Unsupervised Learning

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning

Outline

Outline

Unsupervised Learning

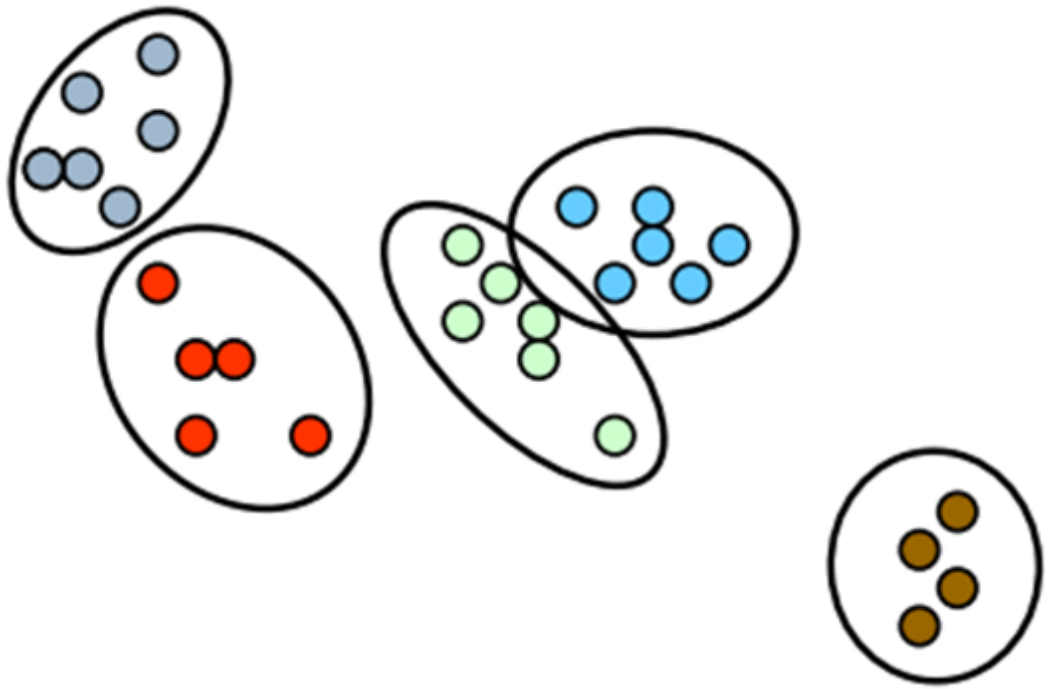
- Dataset: $\mathbb{X} = \{\boldsymbol{x}^{(i)}\}_i$
  - No supervision

Unsupervised Learning

- Dataset: $\mathbb{X} = \{\boldsymbol{x}^{(i)}\}_i$
  - No supervision
- What can we learn?

