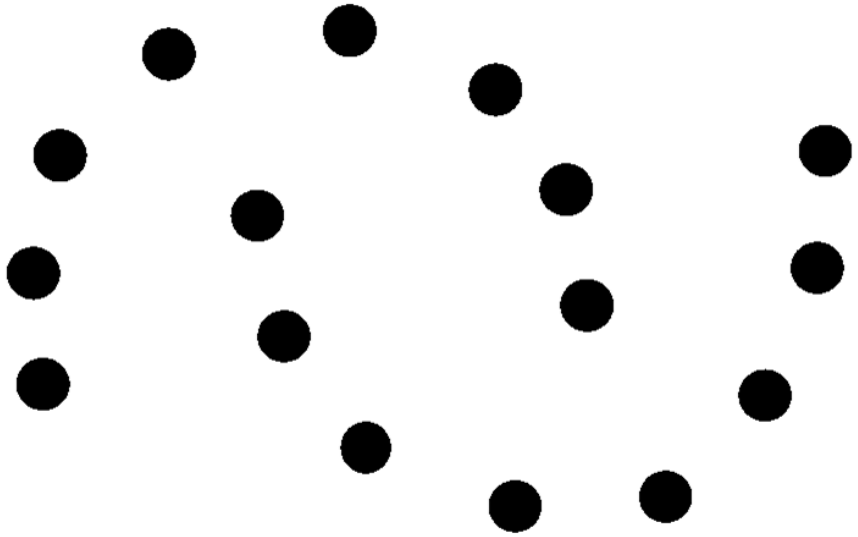
https://web.stanford.edu/class/cs224n/lectures/lecture17.pdf

Don’t have any labels



https://web.stanford.edu/class/cs224n/lectures/lecture17.pdf

Definition



“Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. The most common **unsupervised learning** method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.”

https://www.mathworks.com/discovery/unsupervised-learning.html

Two Categories

Unsupervised learning

Clustering

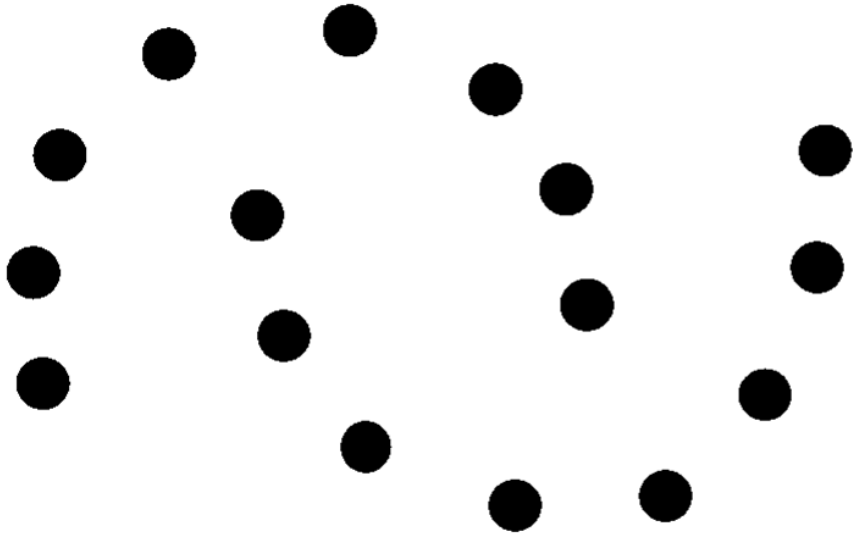
Two Categories

Unsupervised learning

Clustering

k-Means clustering
Gaussian mixture models
Hierarchical clustering
Mean shift
…

What else



https://www.mathworks.com/discovery/unsupervised-learning.html

Two Categories

Clustering

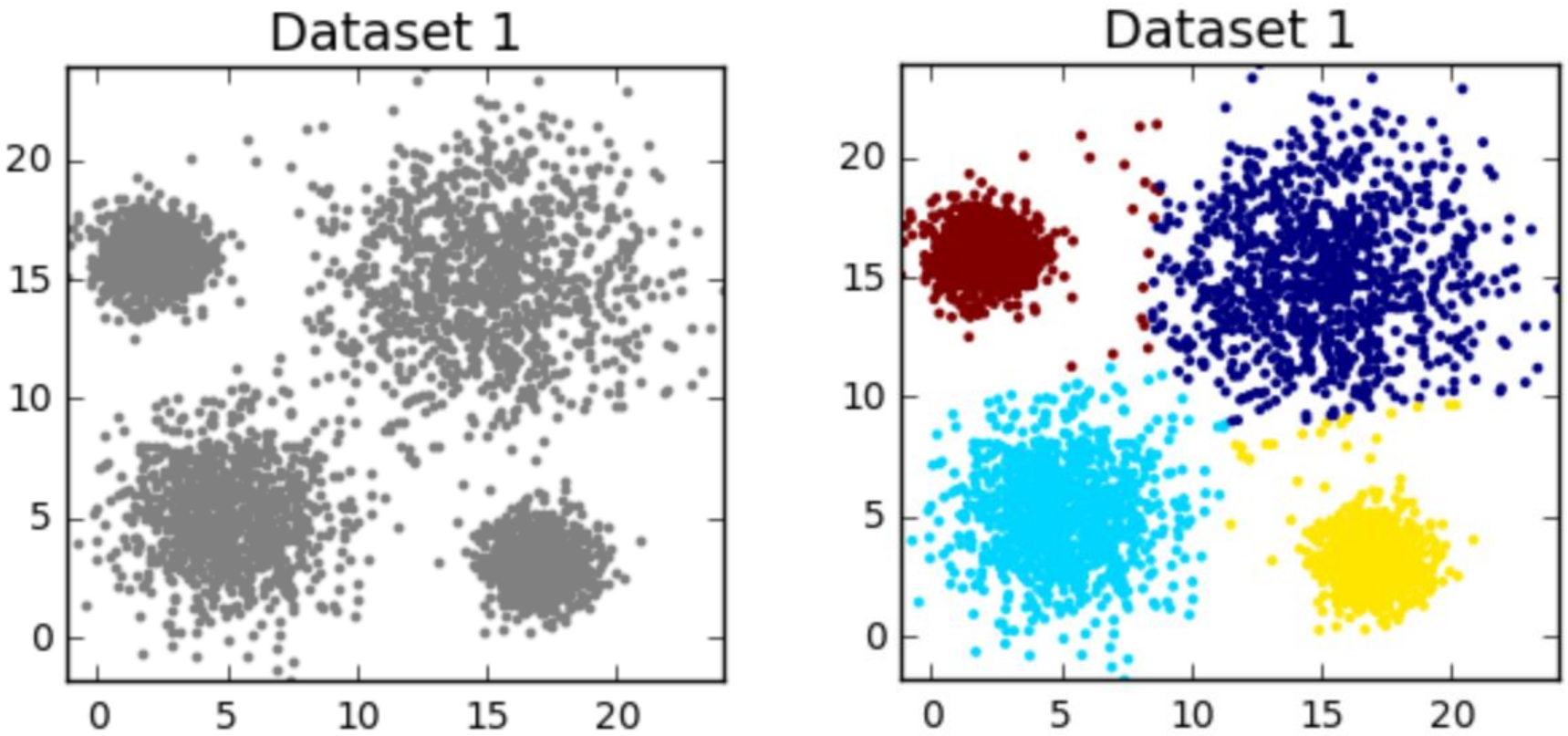
k-Means clustering
Gaussian mixture models
Hierarchical clustering
Mean shift
…

Feature Extraction (Dimension Reduction)

PCA
ICA
Sparse Coding
Manifold Learning
…

Clustering

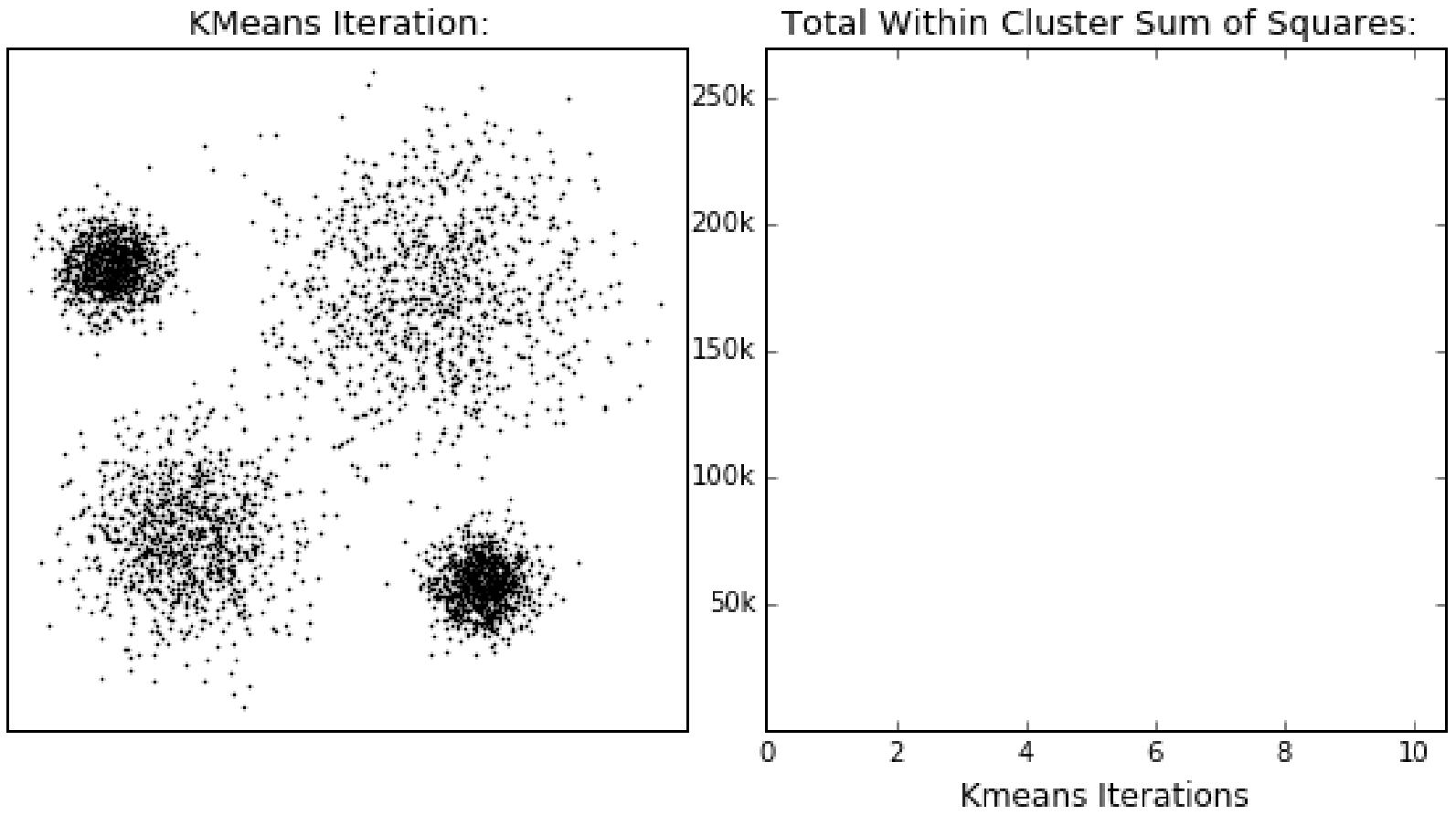
K-means



https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Clustering

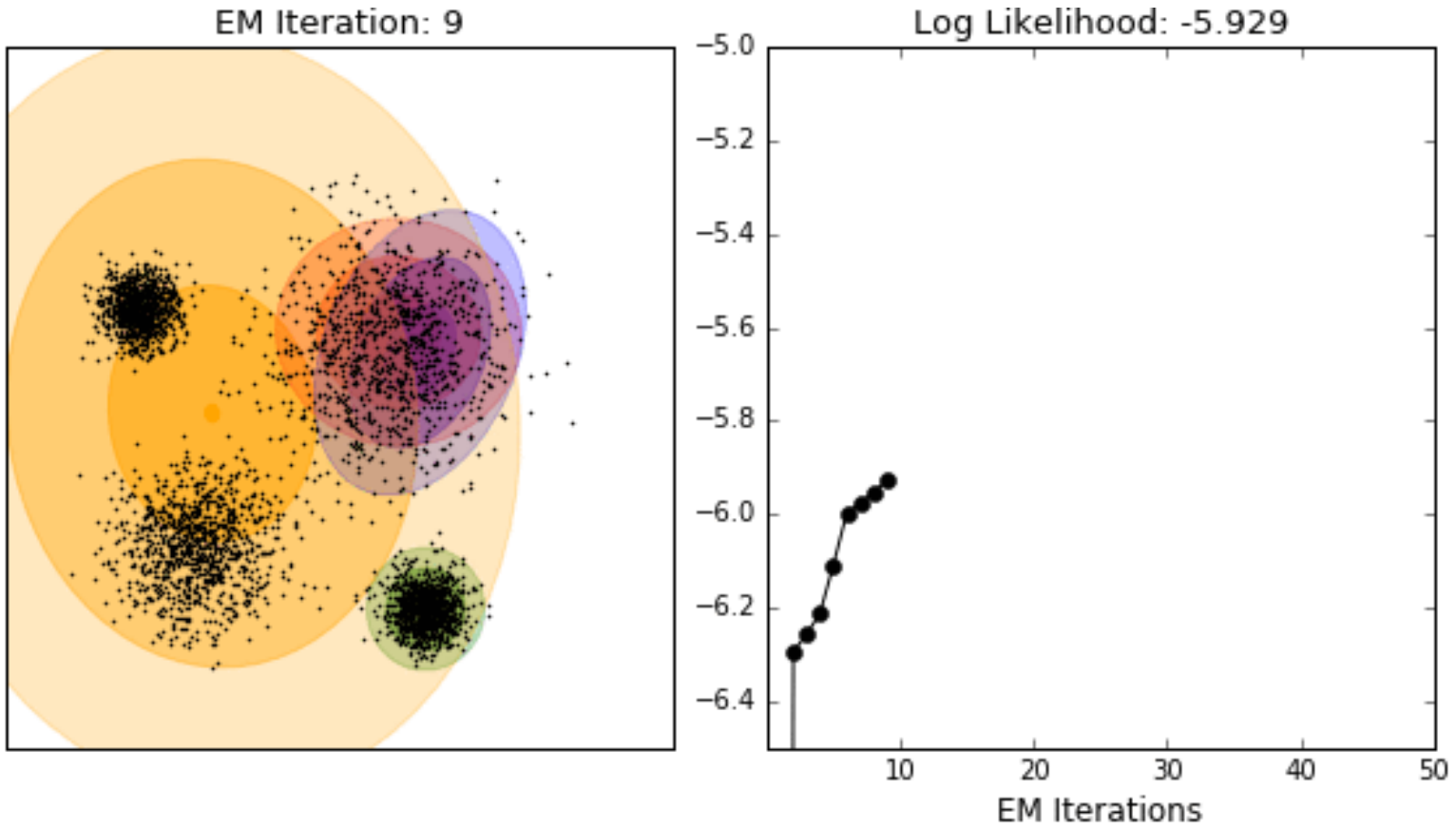
K-means



https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Clustering

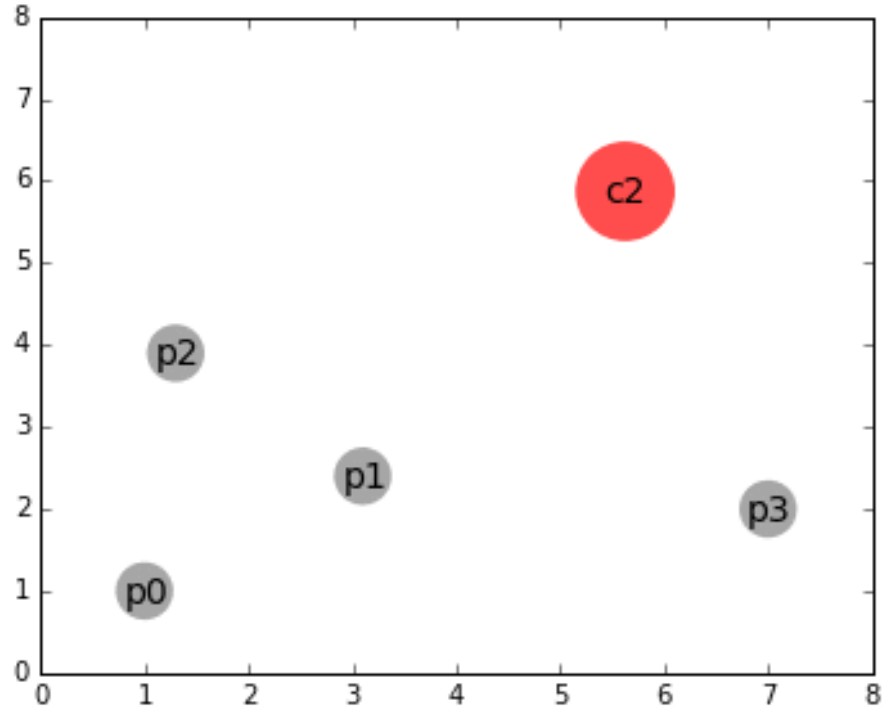
Gaussian Mixture Model (GMM)