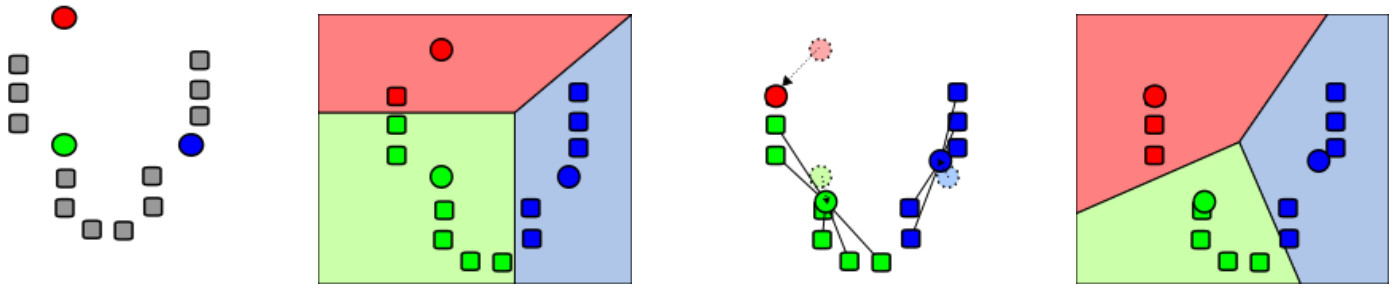
Clustering I

- Goal: to group similar $\boldsymbol{x}^{(i)}$'s



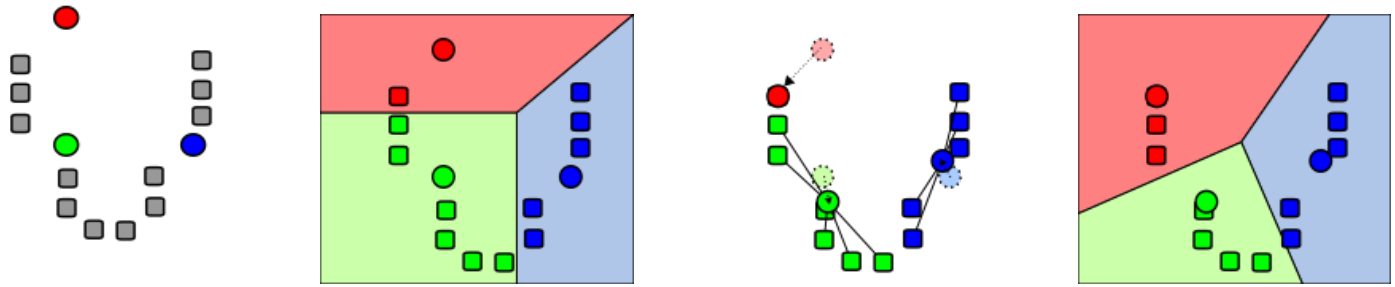
Clustering II

- K-means algorithm:

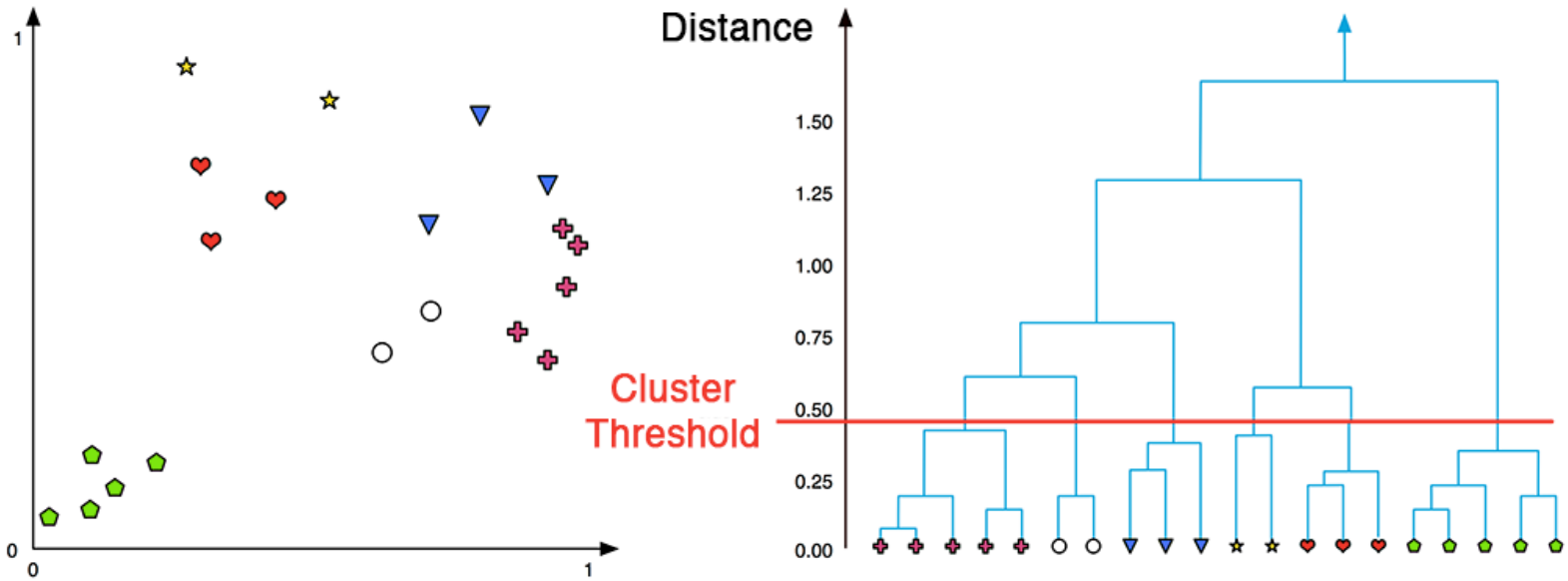


Clustering II

- K-means algorithm:

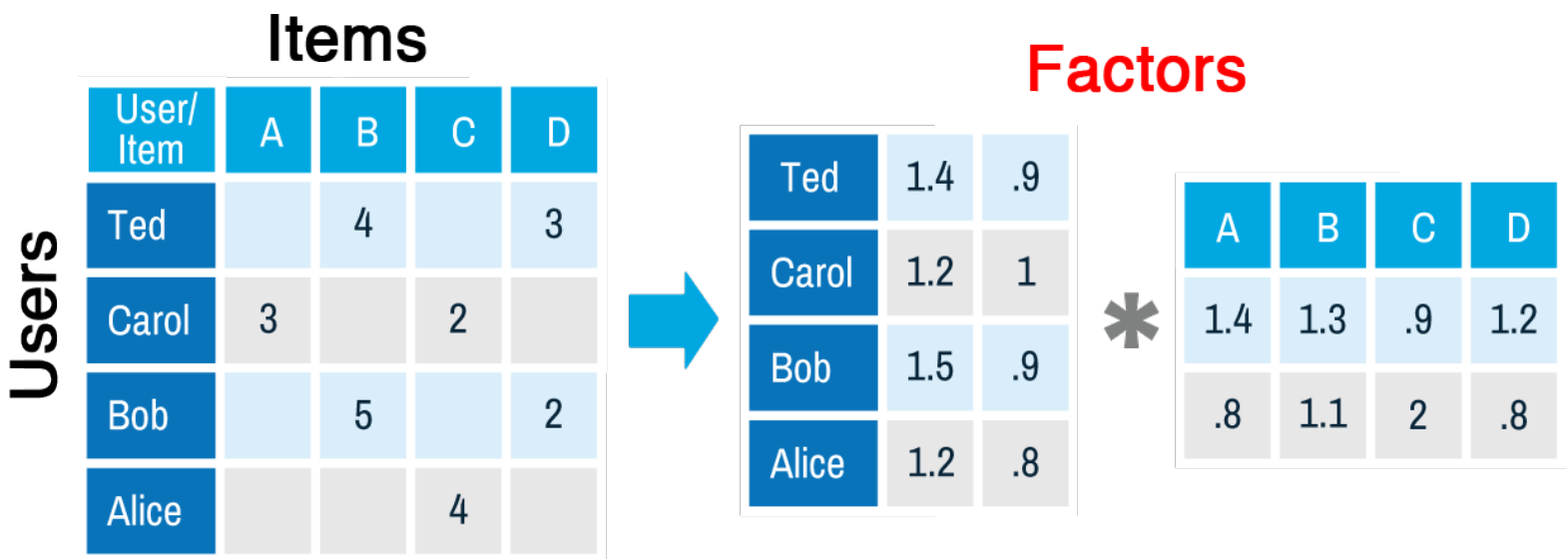


- Hierarchical clustering:



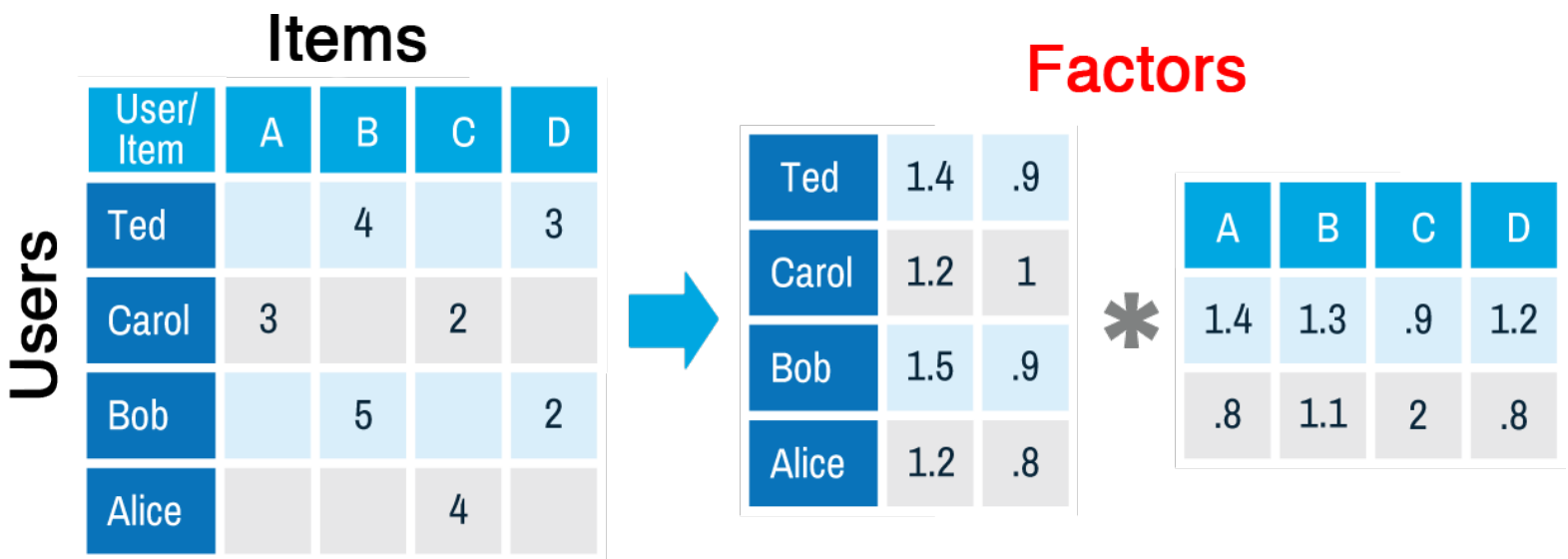
Factorization and Recommendation

- Goal: to uncover the ***factors*** behind data (rating matrix)



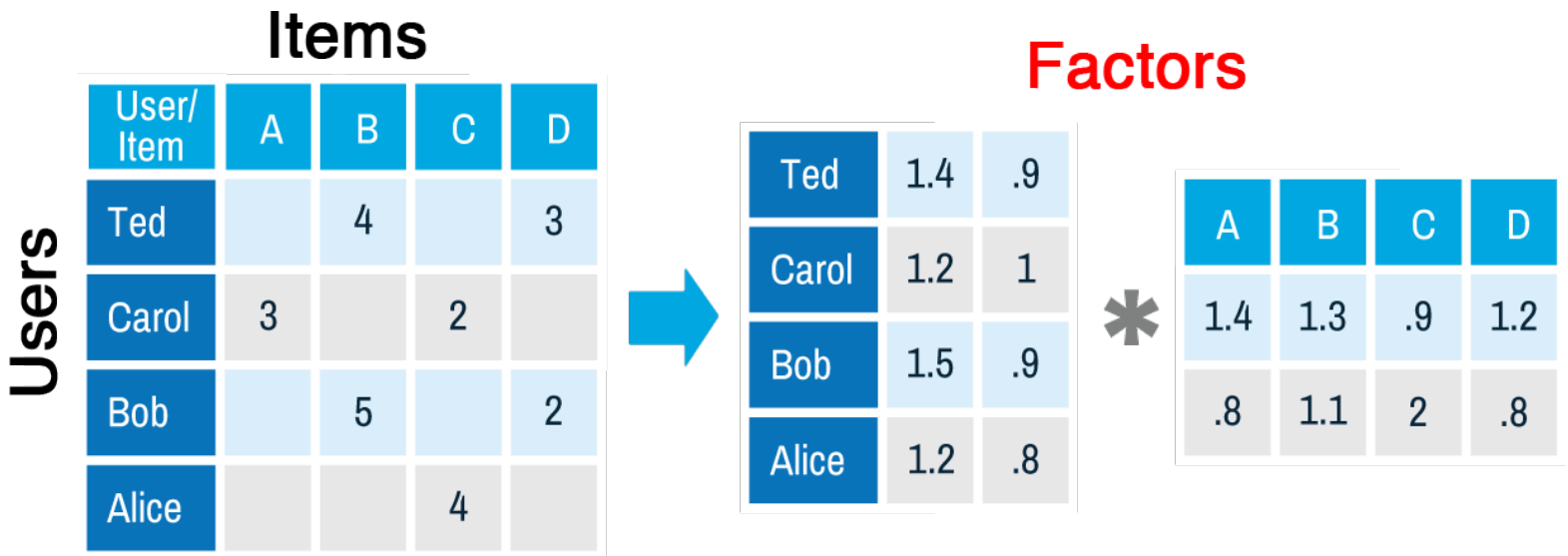
Factorization and Recommendation

- Goal: to uncover the ***factors*** behind data (rating matrix)
  - Commonly used in the recommender systems