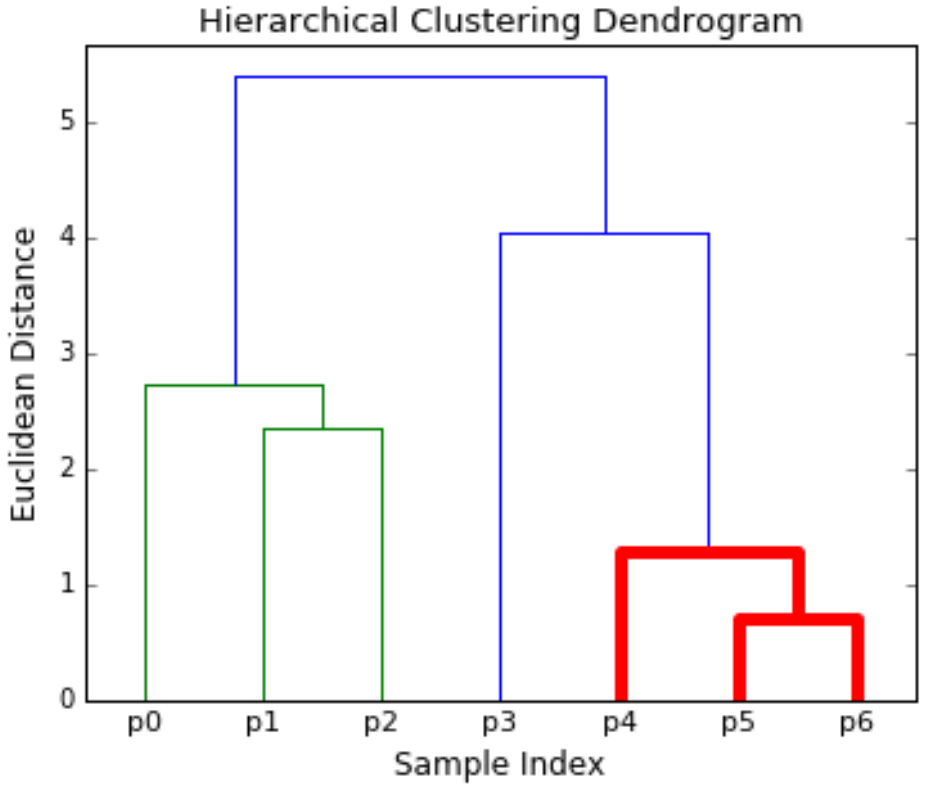


https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Clustering

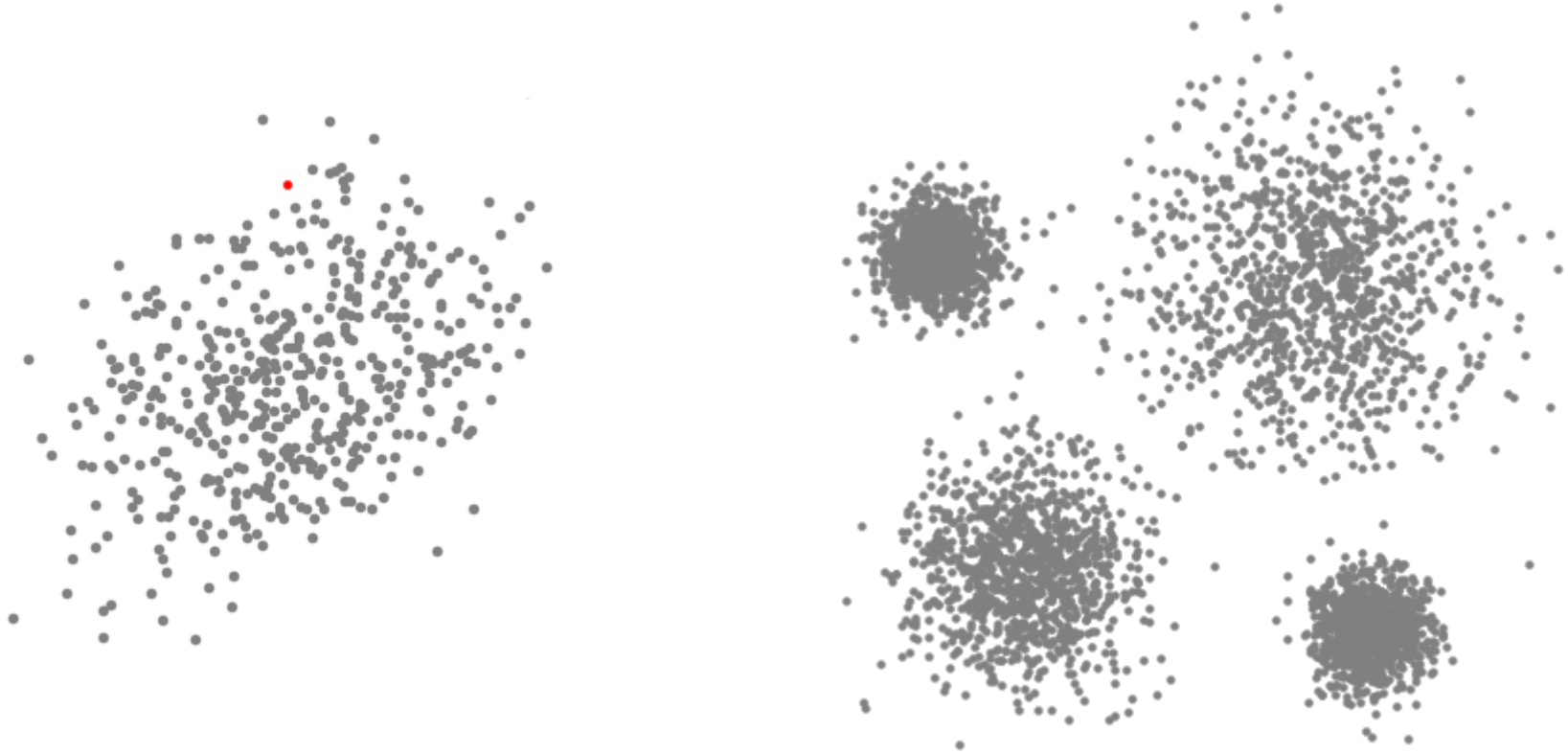
Hierarchical Clustering





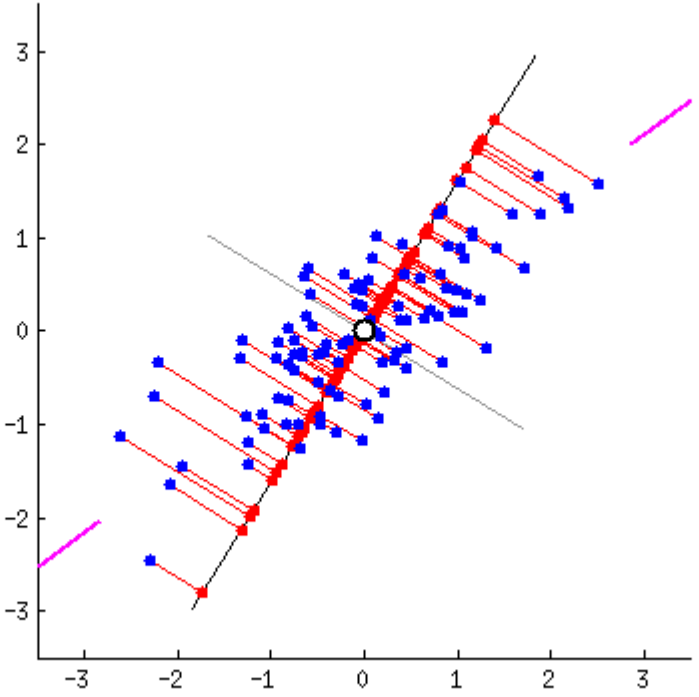
Clustering

Mean-shift



Feature Extraction

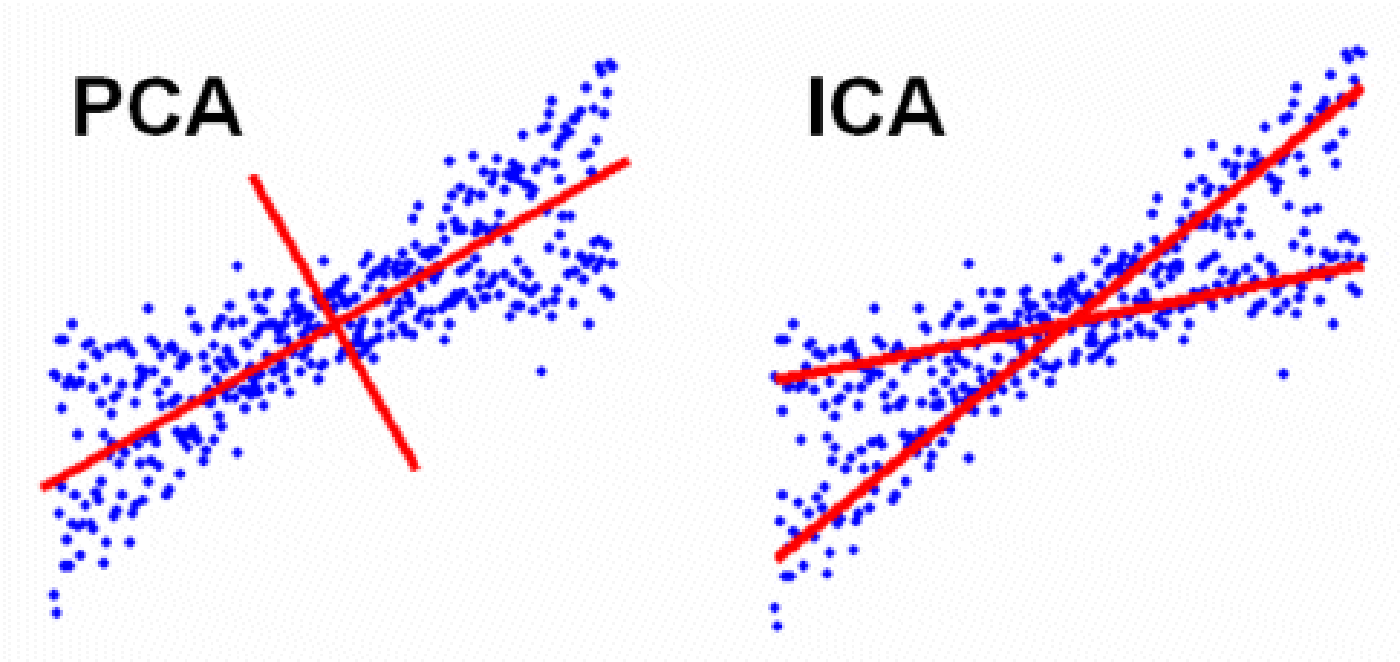
PCA:
Principle Component Analysis



https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Feature Extraction

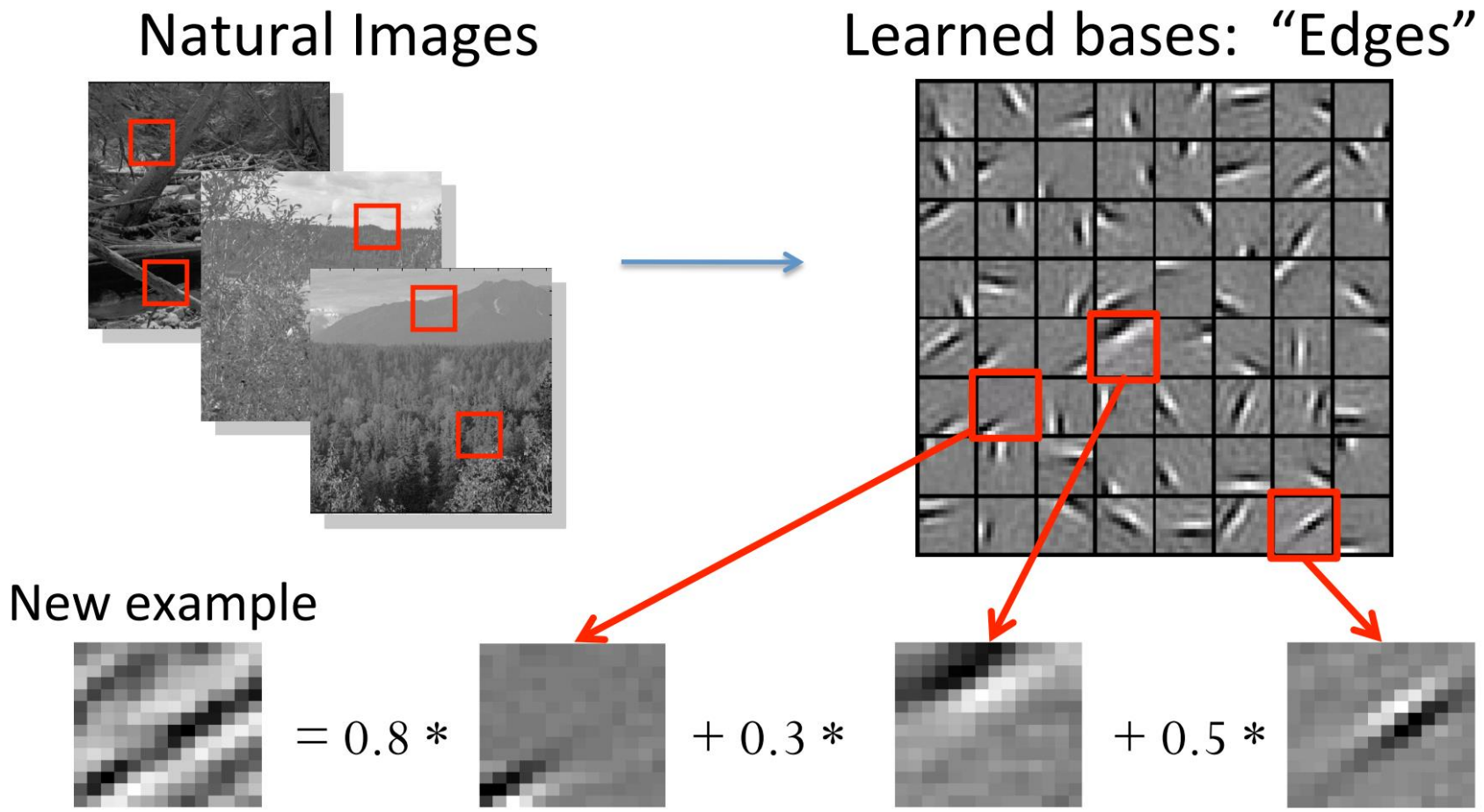
ICA:
Independent Component Analysis



http://meg.aalip.jp/ICA/

Feature Extraction

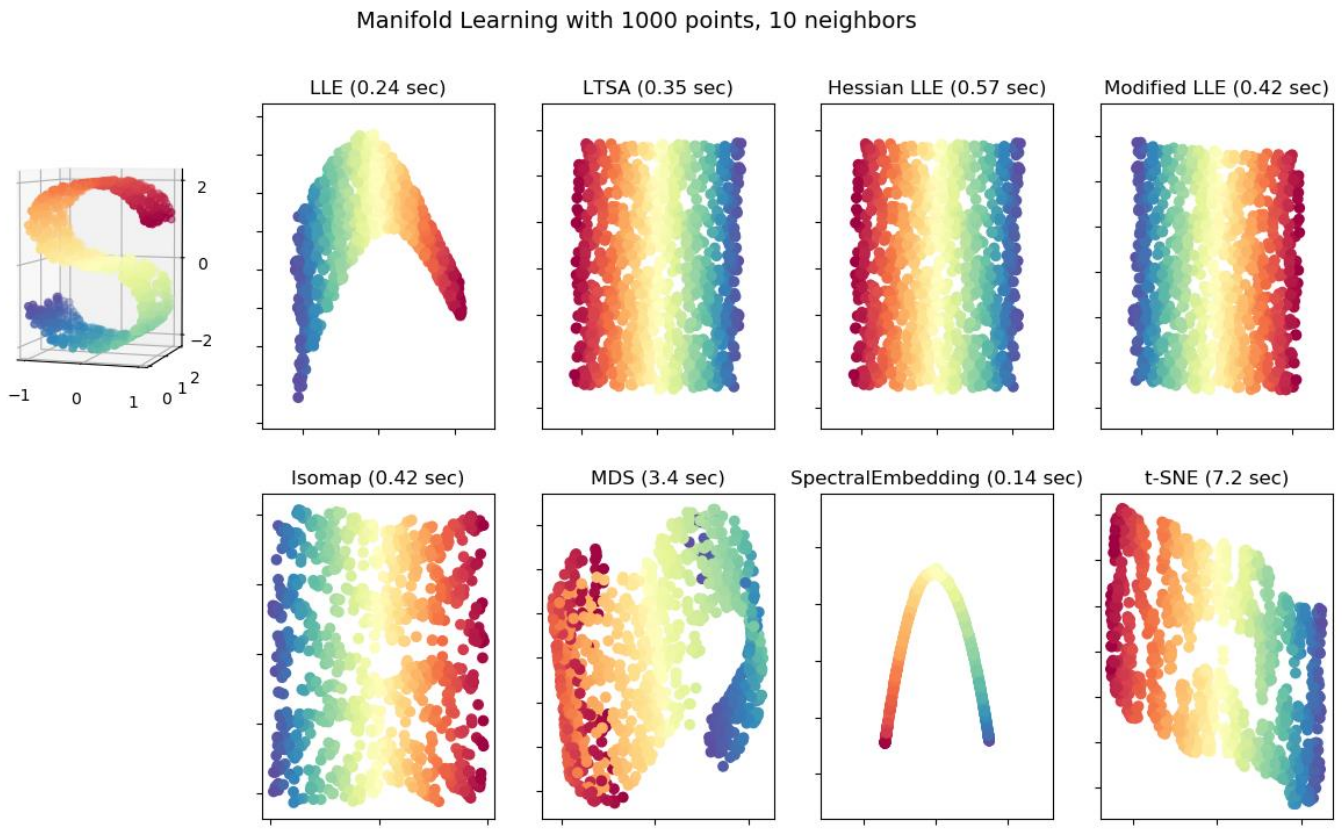
Sparse Coding



https://www.cs.cmu.edu/~rsalakhu/talk_MLSS_part2.pdf

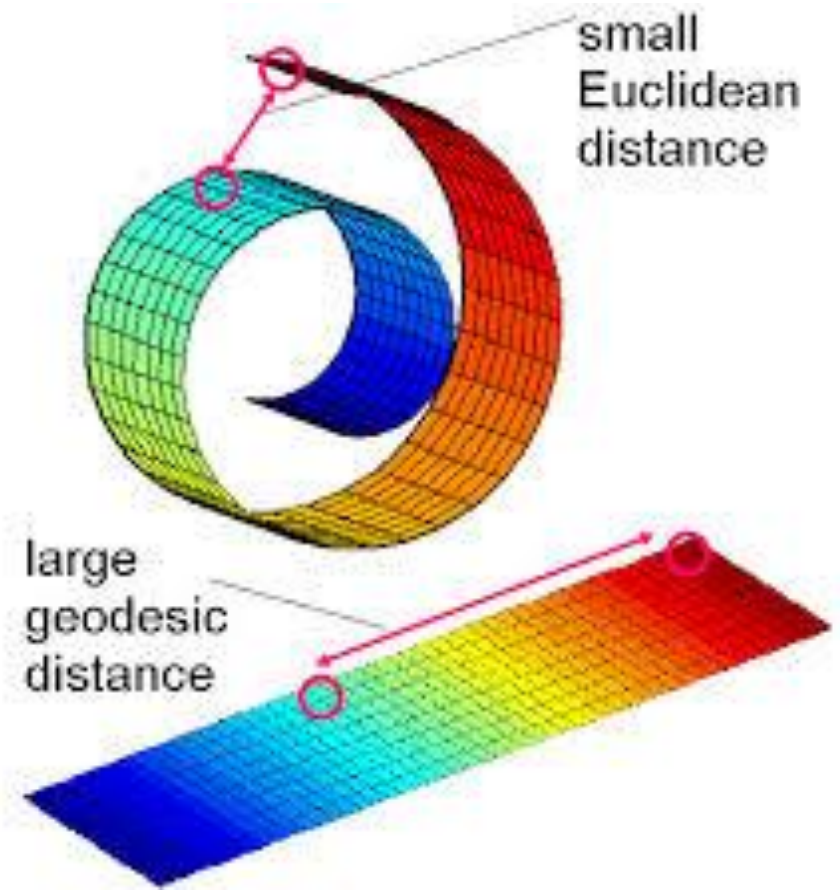
Feature Extraction

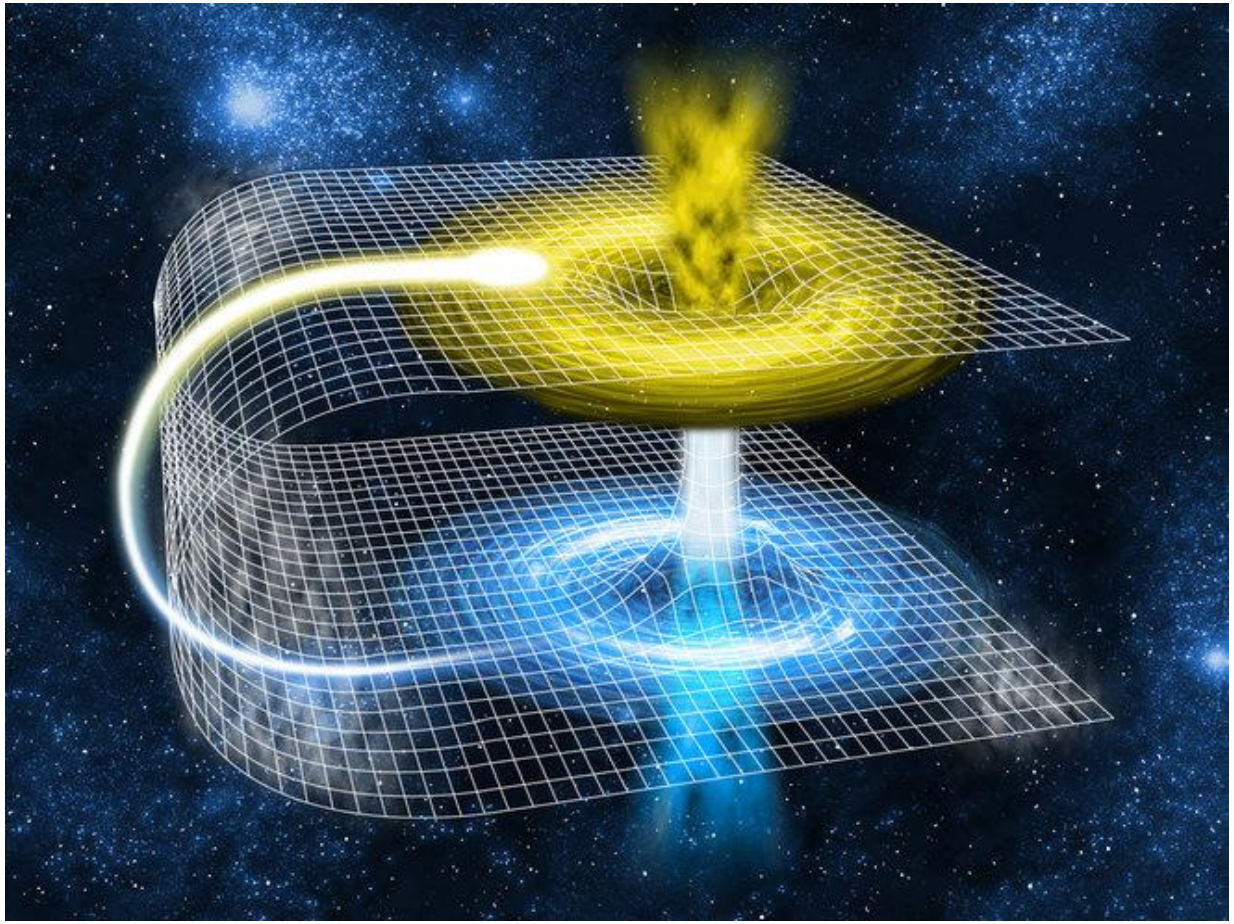
Manifold Learning



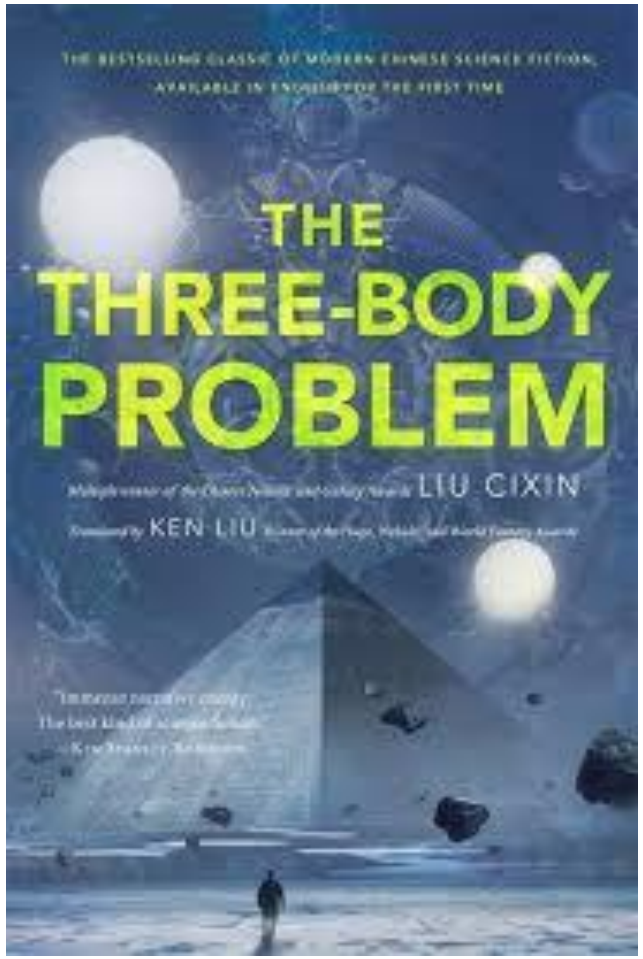
http://scikit-learn.org/stable/modules/manifold.html

Manifold





Manifold



https://www.skeptic.com/insight/flat-earth-conspiracy-theory/

Two Categories

Unsupervised learning

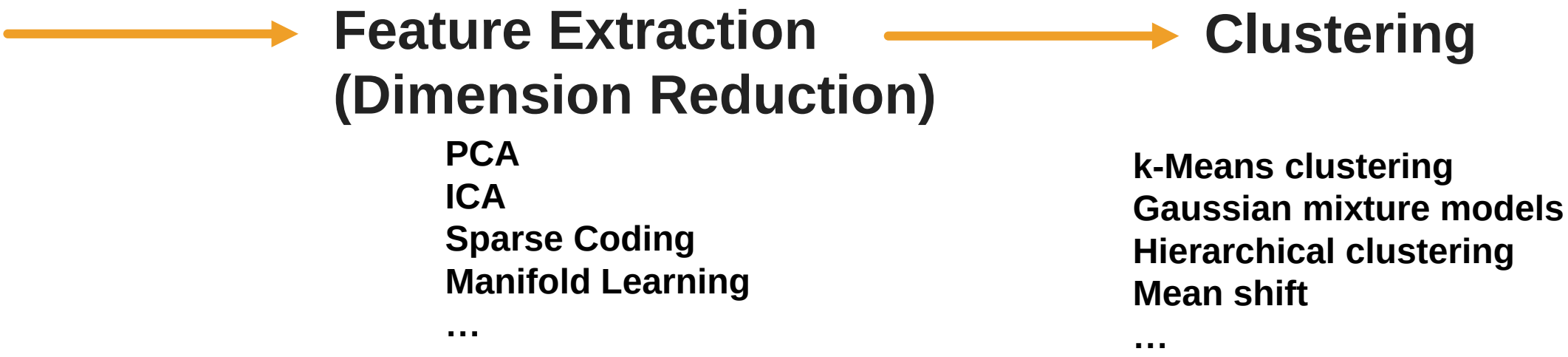
Clustering

k-Means clustering
Gaussian mixture models
Hierarchical clustering
Mean shift
…

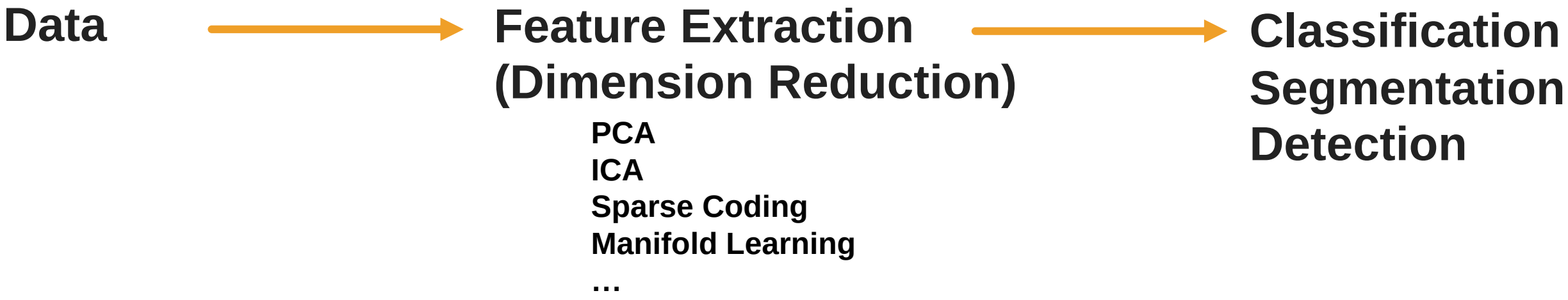
Feature Extraction (Dimension Reduction)

PCA
ICA
Sparse Coding
Manifold Learning
…

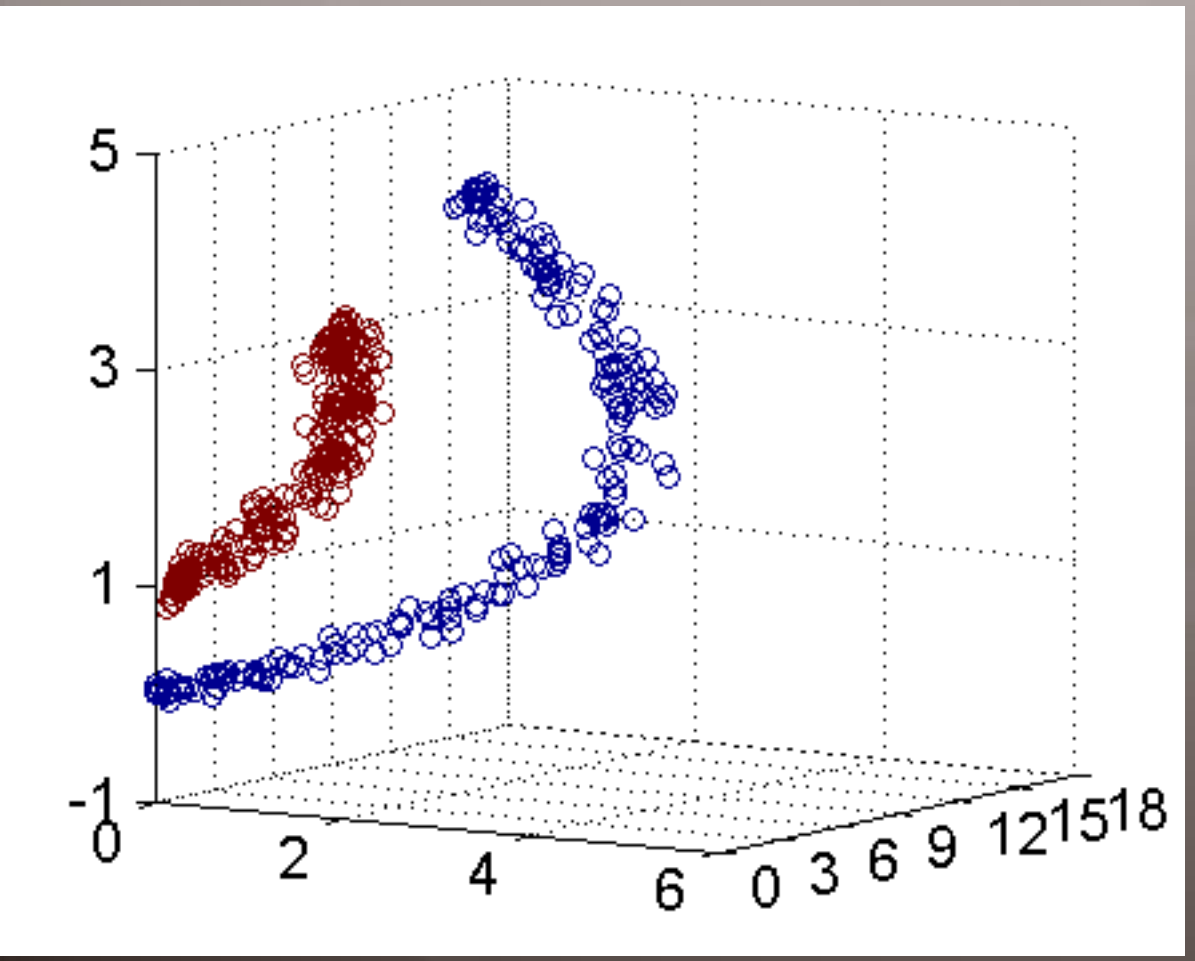
Best Practice

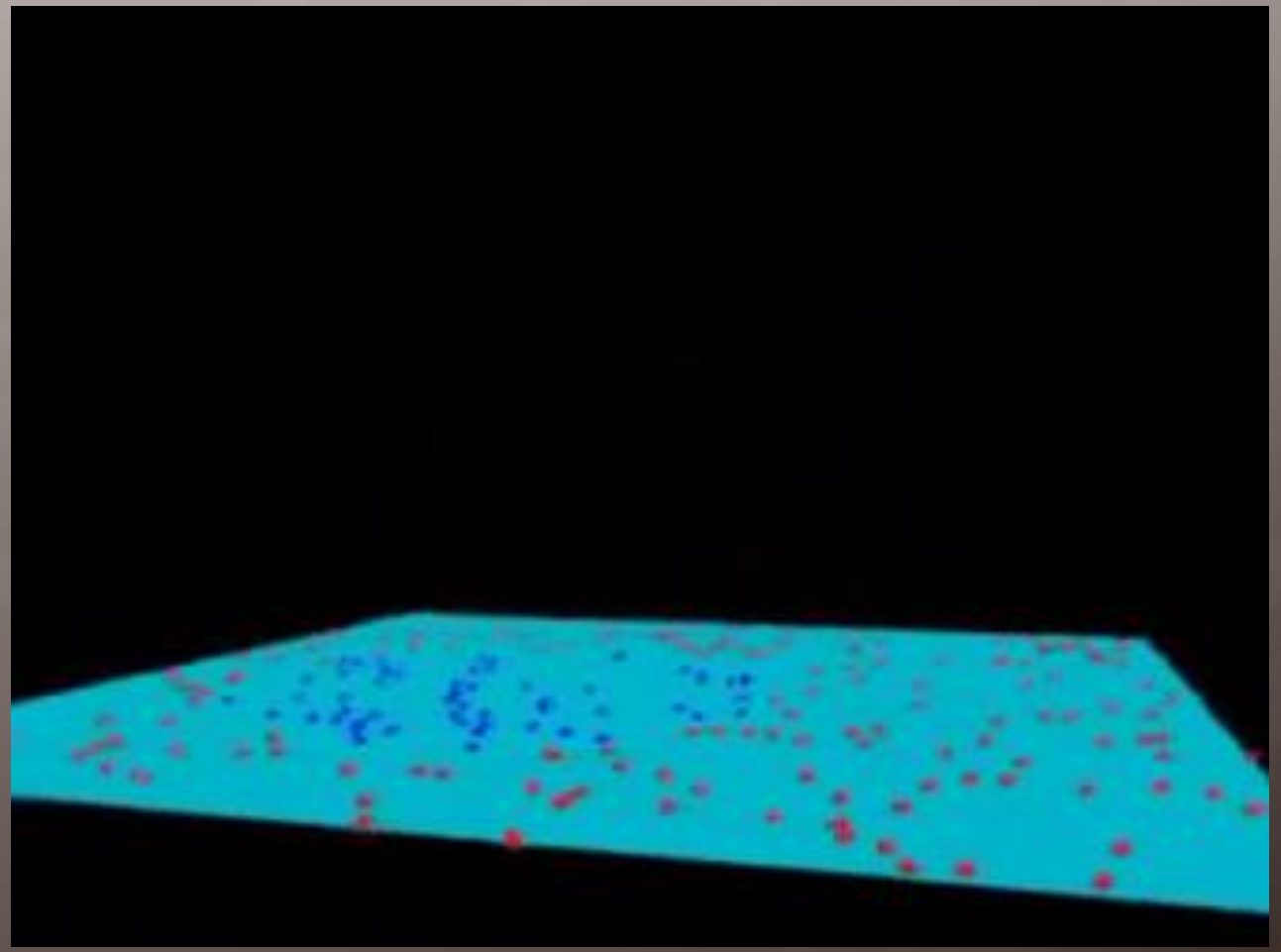


To semi-supervised learning

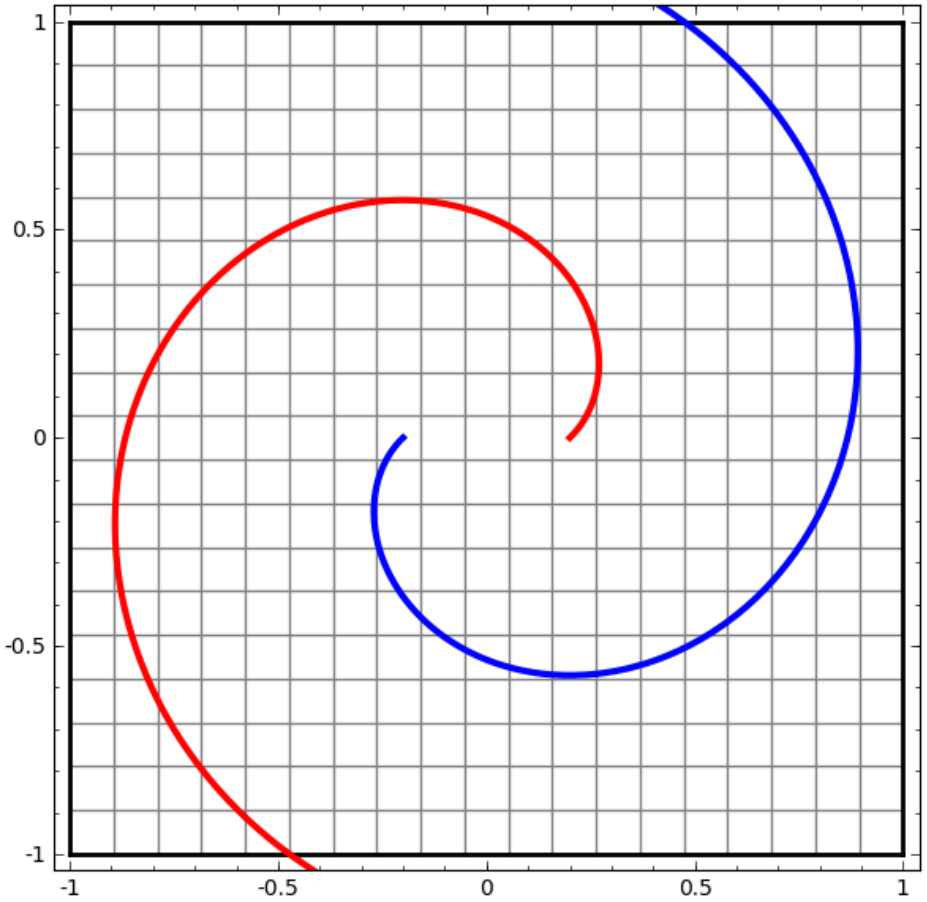


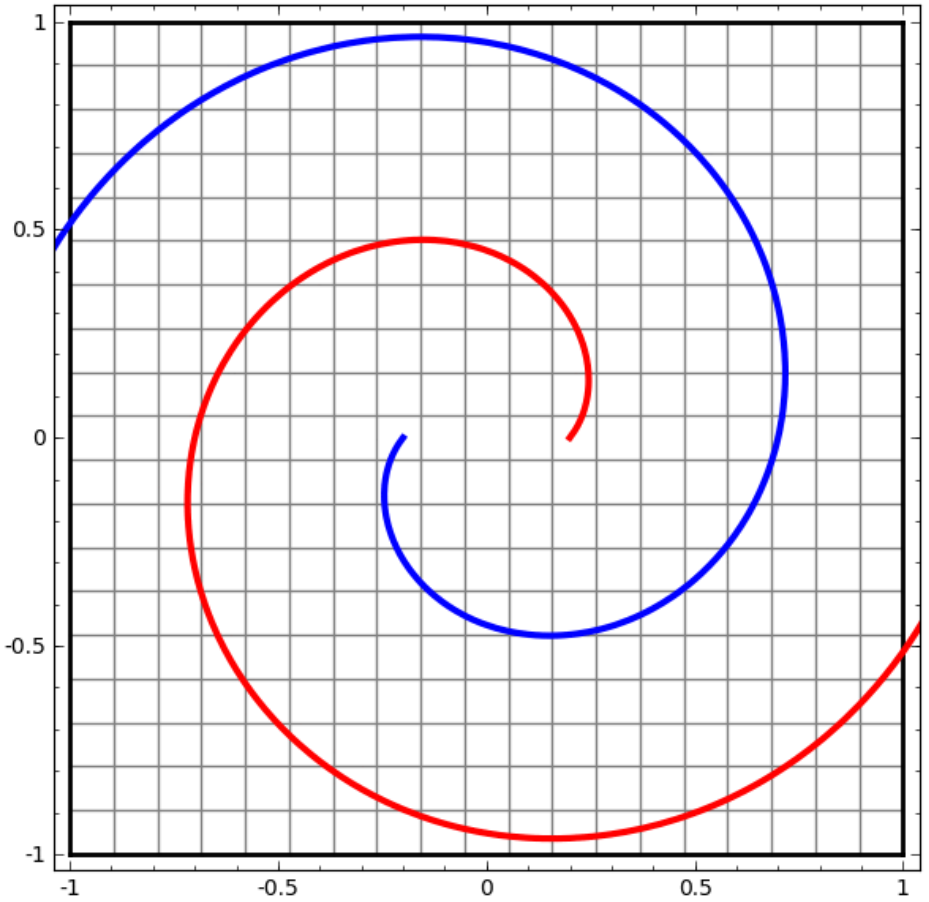
Hyper Plane





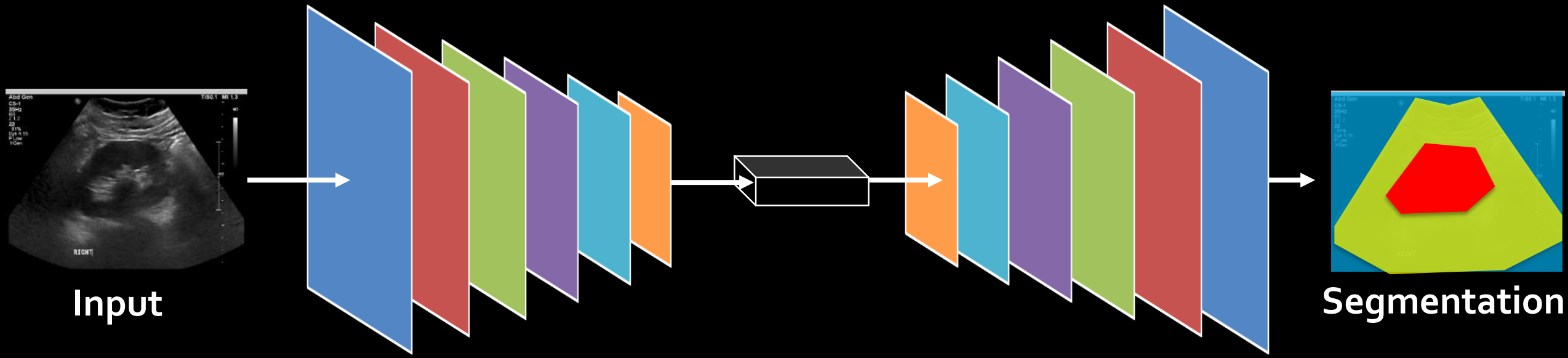
More Hidden Layers



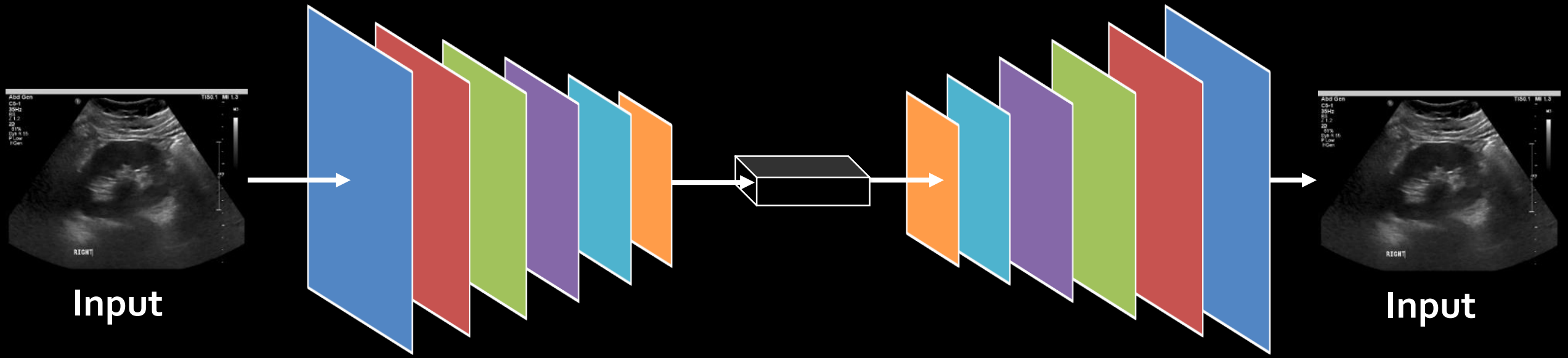


http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

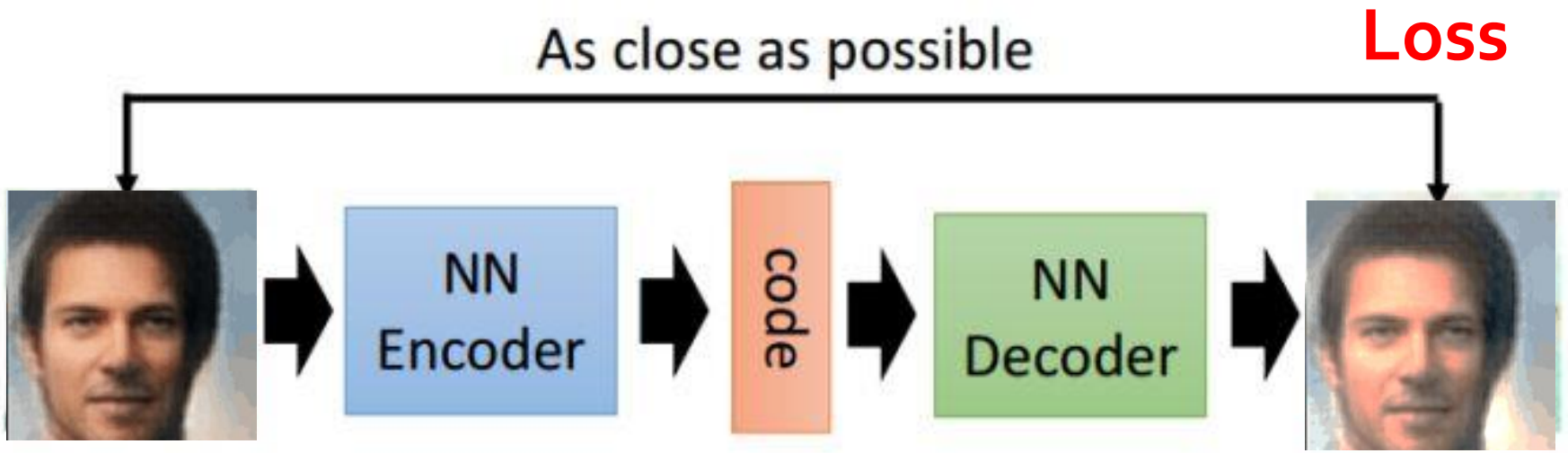
Segmentation (Image to Image)



Reconstruct Itself



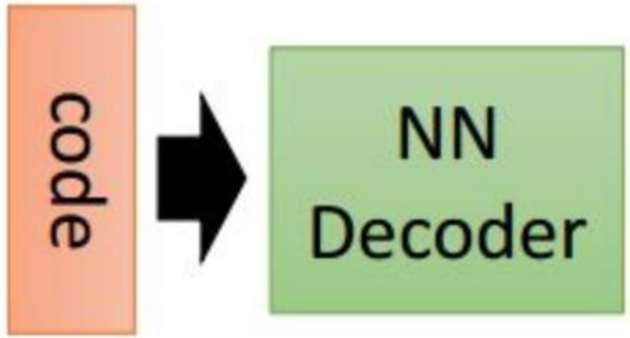
Let Deep Network Do It!