Factorization and Recommendation

- Goal: to uncover the ***factors*** behind data (rating matrix)
  - Commonly used in the recommender systems



- Non-negative matrix factorization (NMF) [9, 10]

Dimension Reduction

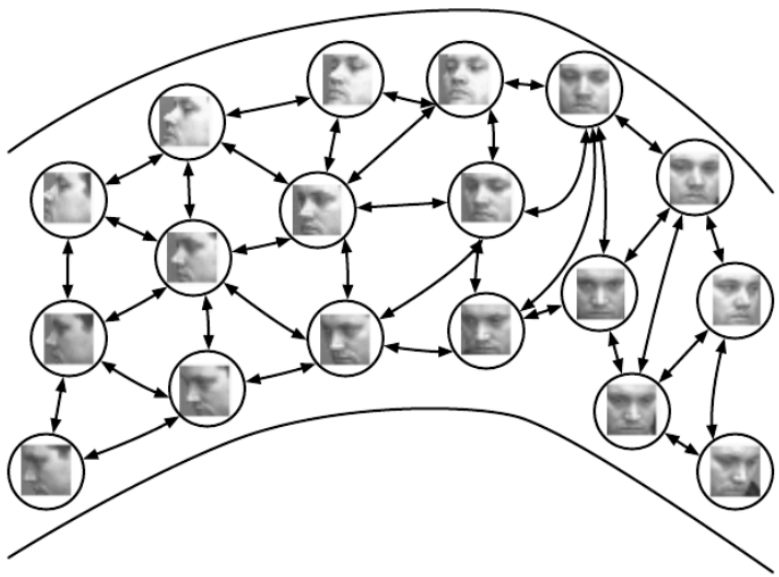
- Goal: to reduce the dimension of each $\boldsymbol{x}^{(i)}$
  - E.g., PCA

Dimension Reduction

- Goal: to reduce the dimension of each $\boldsymbol{x}^{(i)}$
  - E.g., PCA
- ***Predictive learning***
  - Learn to "fill in the blanks"

Dimension Reduction

- Goal: to reduce the dimension of each $\boldsymbol{x}^{(i)}$
  - E.g., PCA
- ***Predictive learning***
  - Learn to "fill in the blanks"
- ***Manifold learning***
  - Learn tangent vectors of a given point