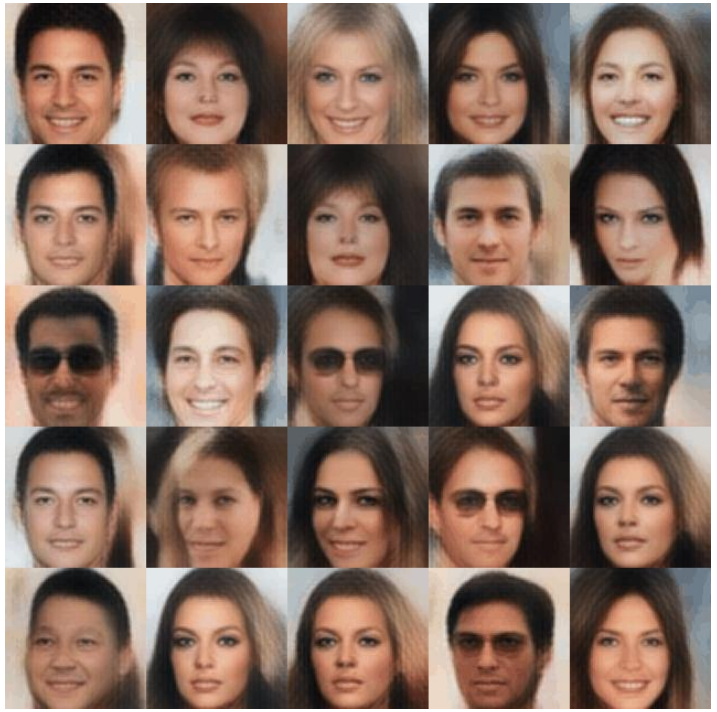


https://zhuanlan.zhihu.com/p/27549418

Auto-encoder

Randomly generate a vector as code





https://zhuanlan.zhihu.com/p/27549418

https://houxianxu.github.io/assets/project/dfcvae

```
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28*28, 128),
            nn.ReLU(True),
            nn.Linear(128, 64),
            nn.ReLU(True),
            nn.Linear(64, 12),
            nn.ReLU(True),
            nn.Linear(12, 3)
        )
        self.decoder = nn.Sequential(
            nn.Linear(3, 12),
            nn.ReLU(True),
            nn.Linear(12, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True),
            nn.Linear(128, 28*28),
            nn.Tanh()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

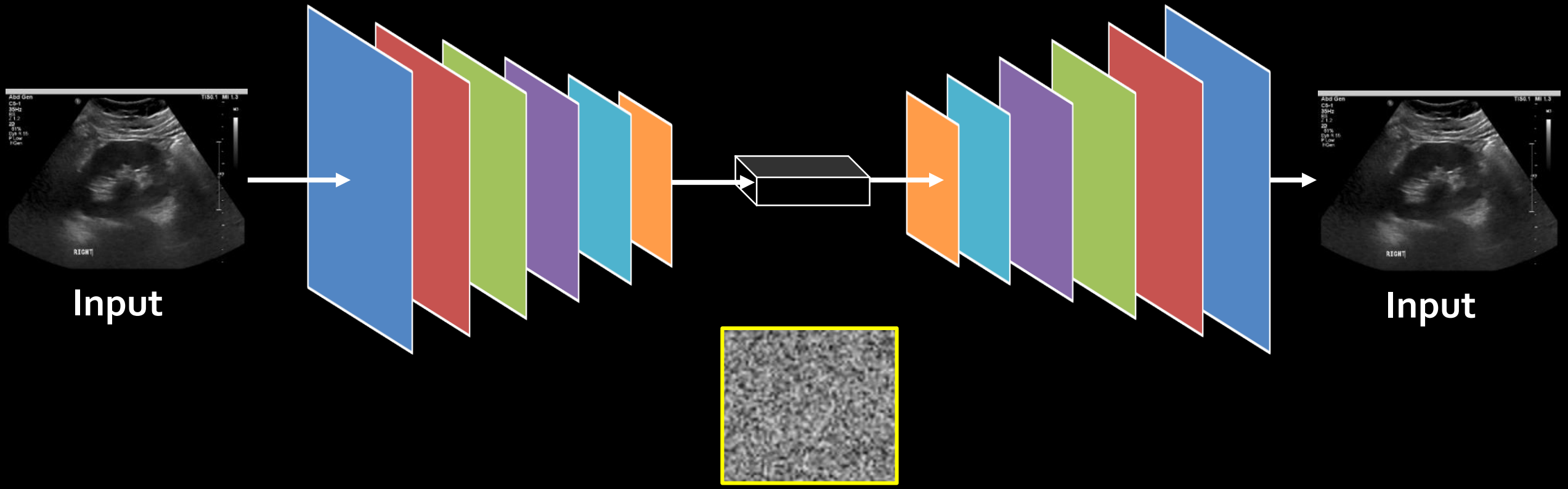
```python
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28*28, 128),
            nn.ReLU(True),
            nn.Linear(128, 64),
            nn.ReLU(True),
            nn.Linear(64, 12),
            nn.ReLU(True),
            nn.Linear(12, 3)
        )
        self.decoder = nn.Sequential(
            nn.Linear(3, 12),
            nn.ReLU(True),
            nn.Linear(12, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True),
            nn.Linear(128, 28*28),
            nn.Tanh()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