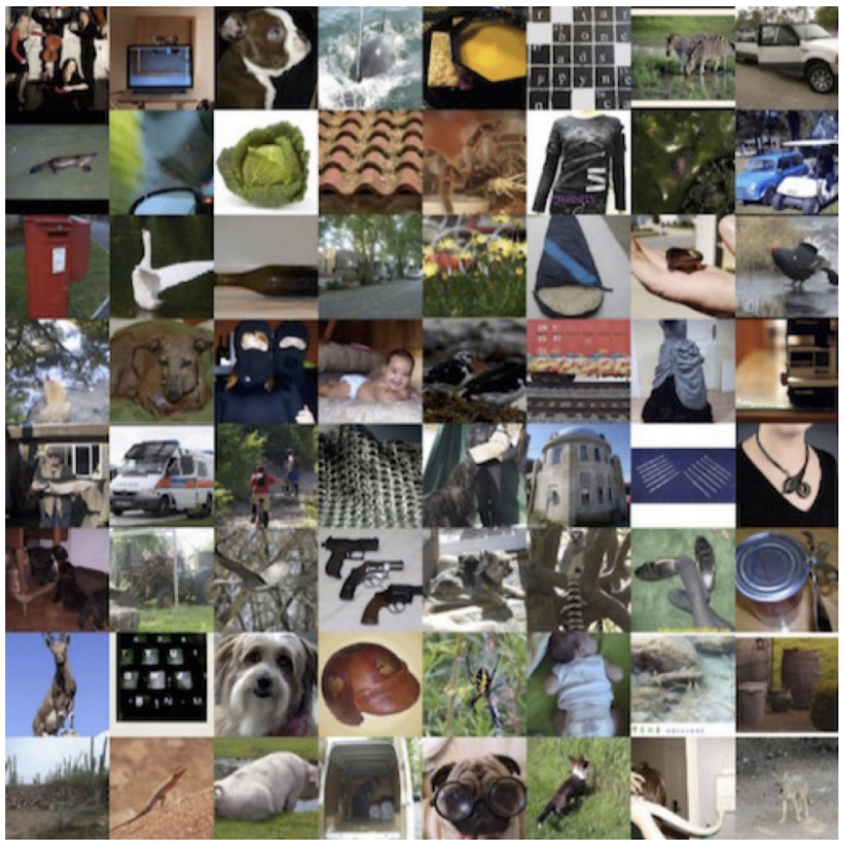


Data Generation I

- Goal: to generate new data points/samples

Data Generation I

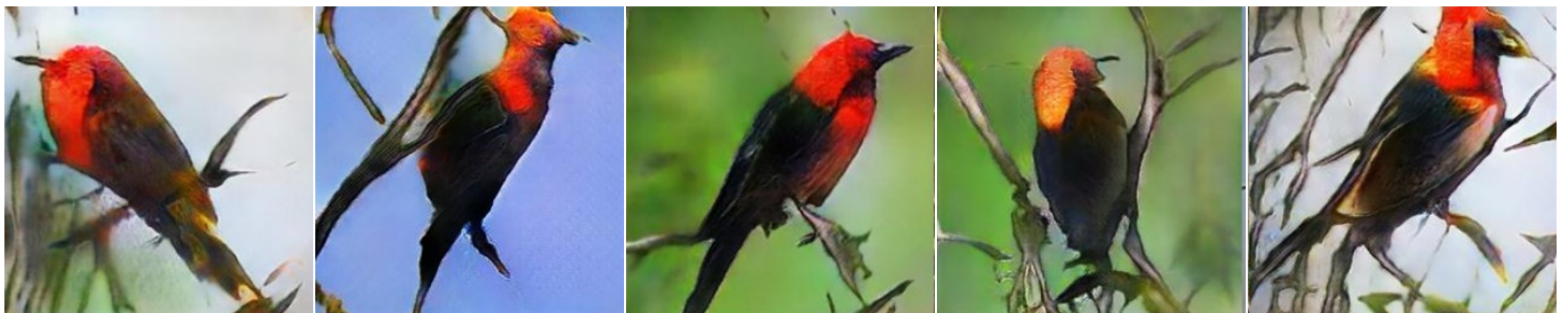
- Goal: to generate new data points/samples
- *Generative adversarial networks* (GANs)



Data Generation II

- Text to image based on conditional GANs:

  "*This bird is completely red with black wings and pointy beak.*"