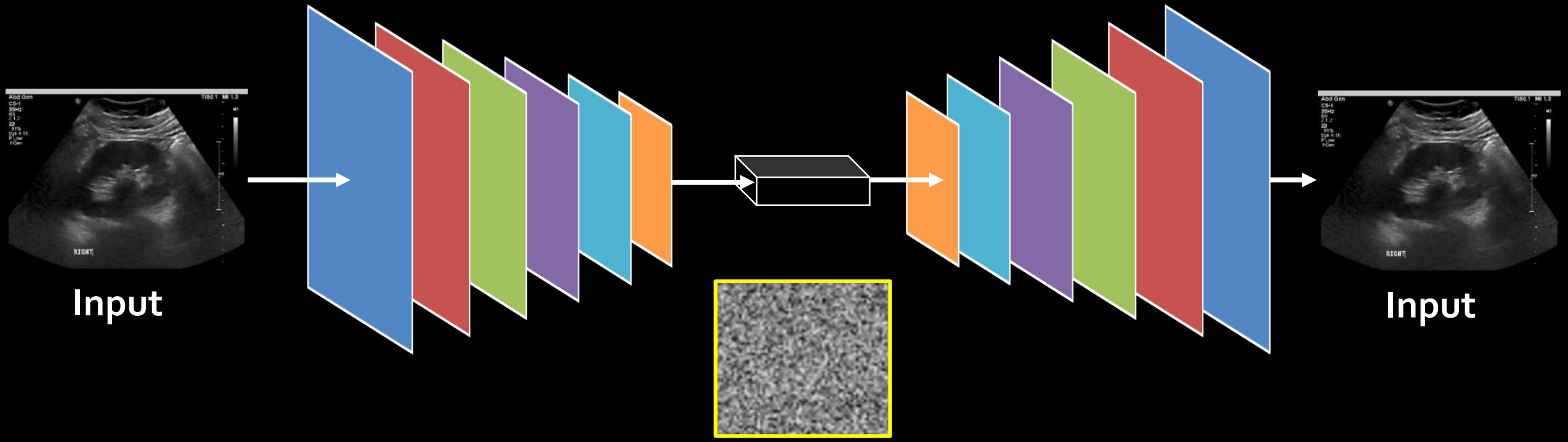
```python
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 16, 3, stride=3, padding=1),  # b, 16,
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=2),  # b, 16, 5, 5
            nn.Conv2d(16, 8, 3, stride=2, padding=1),  # b, 8,
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=1)  # b, 8, 2, 2
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(8, 16, 3, stride=2),  # b, 16, 5
            nn.ReLU(True),
            nn.ConvTranspose2d(16, 8, 5, stride=3, padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(8, 1, 2, stride=2, padding=1),
            nn.Tanh()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

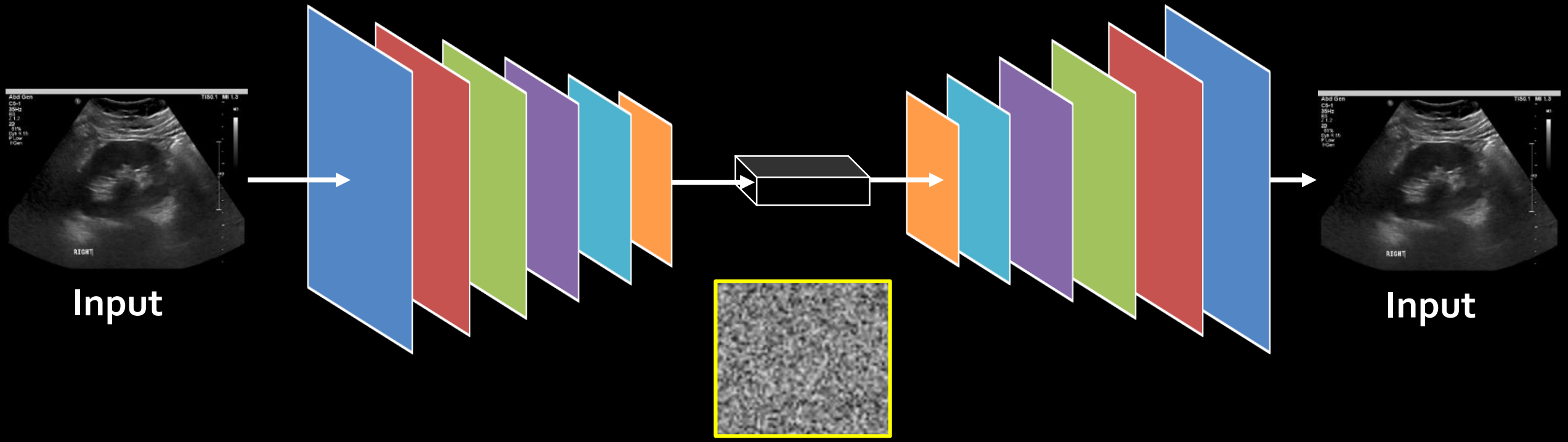
Auto encoder



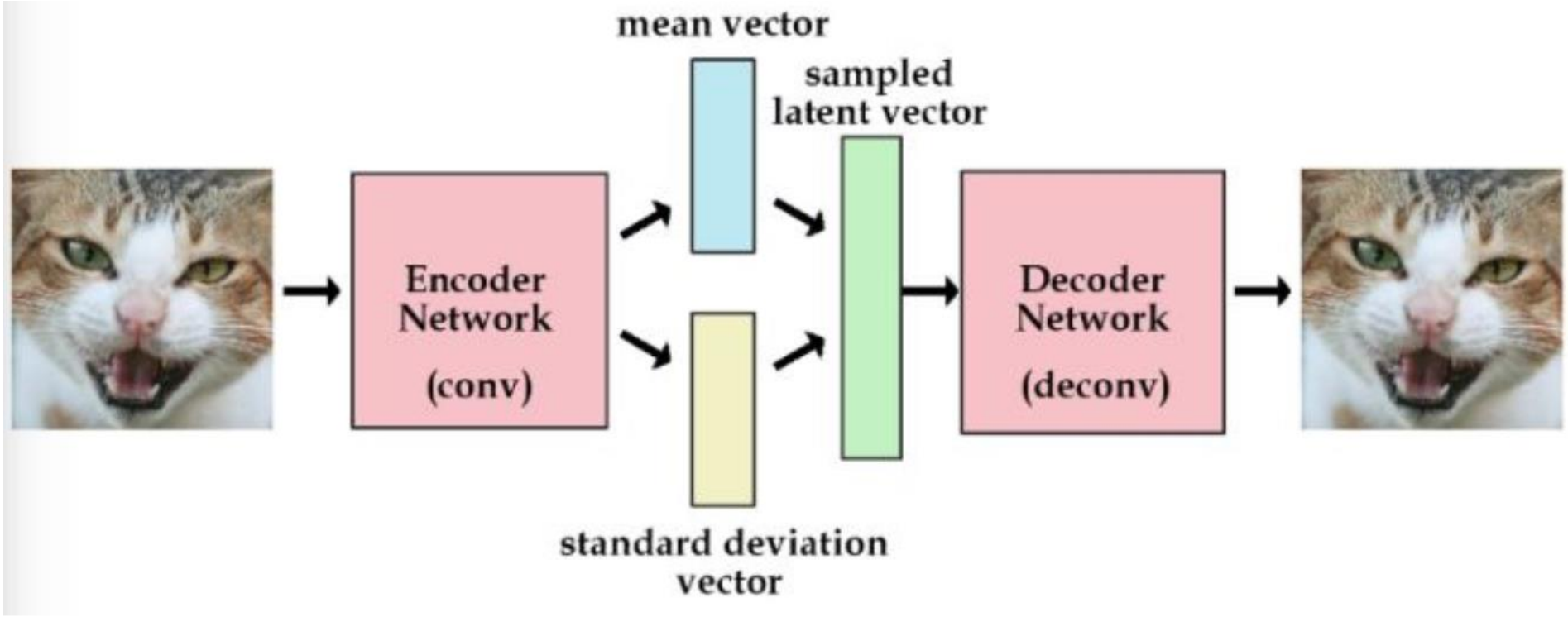
Auto encoder



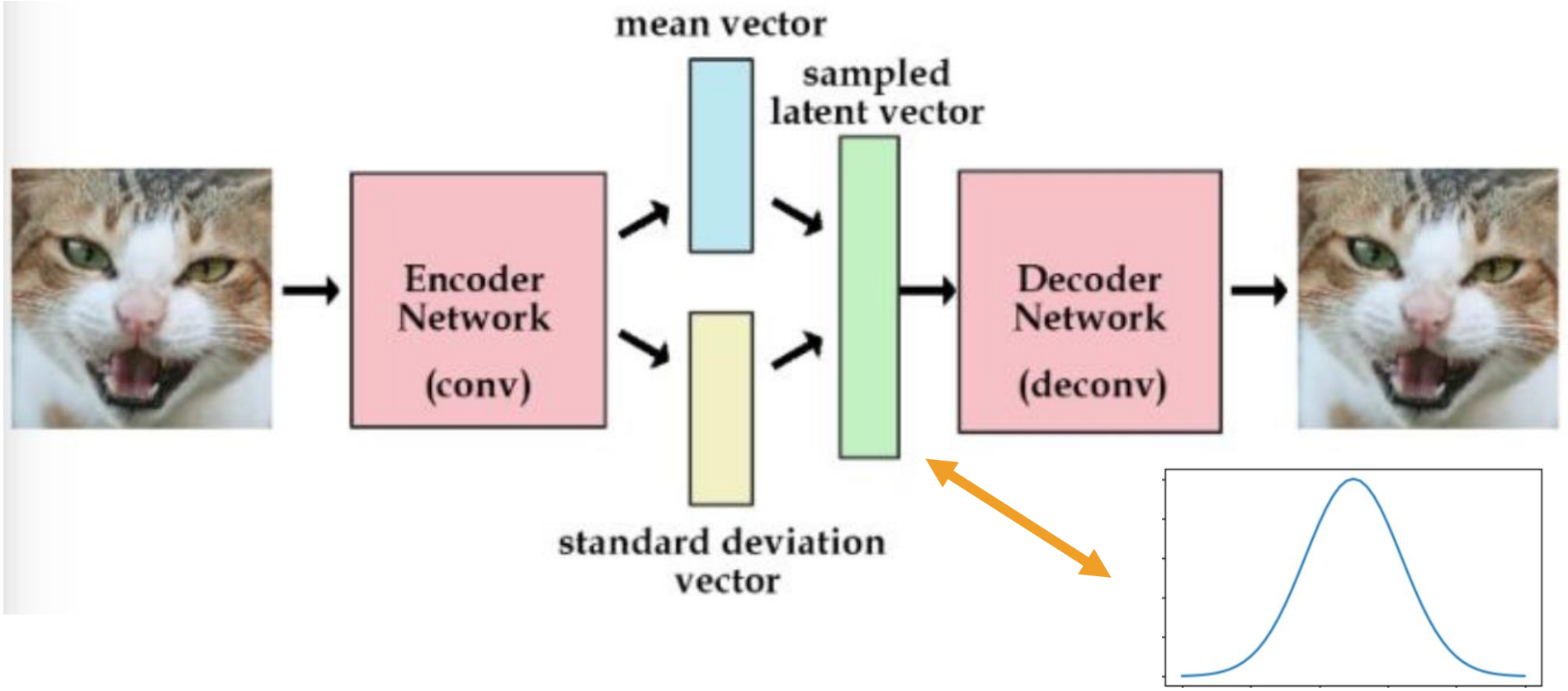
Auto encoder



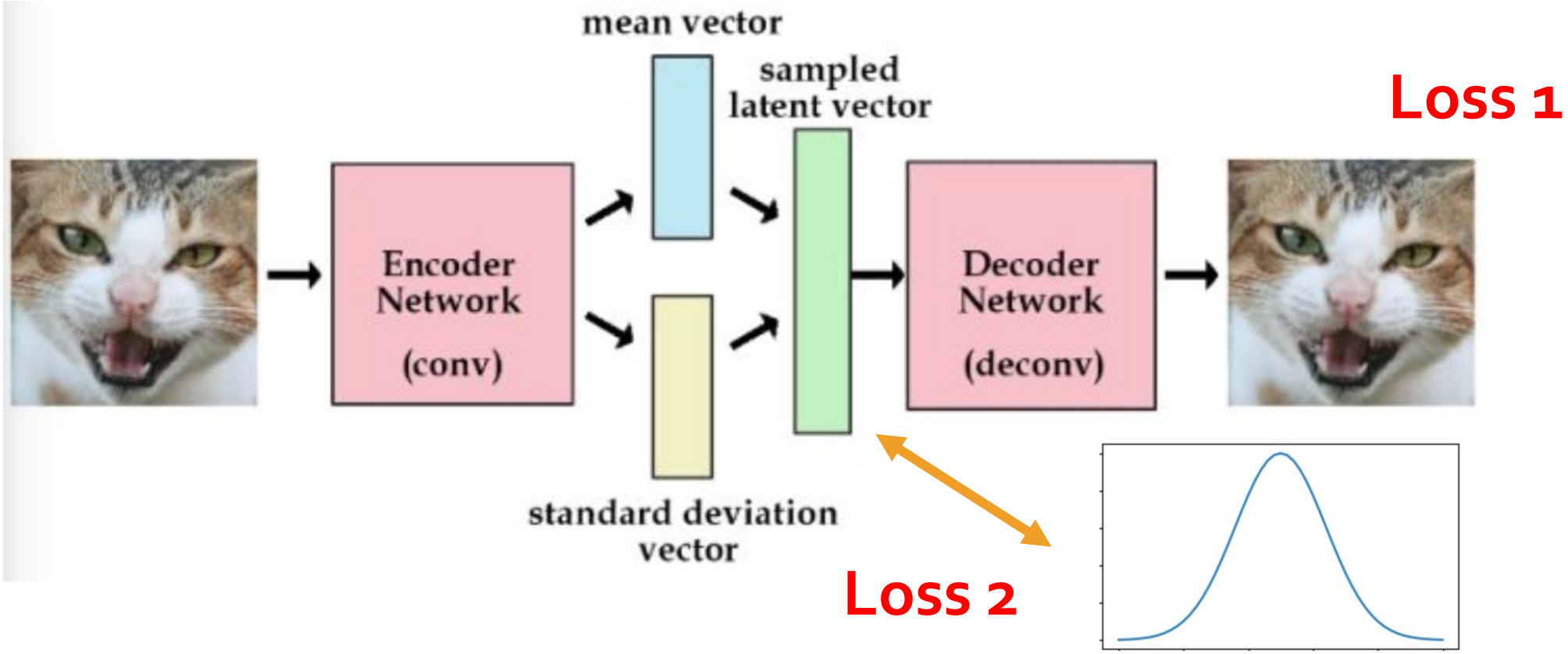
Variational Auto Encoder (VAE)



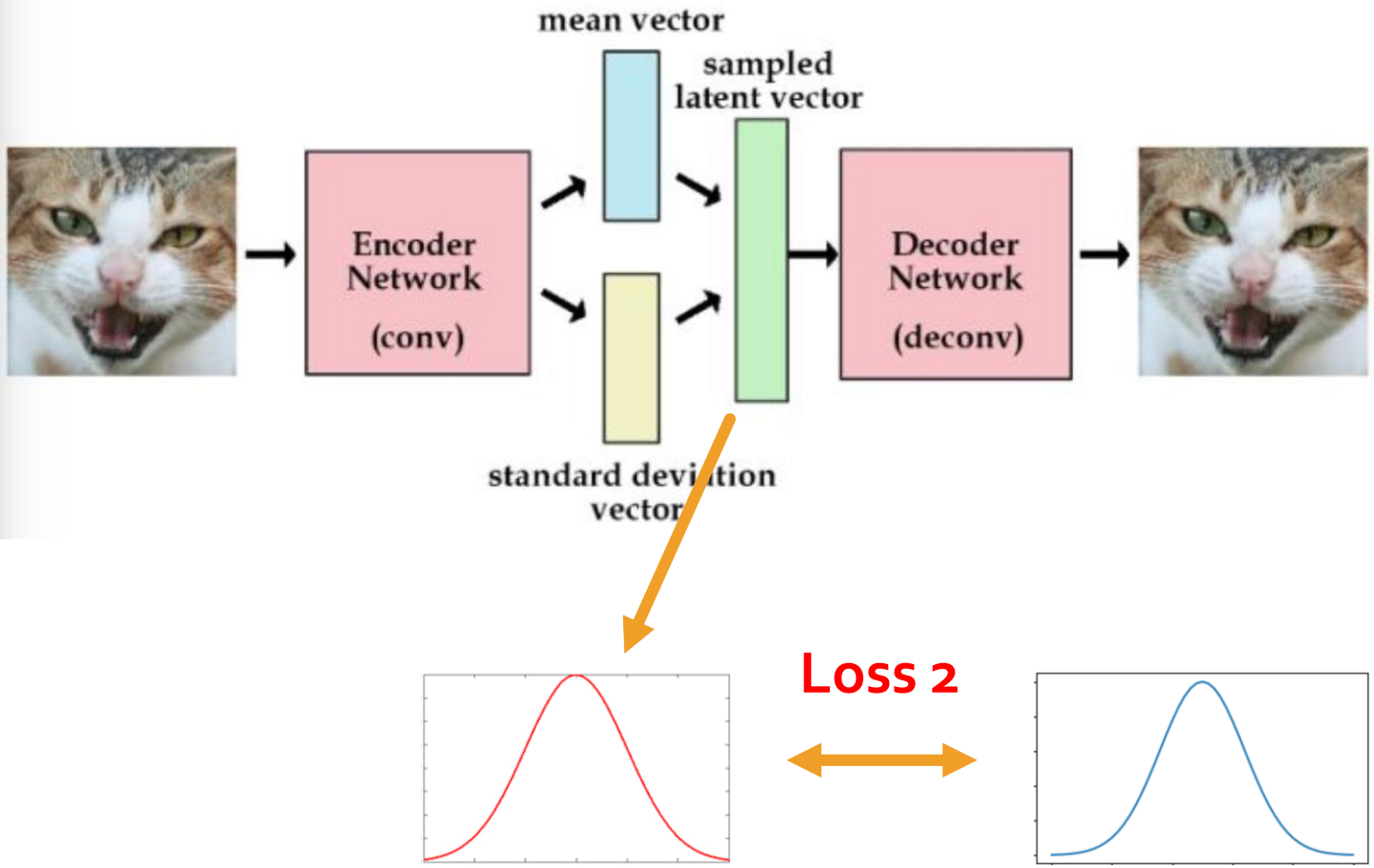
Variational Auto Encoder (VAE)



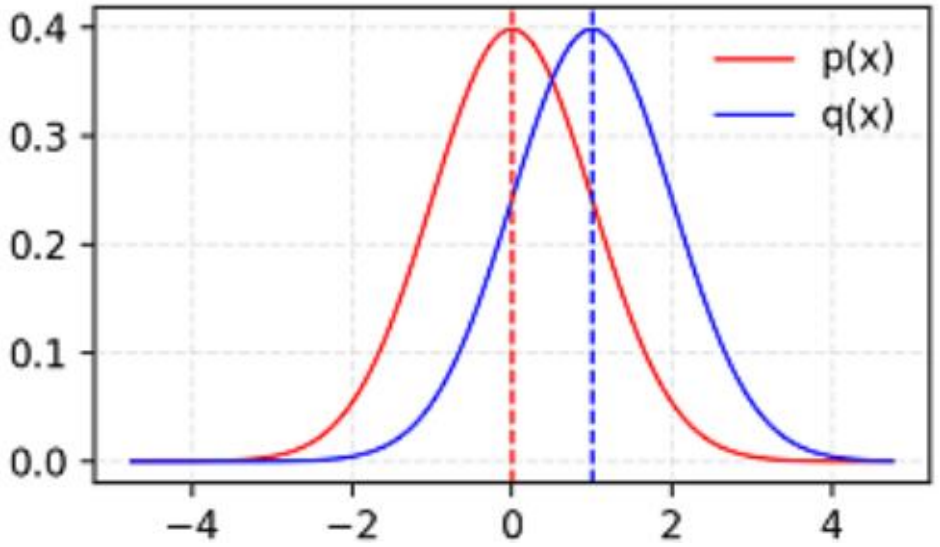
Variational Auto Encoder (VAE)



KL-divergence

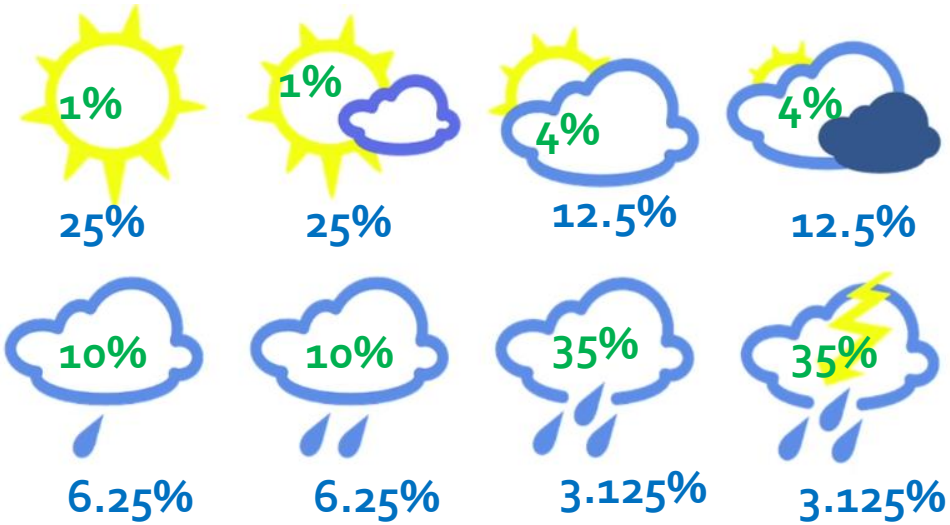


$$DKL(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$



Entropy, Cross-Entropy, KL Divergence

Bad Weather Report



p = true distribution

q = predict distribution

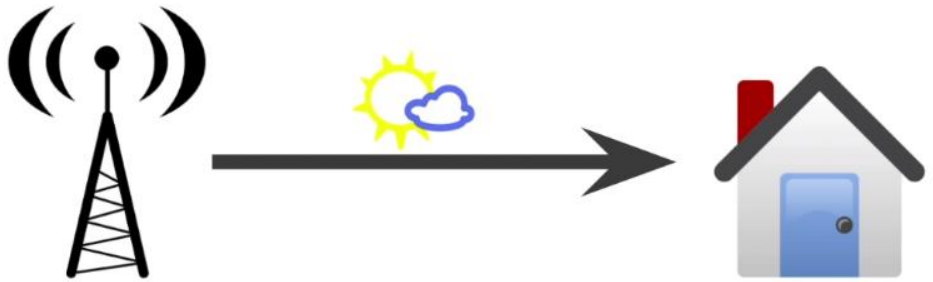
Entropy:
$H(\mathbf{p}) = -\Sigma_i\, p_i \log_2(p_i)$

Cross-Entropy:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log_2(q_i)$

KL Divergence:
$D_{KL}(\mathbf{p} \,||\, \mathbf{q}) = H(\mathbf{p}, \mathbf{q}) - H(\mathbf{p})$

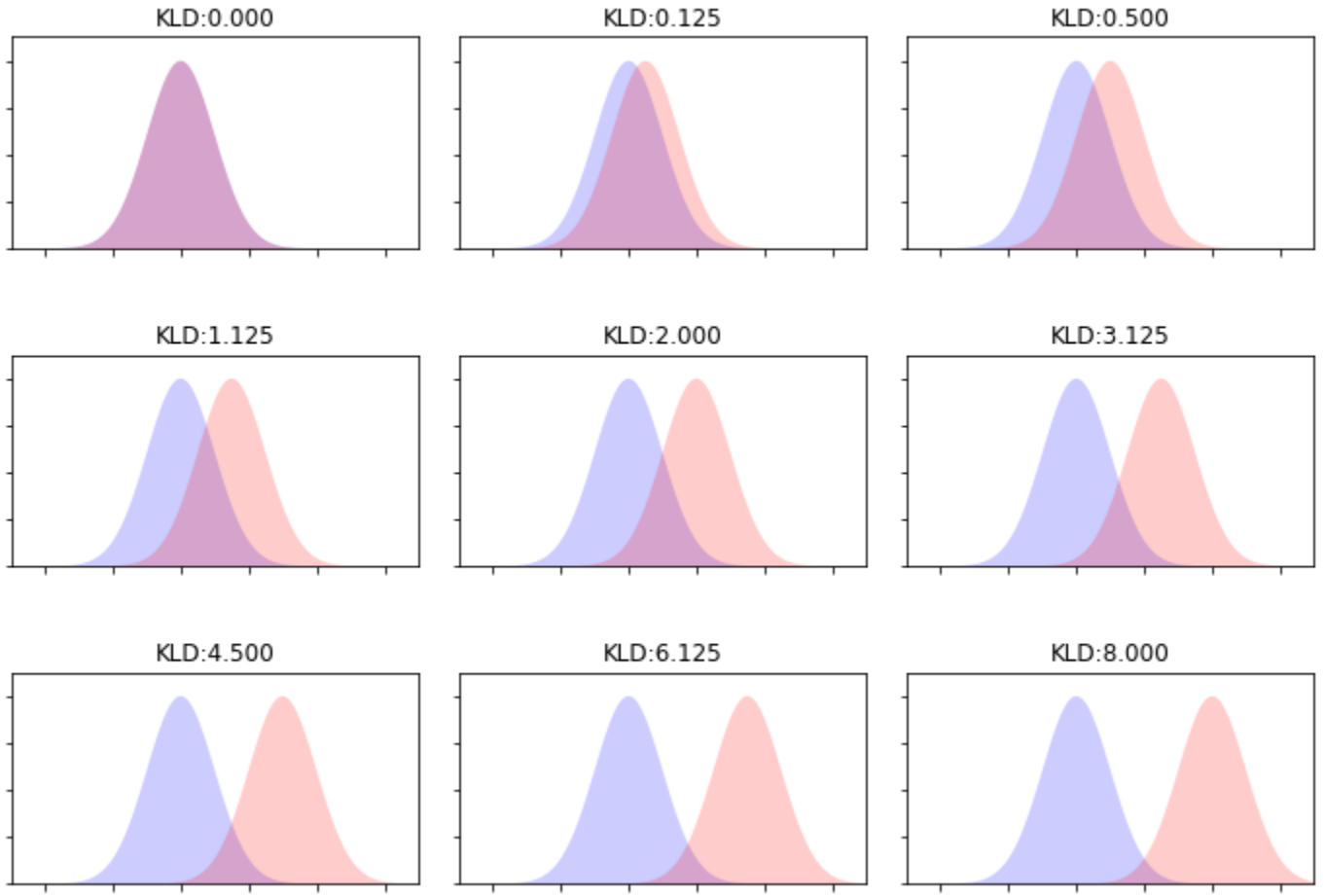
Cross-Entropy = Entropy + KL Divergence

KL Divergence:
$D_{KL}(\mathbf{p} \,||\, \mathbf{q}) = 4.58 - 2.23$



https://www.youtube.com/watch?v=ErfnhcEV1O8

KL divergence



http://yusuke-ujitoko.hatenablog.com/entry/2017/05/07/200022

Reading

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

2.2 The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \quad (1)$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \quad (2)$$

which can also be written as:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] \quad (3)$$

We want to differentiate and optimize the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ w.r.t. both the variational parameters $\boldsymbol{\phi}$ and generative parameters $\boldsymbol{\theta}$. However, the gradient of the lower bound w.r.t. $\boldsymbol{\phi}$ is a bit problematic. The usual (naïve) Monte Carlo gradient estimator for this type of problem is: $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z})] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z})\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$ where $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. This gradient estimator exhibits exhibits very high variance (see e.g. [BJP12]) and is impractical for our purposes.

2.3 The SGVB estimator and AEVB algorithm

In this section we introduce a practical estimator of the lower bound and its derivatives w.r.t. the parameters. We assume an approximate posterior in the form $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, but please note that the technique can be applied to the case $q_{\boldsymbol{\phi}}(\mathbf{z})$, i.e. where we do not condition on $\mathbf{x}$, as well. The fully variational Bayesian method for inferring a posterior over the parameters is given in the appendix.

Under certain mild conditions outlined in section 2.4 for a chosen approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ we can reparameterize the random variable $\widetilde{\mathbf{z}} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using a differentiable transformation $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ of an (auxiliary) noise variable $\boldsymbol{\epsilon}$:

$$\widetilde{\mathbf{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \quad (4)$$

See section 2.4 for general strategies for chosing such an approriate distribution $p(\boldsymbol{\epsilon})$ and function $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$. We can now form Monte Carlo estimates of expectations of some function $f(\mathbf{z})$ w.r.t. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as follows:

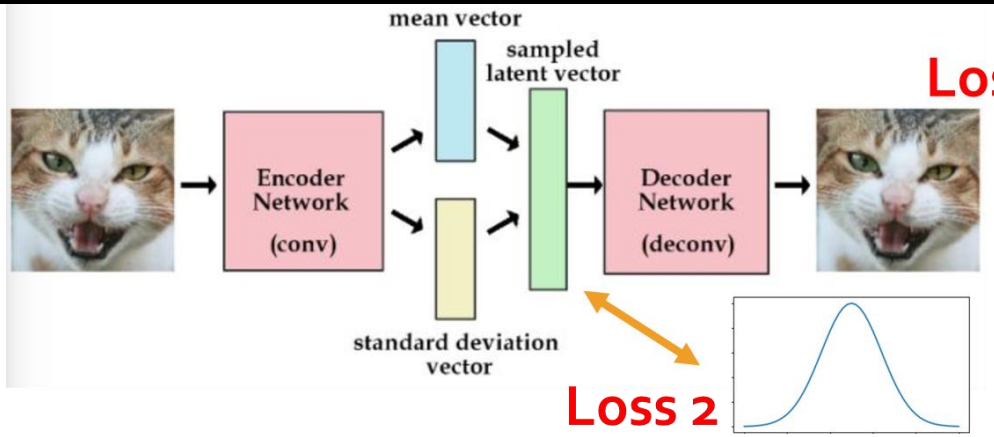
$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}[f(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \quad (5)$$

We apply this technique to the variational lower bound (eq. (2)), yielding our generic Stochastic Gradient Variational Bayes (SGVB) estimator $\widetilde{\mathcal{L}}^{A}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$:

$$\widetilde{\mathcal{L}}^{A}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)}|\mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \quad (6)$$

Two Loss



$$DKL(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

```python
def loss_function(recon_x, x, mu, logvar):
    """
    recon_x: generating images
    x: origin images
    mu: latent mean
    logvar: latent log variance
    """
    BCE = reconstruction_function(recon_x, x)
    # loss = 0.5 * sum(1 + log(sigma^2) - mu^2 - sigma^2)
    KLD_element = mu.pow(2).add_(logvar.exp()).mul_(-1).add_(1).add_(logvar)
    KLD = torch.sum(KLD_element).mul_(-0.5)
    # KL divergence
    return BCE + KLD
```

```python
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        self.fc1 = nn.Linear(784, 400)
        self.fc21 = nn.Linear(400, 20)
        self.fc22 = nn.Linear(400, 20)
        self.fc3 = nn.Linear(20, 400)
        self.fc4 = nn.Linear(400, 784)

    def encode(self, x):
        h1 = F.relu(self.fc1(x))
        return self.fc21(h1), self.fc22(h1)

    def reparametrize(self, mu, logvar):
        std = logvar.mul(0.5).exp_()
        if torch.cuda.is_available():
            eps = torch.cuda.FloatTensor(std.size()).normal_()
        else:
            eps = torch.FloatTensor(std.size()).normal_()
        eps = Variable(eps)
        return eps.mul(std).add_(mu)

    def decode(self, z):
        h3 = F.relu(self.fc3(z))
        return F.sigmoid(self.fc4(h3))

    def forward(self, x):
        mu, logvar = self.encode(x)
        z = self.reparametrize(mu, logvar)
        return self.decode(z), mu, logvar
```

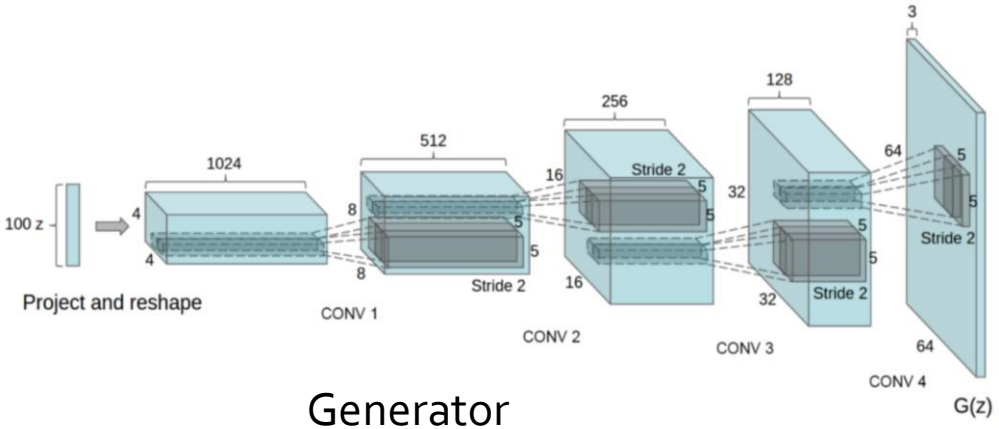
AE vs. VAE

Auto Encoder

Variational Auto Encoder

https://zhuanlan.zhihu.com/p/27549418

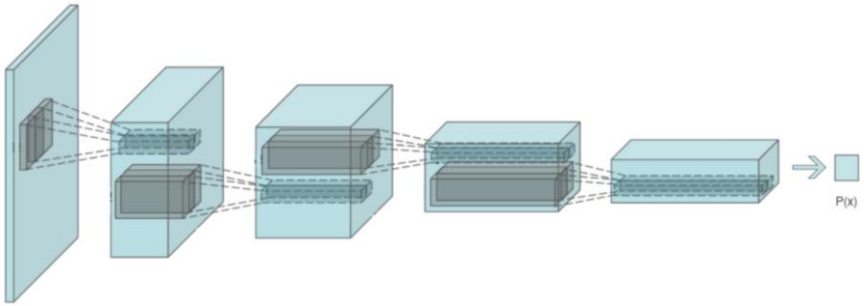
GAN



Generator

Fake

True



Discriminator

https://arxiv.org/abs/1511.06434

VAE vs. GAN

Entropy:
$H(\mathbf{p}) = -\Sigma_i p_i \log_2(p_i)$

Cross-Entropy:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i p_i \log_2(q_i)$

KL Divergence:
$D_{KL}(\mathbf{p} \,||\, \mathbf{q}) = H(\mathbf{p}, \mathbf{q}) - H(\mathbf{p})$

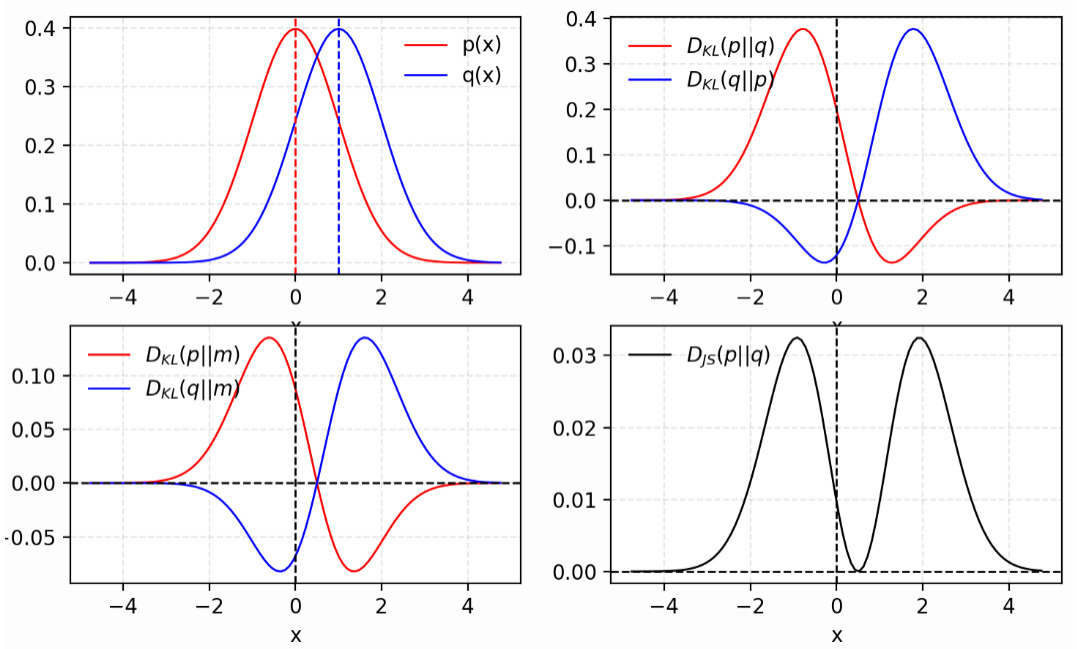
Cross-Entropy = Entropy + KL Divergence

VAE:

$$D_{KL}(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

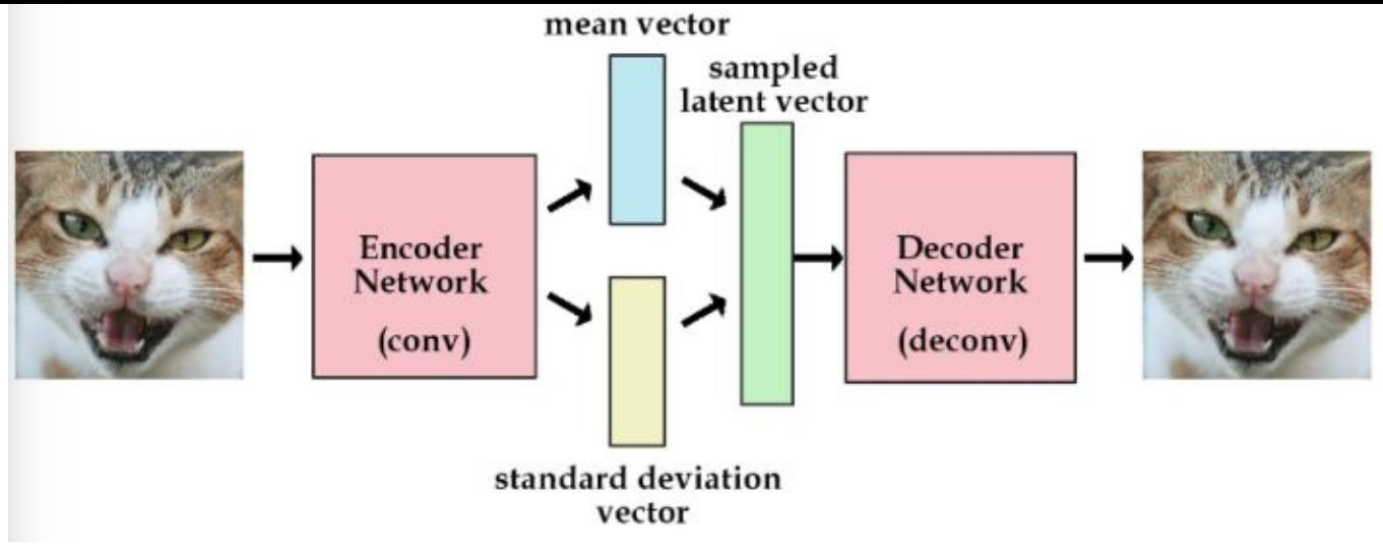
GAN:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2})$$

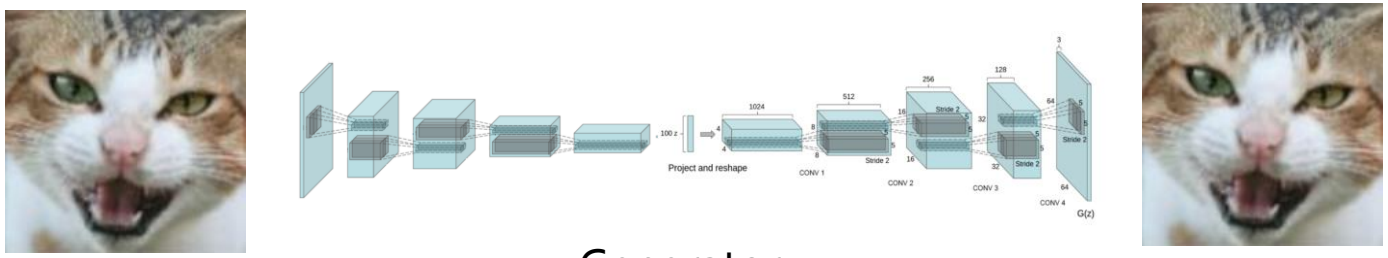


https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html
https://www.youtube.com/watch?v=ErfnhcEV1O8

VAE vs. GAN



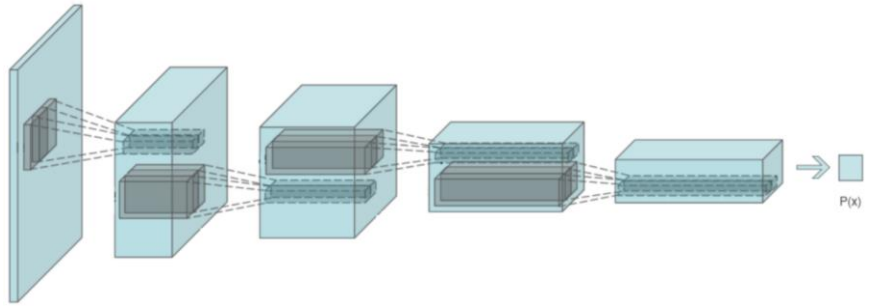
KLD Loss



Generator

Fake

JSD Loss



https://arxiv.org/abs/1511.06434

VAE-GAN

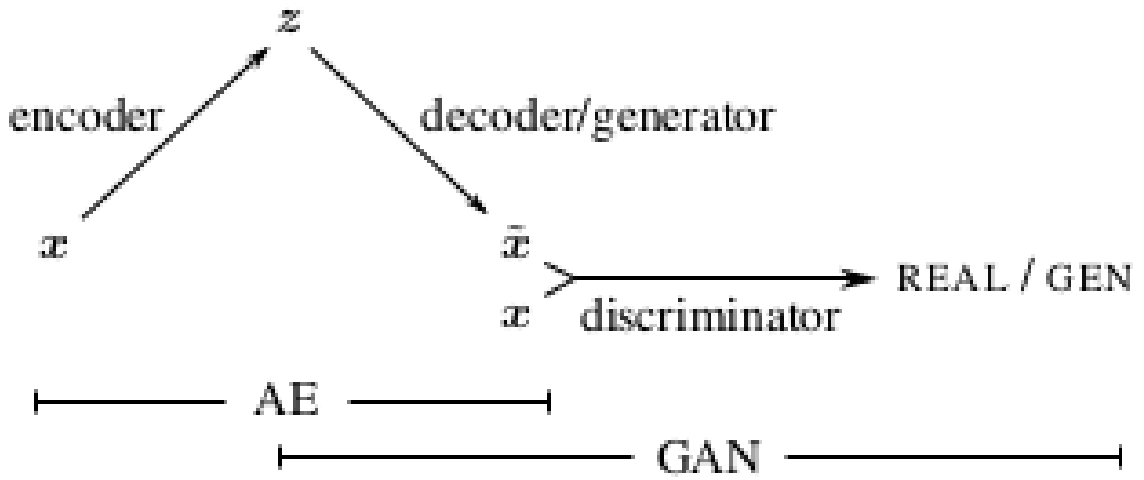
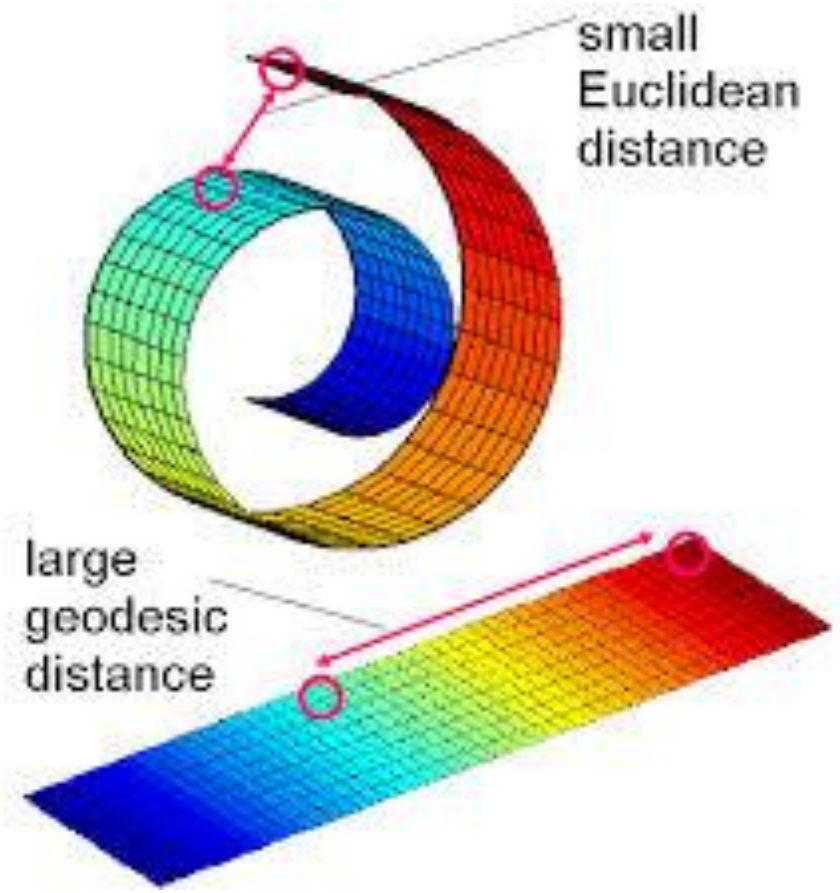
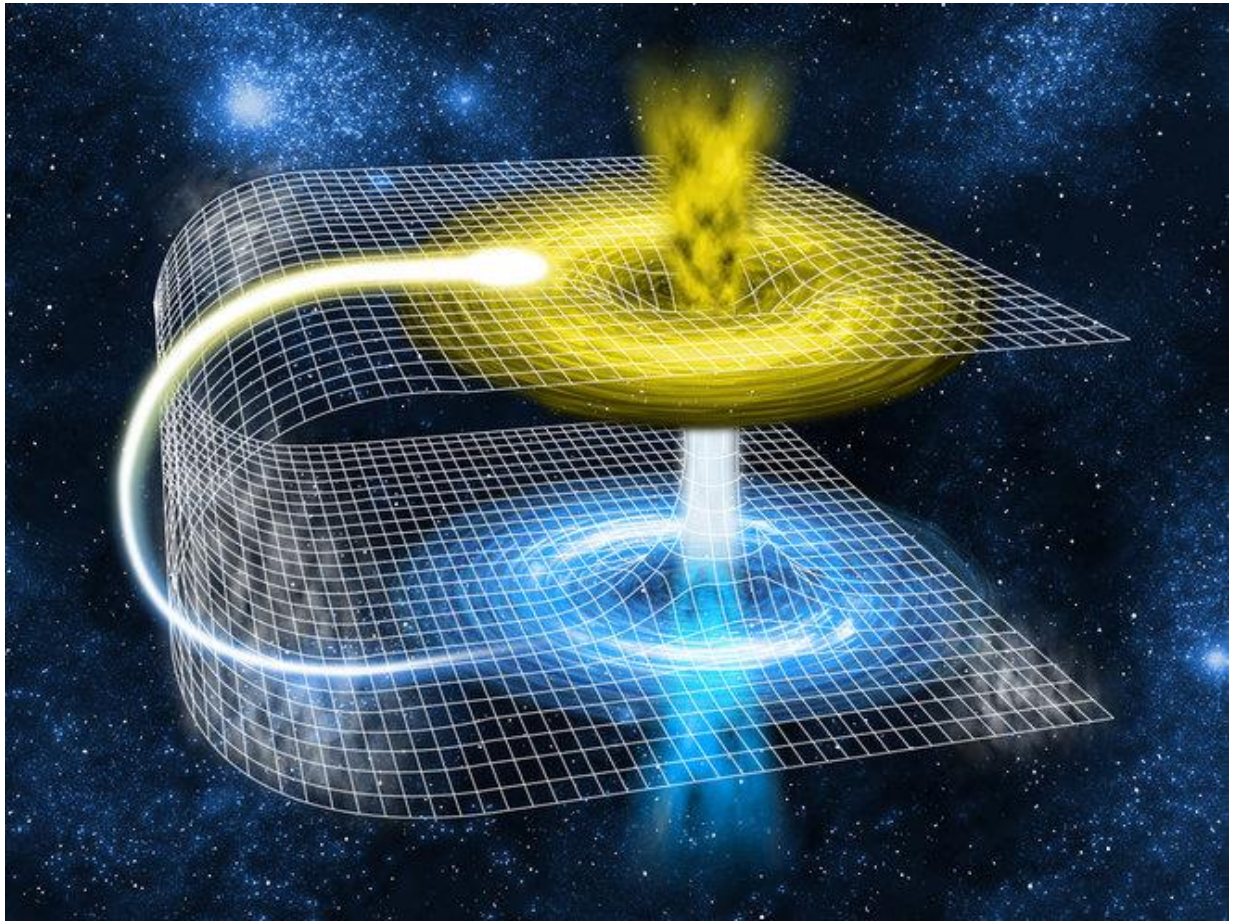


Figure 1. Overview of our network. We combine a VAE with a GAN by collapsing the decoder and the generator into one.

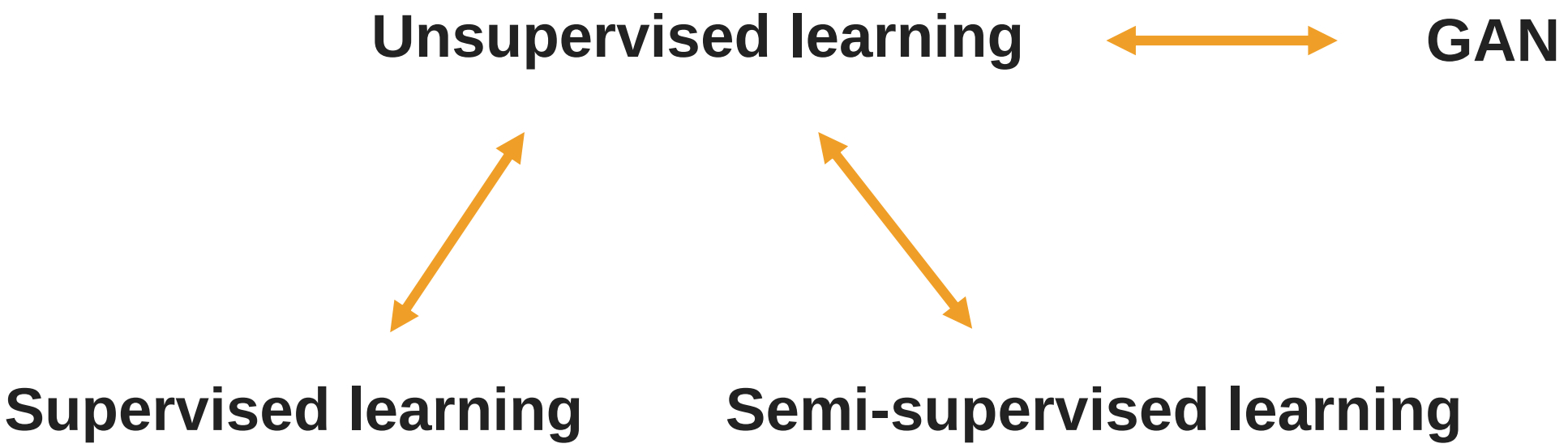
https://pravn.wordpress.com/category/vae-gan-vaegan/

Manifold





Summery


Unsupervised learning
GAN
Supervised learning
Semi-supervised learning

Deep Autoencoding Models for Unsupervised Anomaly Segmentation in Brain MR Images

Christoph Baur[1], Benedikt Wiestler[3], Shadi Albarqouni[1], and Nassir Navab[1,2]

[1] Computer Aided Medical Procedures (CAMP), TU Munich, Germany
[2] Whiting School of Engineering, Johns Hopkins University, Baltimore, United States
[3] Neuroradiology Department, Klinikum Rechts der Isar, TU Munich, Germany

Abstract. Reliably modeling normality and differentiating abnormal appearances from normal cases is a very appealing approach for detecting pathologies in medical images. A plethora of such unsupervised anomaly detection approaches has been made in the medical domain, based on statistical methods, content-based retrieval, clustering and recently also deep learning. Previous approaches towards deep unsupervised anomaly detection model patches of normal anatomy with variants of Autoencoders or GANs, and detect anomalies either as outliers in the learned feature space or from large reconstruction errors. In contrast to these patch-based approaches, we show that deep spatial autoencoding models can be efficiently used to capture normal anatomical variability of entire 2D brain MR images. A variety of experiments on real MR data containing MS lesions corroborates our hypothesis that we can detect and even delineate anomalies in brain MR images by simply comparing input images to their reconstruction. Results show that constraints on the latent space and adversarial training can further improve the segmentation performance over standard deep representation learning.

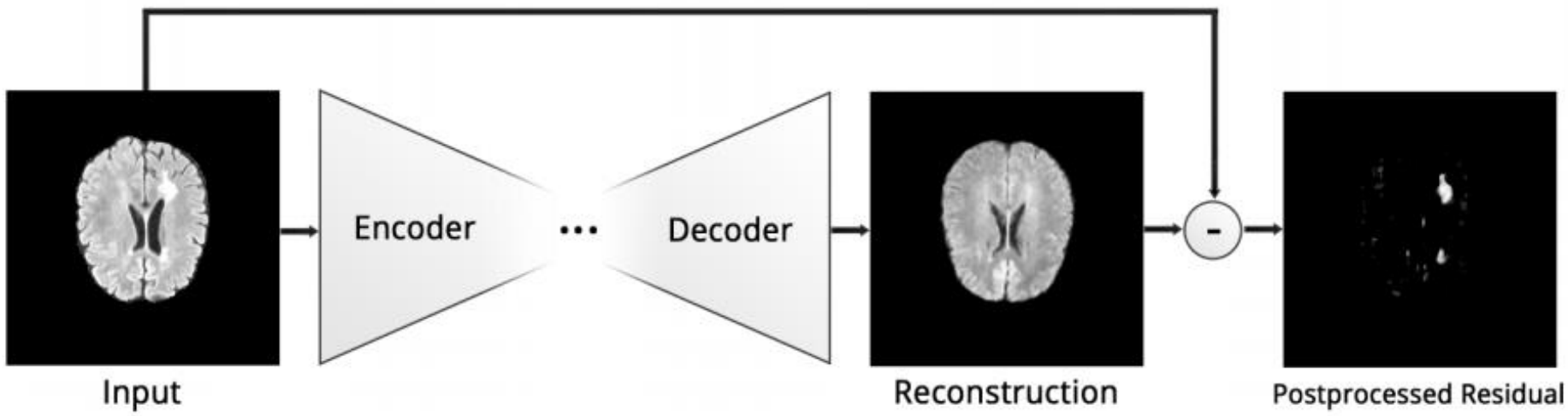
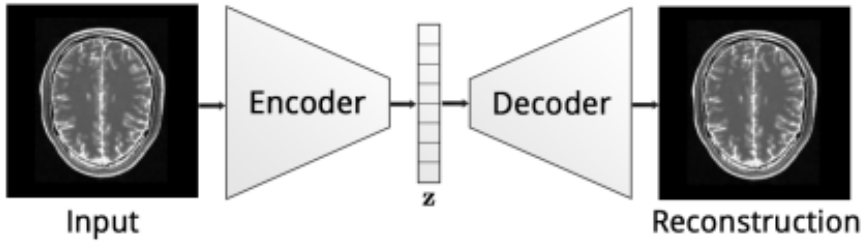
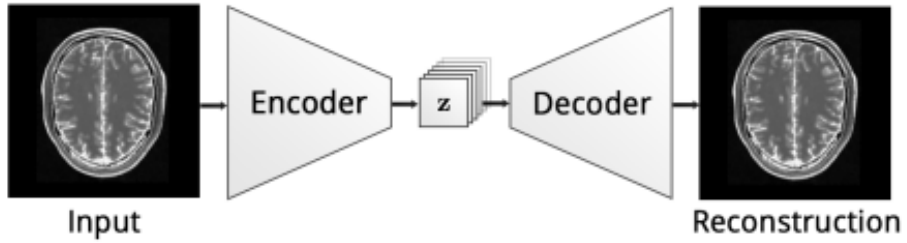


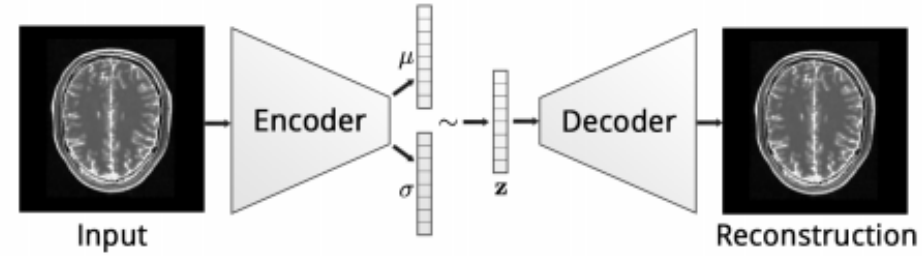
Fig. 1. The proposed anomaly detection oncept at a glance. A simple subtraction of the reconstructed image from the input reveals lesions in the brain.



(a) Dense Autoencoder dAE



(b) Spatial Autoencoder sAE



(c) Dense Variational Autoencoder dVAE

(d) Spatial Variational Autoencoder sVAE

Fig. 3. An overview of different Autoencoder frameworks

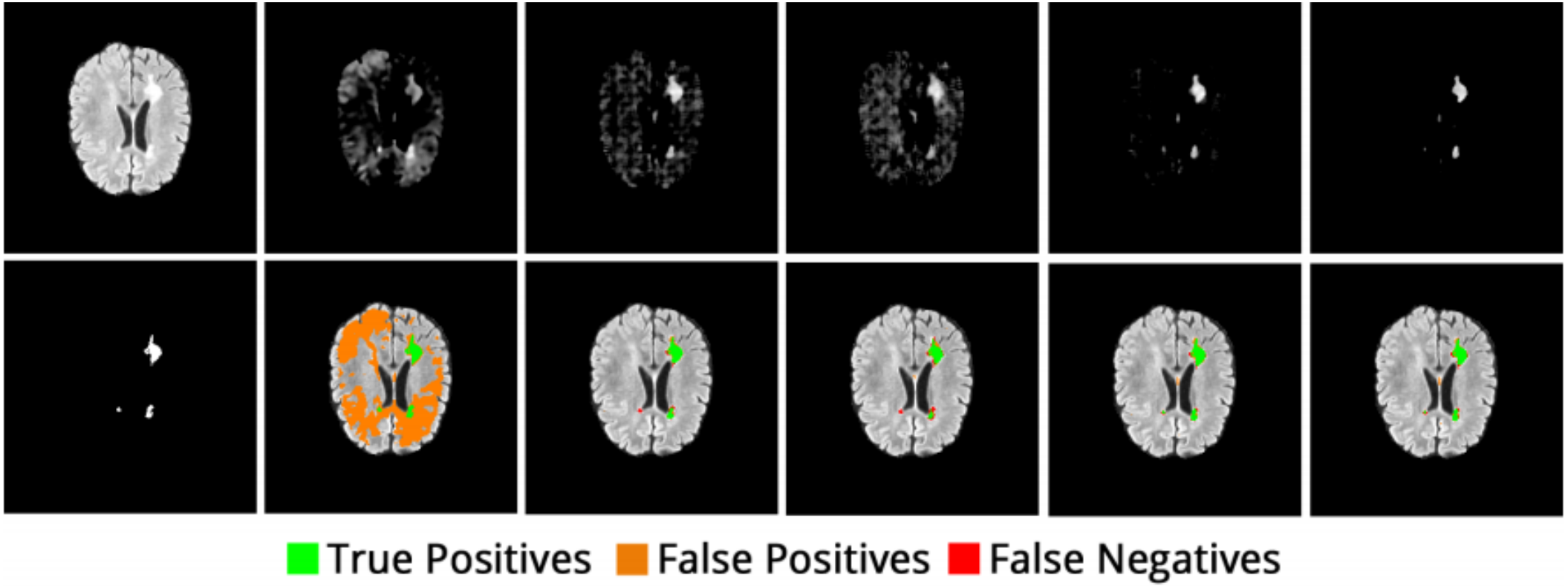


Fig. 5. 1st Column: a selected axial slice and its ground-truth segmentation; Succeeding columns show the filtered difference images (top row) and the resulting segmentation augmented to the input image (bottom row) for the following models (in order): dAE, sAE_3, sAE-GAN, sVAE and sVAE-GAN.

Improving Surgical Training Phantoms by Hyperrealism: Deep Unpaired Image-to-Image Translation from Real Surgeries

Sandy Engelhardt[1,3(✉)], Raffaele De Simone[2], Peter M. Full[2], Matthias Karck[2], and Ivo Wolf[3]

Department of Simulation and Graphics & Research Campus STIMULATE, Magdeburg University, Magdeburg, Germany
s.engelhardt@hs-mannheim.de
[2] Department of Cardiac Surgery, Heidelberg University Hospital, Heidelberg, Germany
[3] Faculty of Computer Science, Mannheim University of Applied Sciences, Mannheim, Germany

Abstract. Current 'dry lab' surgical phantom simulators are a valuable tool for surgeons which allows them to improve their dexterity and skill with surgical instruments. These phantoms mimic the haptic and shape of organs of interest, but lack a realistic visual appearance. In this work, we present an innovative application in which representations learned from real intraoperative endoscopic sequences are transferred to a surgical phantom scenario. The term *hyperrealism* is introduced in this field, which we regard as a novel subform of surgical augmented reality for approaches that involve real-time object transfigurations. For related tasks in the computer vision community, unpaired cycle-consistent Generative Adversarial Networks (GANs) have shown excellent results on still RGB images. Though, application of this approach to continuous video frames can result in flickering, which turned out to be especially prominent for this application. Therefore, we propose an extension of cycle-consistent GANs, named *tempCycleGAN*, to improve temporal consistency. The novel method is evaluated on captures of a silicone phantom for training endoscopic reconstructive mitral valve procedures. Synthesized videos show highly realistic results with regard to (1) replacement of the silicone appearance of the phantom valve by intraoperative tissue texture, while (2) explicitly keeping crucial features in the scene, such as instruments, sutures and prostheses. Compared to the original CycleGAN approach, *tempCycleGAN* efficiently removes flickering between frames. The overall approach is expected to change the future design of surgical training simulators since the generated sequences clearly demonstrate the feasibility to enable a considerably more realistic training experience for minimally-invasive procedures.

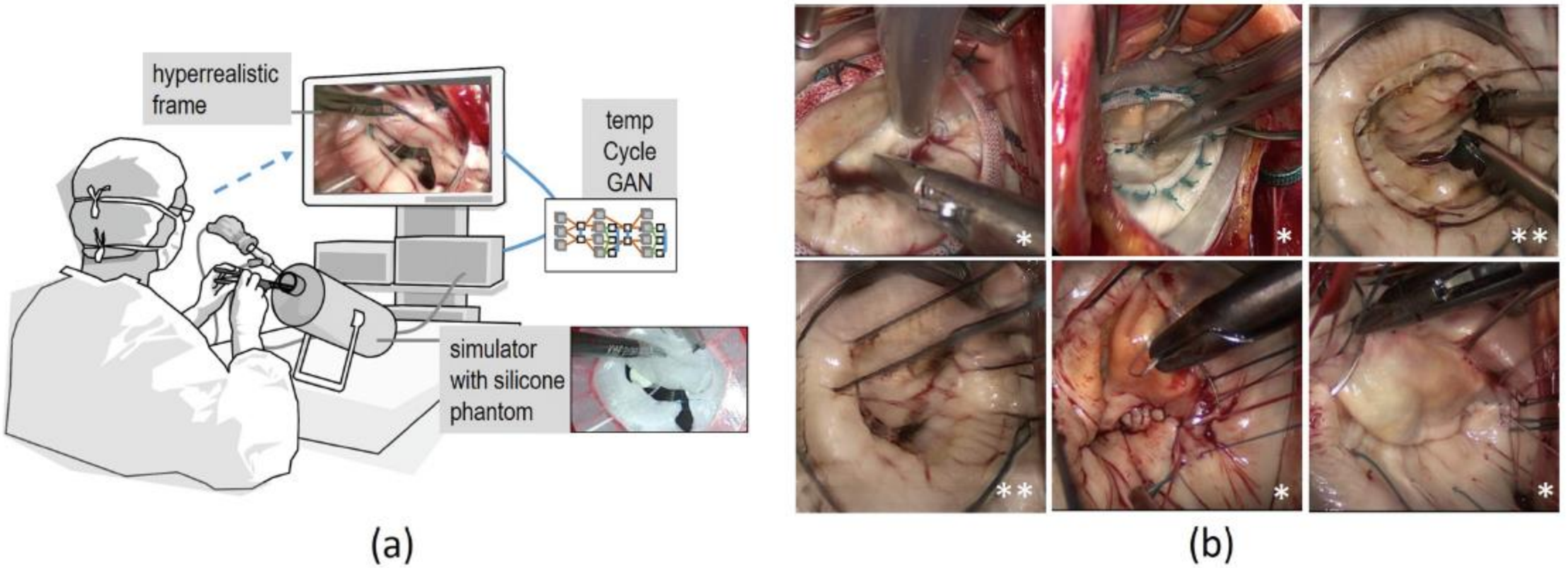


Fig. 1. (a) Vision: Augmentation of the minimally invasive training process with real-time generated *hyperrealistic* frames. (b) Visual comparison of real intraoperative frames from mitral valve surgery (*) and generated fake images (**).

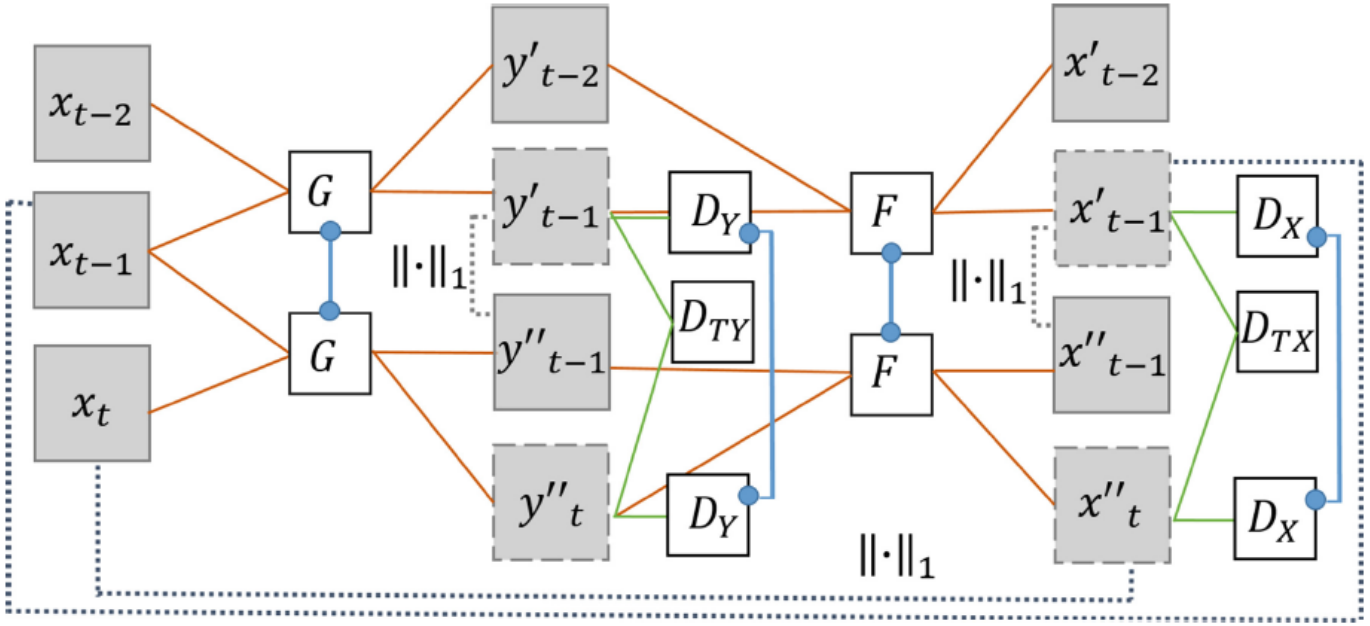


Fig. 2. Training setup of the $X \rightarrow Y \rightarrow X$ cycle of the proposed tempCycleGAN network (reverse cycle accordingly) using temporal pairs: the generators G, F and the temporal discriminators $D_{T\{X,Y\}}$ take the current frame and a single preceding frame. Each run of G (and F) synthesizes outputs for both frames. In the application of the generator, the frame of interest is the second output (y'_{t-1} and y''_t, respectively). The temporal discriminators are trained on these frames of interest, thus, the generator G needs to run twice to generate the two frames of interest (y'_{t-1} and y''_t) for D_{TY} (for F and D_{TX} accordingly). L1-distances (dotted lines) between matching time frames are used in the loss function to further enforce time consistency. $D_{\{X,Y\}}$: discriminators with 1 input; blue connections: shared weights.

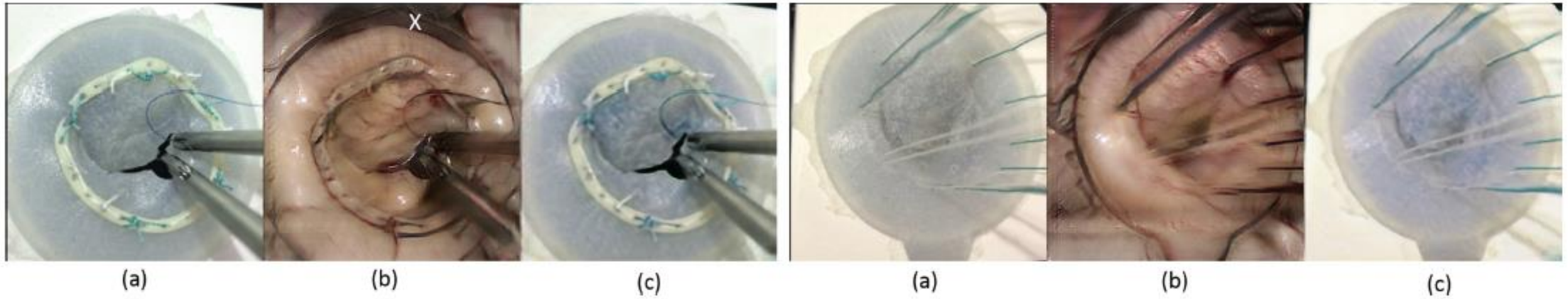


Fig. 3. tempCycleGAN results for two examples shown left and right, where (a) shows the real phantom x_t, (b) shows corresponding synthesized intraoperative images y_t'' and (c) shows the re-synthesized phantom image x_t''. X marks a synthesized atrial retractor.

Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration

Adrian V. Dalca[1,2,3(✉)], Guha Balakrishnan[1], John Guttag[1], and Mert R. Sabuncu[3]

[1] Computer Science and Artificial Intelligence Lab, MIT, Cambridge, USA
adalca@mit.edu
[2] Martinos Center for Biomedical Imaging, Massachusetts General Hospital, HMS, Charlestown, USA
[3] School of Electrical and Computer Engineering, Cornell University, Ithaca, USA

Abstract. Traditional deformable registration techniques achieve impressive results and offer a rigorous theoretical treatment, but are computationally intensive since they solve an optimization problem for each image pair. Recently, learning-based methods have facilitated fast registration by learning spatial deformation functions. However, these approaches use restricted deformation models, require supervised labels, or do not guarantee a diffeomorphic (topology-preserving) registration. Furthermore, learning-based registration tools have not been derived from a probabilistic framework that can offer uncertainty estimates. In this paper, we present a probabilistic generative model and derive an unsupervised learning-based inference algorithm that makes use of recent developments in convolutional neural networks (CNNs). We demonstrate our method on a 3D brain registration task, and provide an empirical analysis of the algorithm. Our approach results in state of the art accuracy and very fast runtimes, while providing diffeomorphic guarantees and uncertainty estimates. Our implementation is available online at http://voxelmorph.csail.mit.edu.

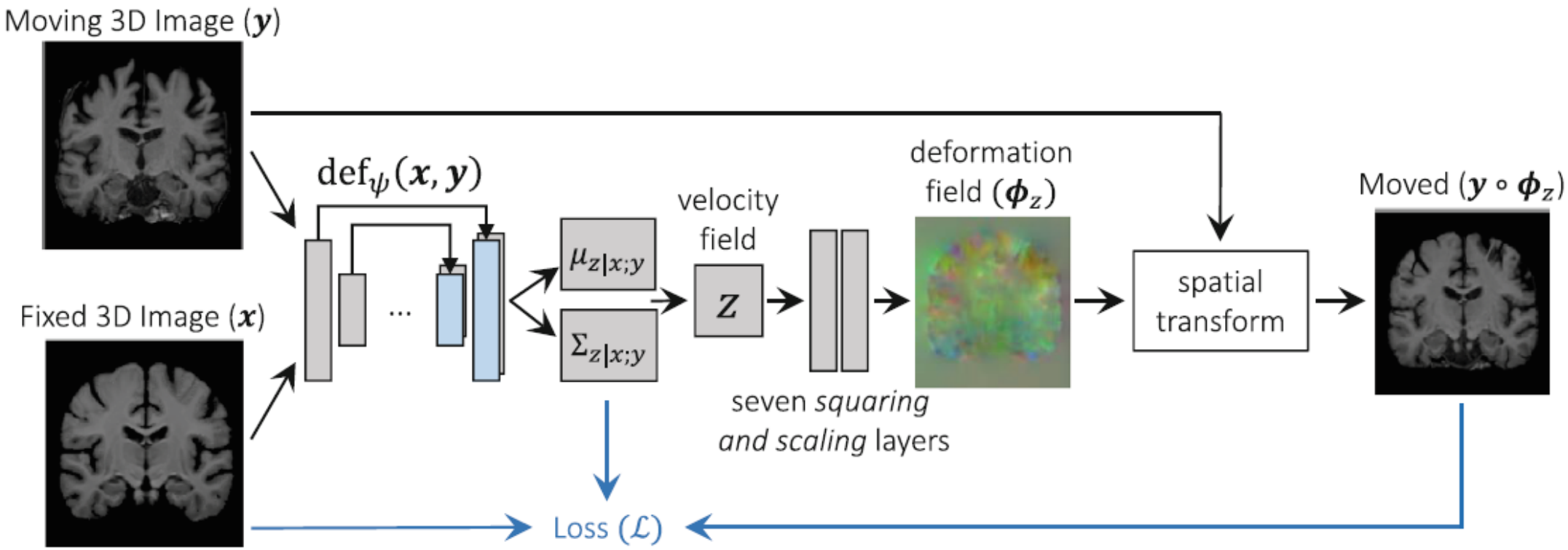


Fig. 1. Overview of end-to-end unsupervised architecture. The first part of the network, $\text{def}_\psi(\boldsymbol{x}, \boldsymbol{y})$ takes the input images and outputs the approximate posterior probability parameters representing the velocity field mean, $\boldsymbol{\mu}_{z|x;y}$, and variance, $\boldsymbol{\Sigma}_{z|x;y}$. A velocity field $\boldsymbol{z}$ is sampled and transformed to a diffeomorphic deformation field $\boldsymbol{\phi}_z$ using novel differentiable *squaring and scaling* layers. Finally, a spatial transform warps $\boldsymbol{y}$ to obtain $\boldsymbol{y} \circ \boldsymbol{\phi}_z$.

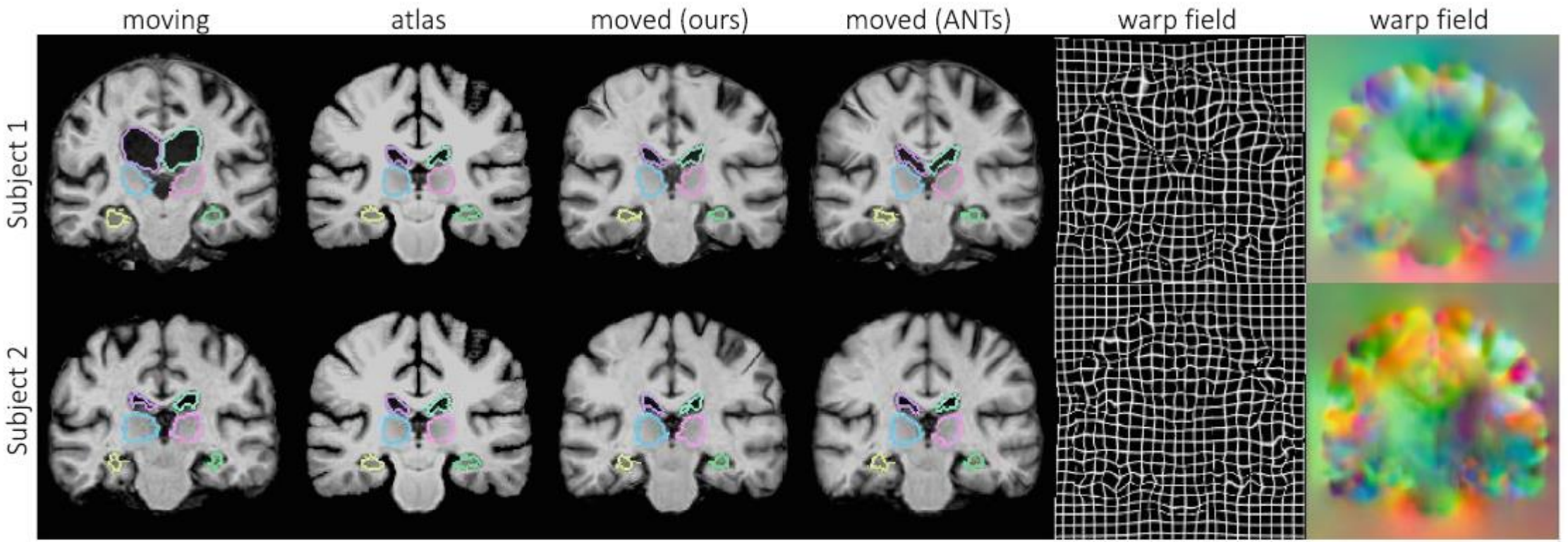


Fig. 2. Example MR slices of input moving image, atlas, and resulting warped image for our method and ANTs, with overlaid boundaries of ventricles, thalami and hippocampi. Our resulting registration field is shown as a warped grid and RGB image, with each channel representing dimension. Due to space constraints, we omit VoxelMorph examples, which are *visually* similar to our results and ANTs.

Deep Convolutional Gaussian Mixture Model for Stain-Color Normalization of Histopathological Images

Farhad Ghazvinian Zanjani(✉), Svitlana Zinger, and Peter H. N. de With

Eindhoven University of Technology, 5612 AJ Eindhoven, The Netherlands
f.ghazvinian.zanjani@tue.nl

Abstract. Automated microscopic analysis of stained histopathological images is degraded by the amount of color and intensity variations in data. This paper presents a novel unsupervised probabilistic approach by integrating a convolutional neural network (CNN) and the Gaussian mixture model (GMM) in a unified framework, which jointly optimizes the modeling and normalizing the color and intensity of hematoxylin- and eosin-stained (H&E) histological images. In contrast to conventional GMM-based methods that are applied only on the color distribution of data for stain color normalization, our proposal learns how to cluster the tissue structures according to their shape and appearance and simultaneously fits a multivariate GMM to the data. This approach is more robust than standard GMM in the presence of strong staining variations because fitting the GMM is conditioned on the appearance of tissue structures in the density channel of an image. Performing a gradient descent optimization in an end-to-end learning, the network learns to maximize the log-likelihood of data given estimated parameters of multivariate Gaussian distributions. Our method does not need ground truth, shape and color assumptions of image contents or manual tuning of parameters and thresholds which makes it applicable to a wide range of histopathological images. Experiments show that our proposed method outperforms the state-of-the-art algorithms in terms of achieving a higher color constancy.

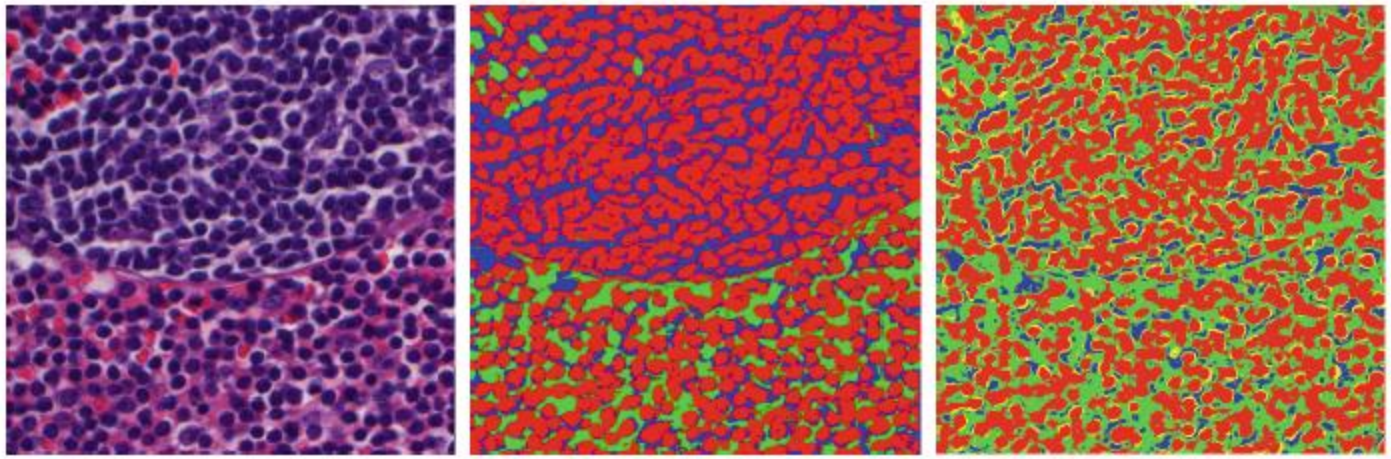


Fig. 1. Example of standard GMM failure in tissue clustering using HSD color space; (a) RGB H&E image; (b) 3-class standard GMM; (c) 3-class DCGMM.

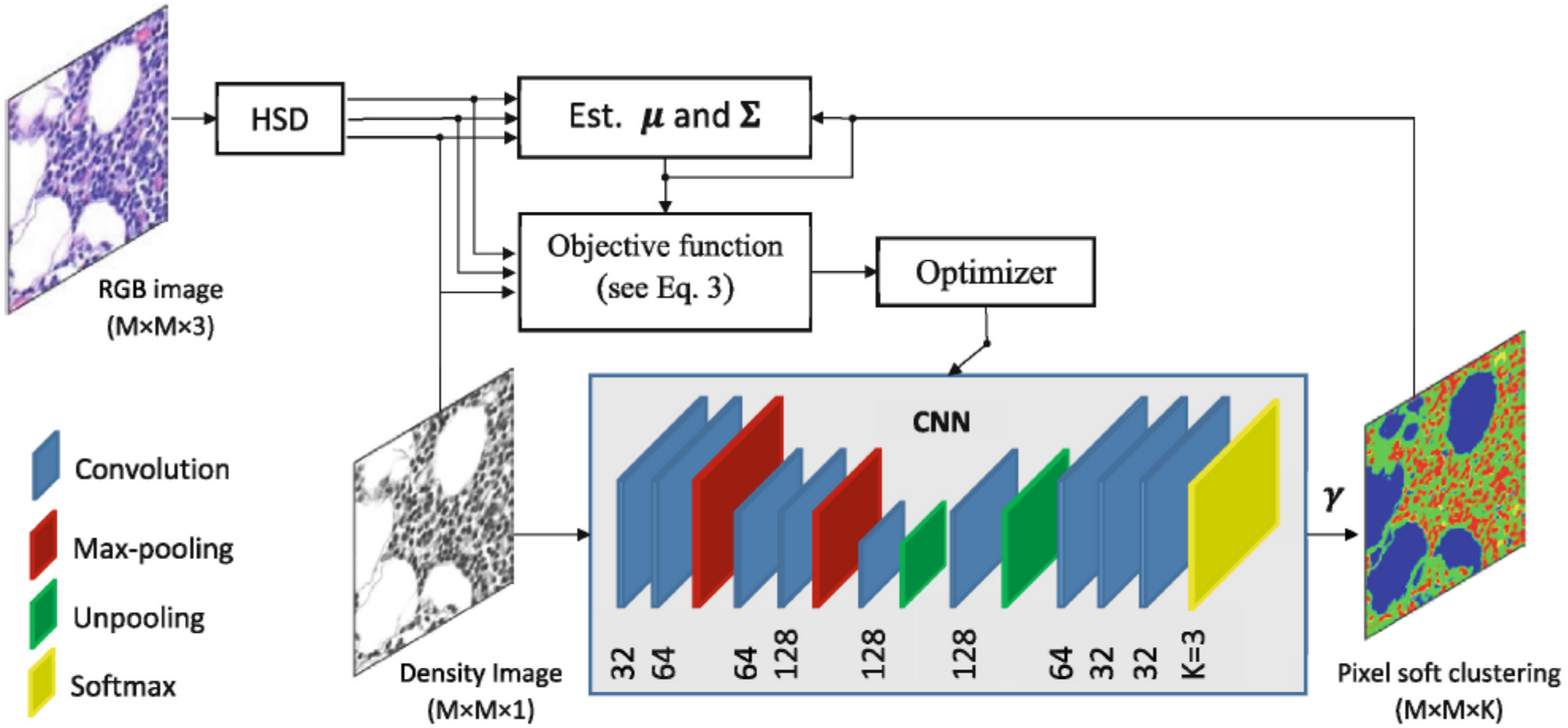


Fig. 2. Block diagram of DCGMM in training phase.

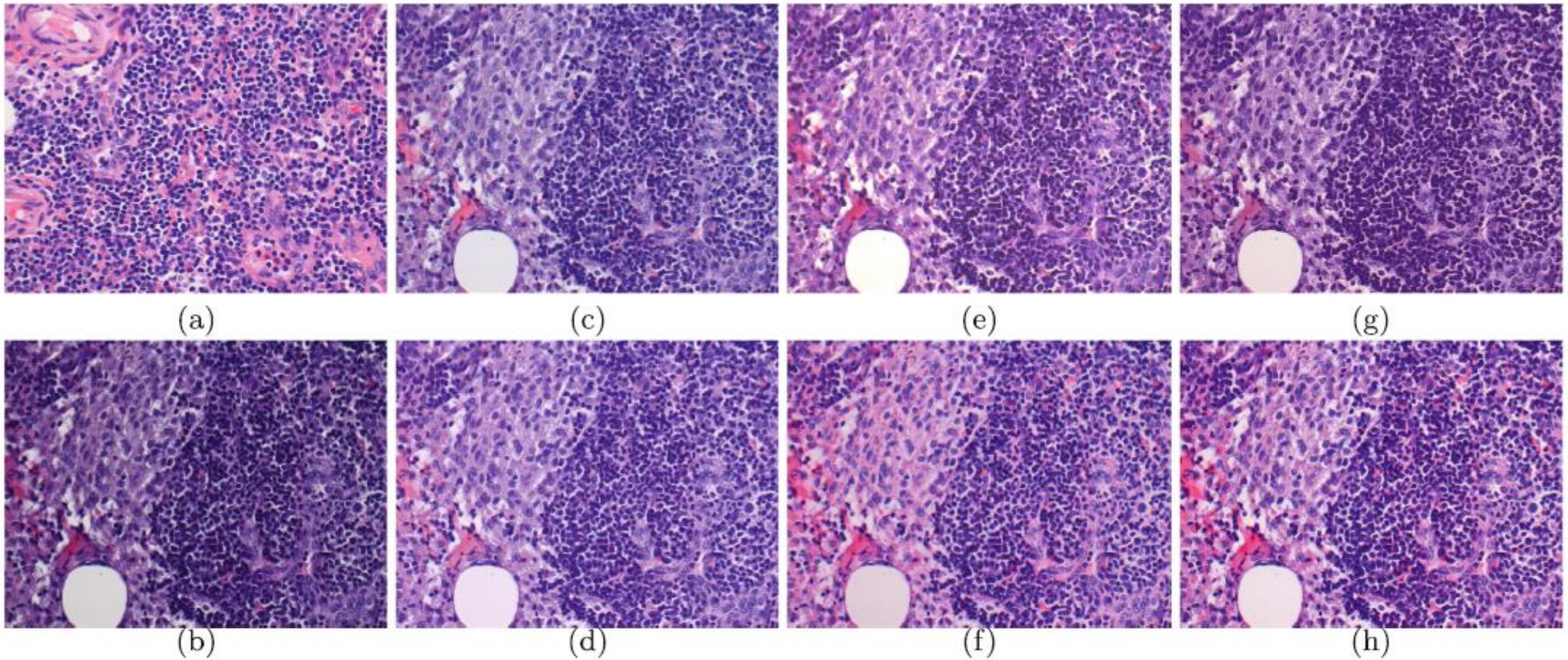


Fig. 3. Performance of different stain color-normalization methods on an $H\&E$ image. (a) template image (b) original images, (c) Macenko *et al.*[6], (d) Vahadane *et al.*[7], (e) Reinhard *et al.*[5], (f) Bejnordi *et al.*[1], (g) Khan *et al.* [4], (h) DCGMM.

Towards Automated Colonoscopy Diagnosis: Binary Polyp Size Estimation via Unsupervised Depth Learning

Hayato Itoh[1(✉)], Holger R. Roth[1,2], Le Lu[2], Masahiro Oda[1], Masashi Misawa[3], Yuichi Mori[3], Shin-ei Kudo[3], and Kensaku Mori[1,4,5]

[1] Graduate School of Informatics, Nagoya University, Nagoya, Japan
hitoh@mori.m.is.nagoya-u.ac.jp
[2] AI-Infra, NVIDIA Corporation, Santa Clara, USA
[3] Digestive Disease Center, Showa University Northern Yokohama Hospital, Yokohama, Japan
[4] Information Technology Center, Nagoya University, Nagoya, Japan
[5] Research Center for Medical Bigdata, National Institute of Informatics, Tokyo, Japan

Abstract. In colon cancer screening, polyp size estimation using only colonoscopy images or videos is difficult even for expert physicians although the size information of polyps is important for diagnosis. Towards the fully automated computer-aided diagnosis (CAD) pipeline, a robust and precise polyp size estimation method is highly desired. However, the size estimation problem of a three-dimensional object from a two-dimensional image is ill-posed due to the lack of three-dimensional spatial information. To circumvent this challenge, we formulate a relaxed form of size estimation as a binary classification problem and solve it by a new deep neural network architecture: BseNet. This relaxed form of size estimation is defined as a two-category classification: under and over a certain polyp dimension criterion that would provoke different clinical treatments (resecting the polyp or not). BseNet estimates the depth map image from an input colonoscopic RGB image using unsupervised deep learning, and integrates RGB with the computed depth information to produce a four-channel RGB-D imagery data, that is subsequently encoded by BseNet to extract deep RGB-D image features and facilitate the size classification into two categories: under and over 10 mm polyps. For the evaluation of BseNet, a large dataset of colonoscopic videos of totally over 16 h is constructed. We evaluate the accuracies of both binary polyp size estimation and polyp detection performance since detection is a prerequisite step of a fully automated CAD system. The experimental results show that our proposed BseNet achieves 79.2 % accuracy for binary polyp-size classification. We also combine the image feature extraction by BseNet and classification of short video clips using a long short-term memory (LSTM) network. Polyp detection (if the video clip contains a polyp or not) shows 88.8 % sensitivity when employing the spatio-temporal image feature extraction and classification.

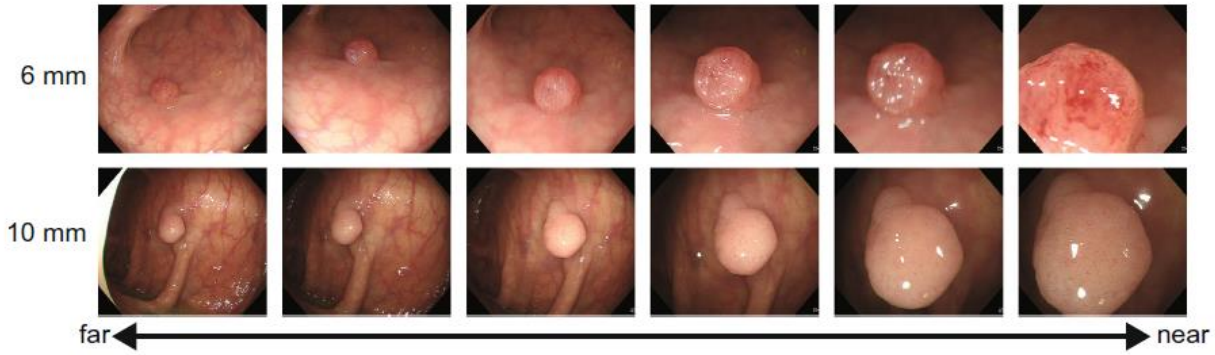


Fig. 1. Examples of polyps on colonoscopic images. Top and bottom rows show images that capture polyps with diameters of 6 mm and 10 mm, respectively. From left to right, columns show images with different (long to short) distances from colonoscope to polyps.

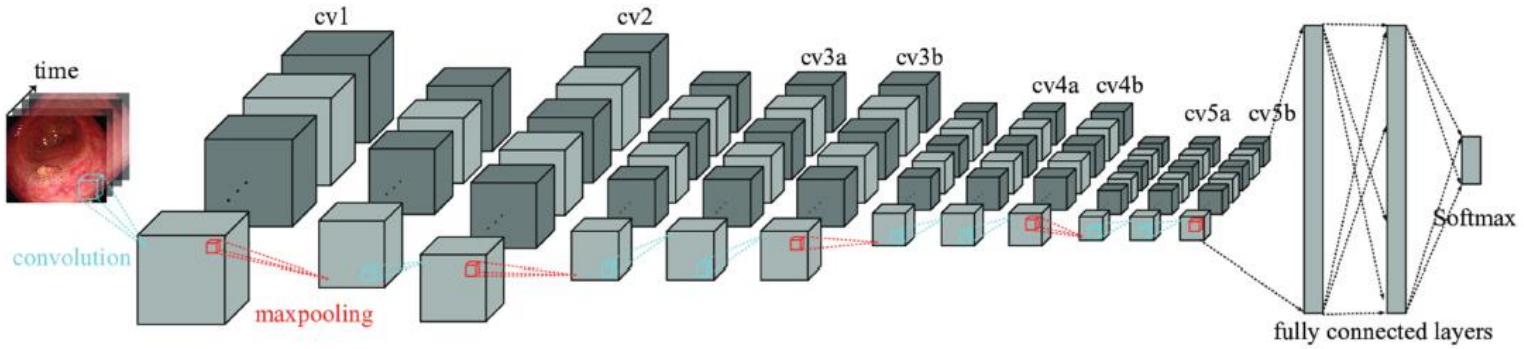


Fig. 2. Architecture of spatio-temporal classification for polyp detection. C3dNet extracts deep image spatial-temporal features via 3D convolutional and pooling procedures.

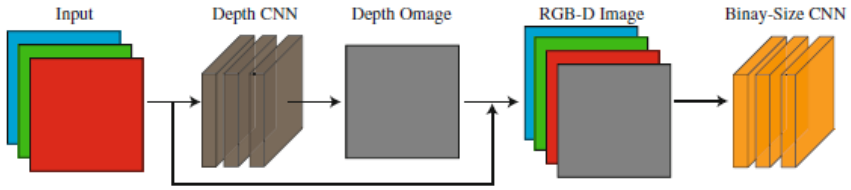


Fig. 3. Architecture of the binary polyp size estimation network (BseNet). BseNet first estimates the depth map from an RGB colonoscopic image by employing depth CNN. The estimated depth image is then combined with the input RGB channels to form an RGB-D image. BsdNet then classifies the newly composite RGB-D image into two categories: polyp over and under 10 mm in diameter.

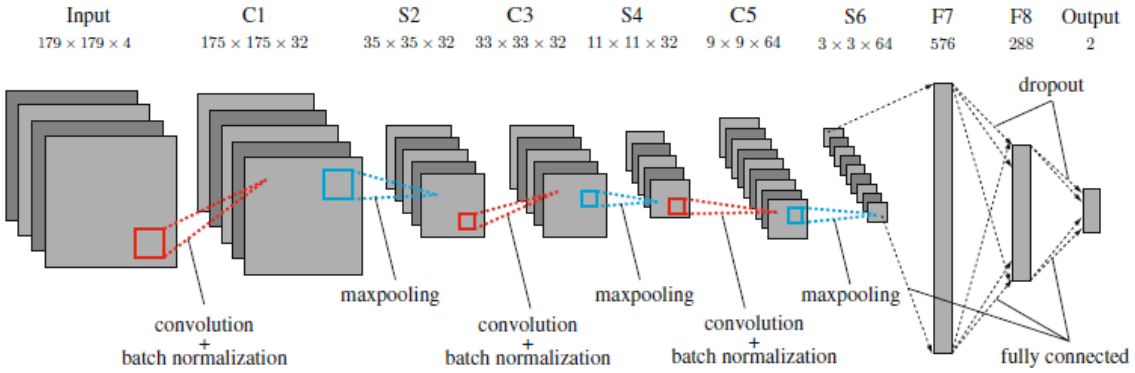


Fig. 4. Architecture of RGB-D CNN. Input is an RGB-D image of four channels.