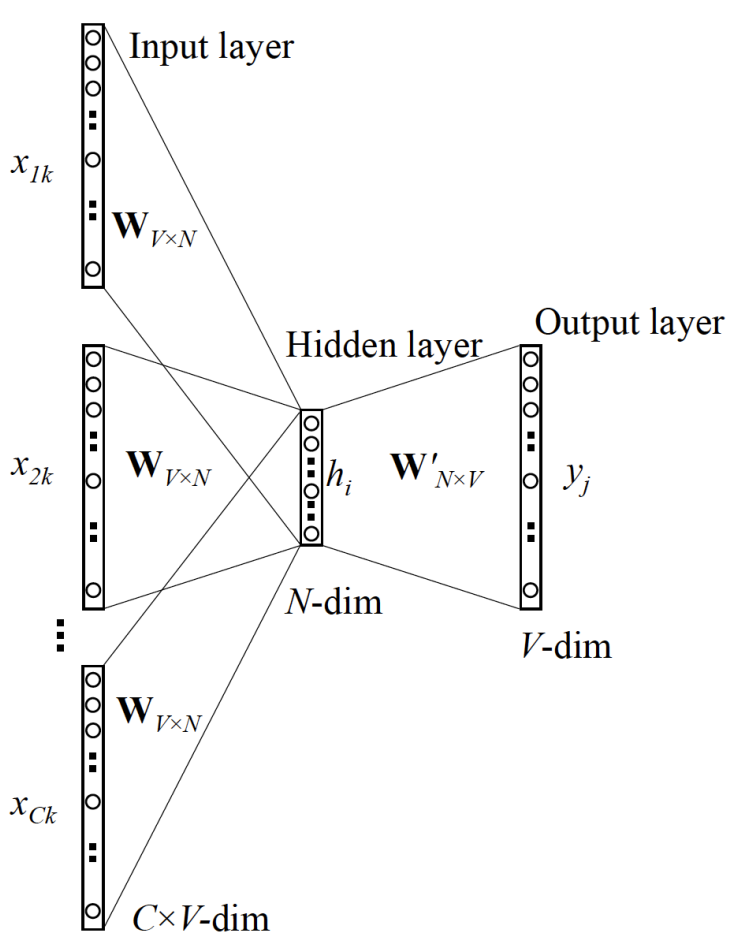
Outline

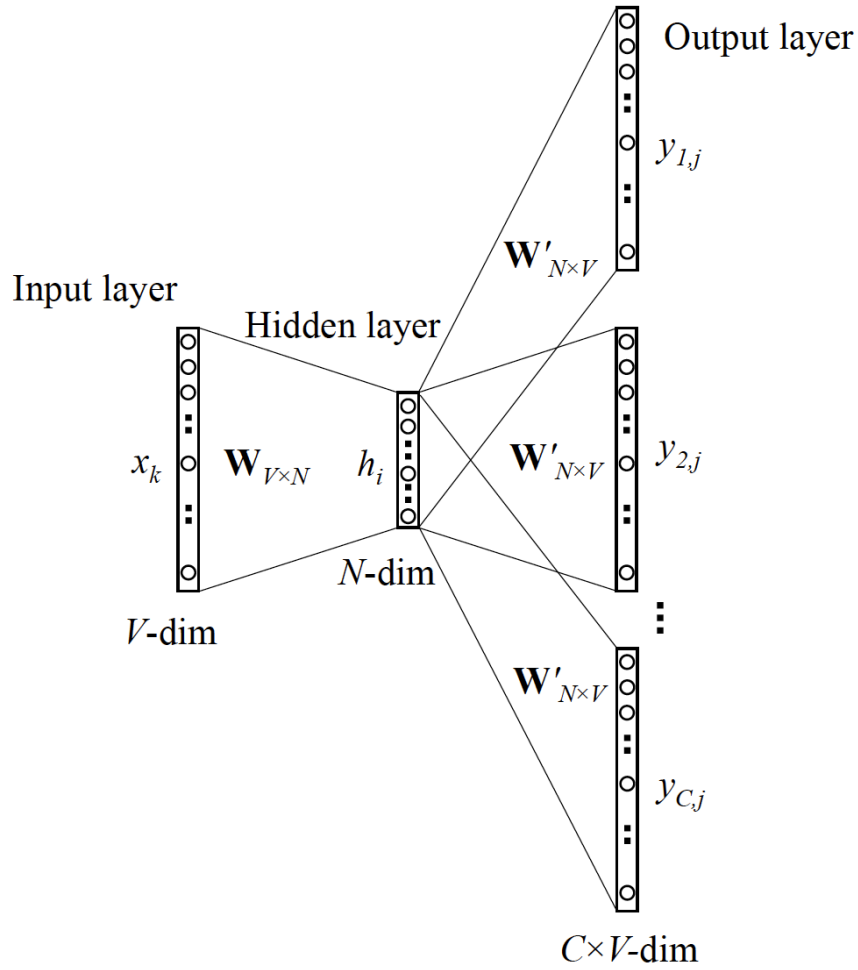
Predictive Learning

- I.e., blank filling

Predictive Learning

- I.e., blank filling
- E.g., word2vec [13, 12]: "*... the cat sat* ***on****...*"





Doc2Vec

- How to encode a document?

Doc2Vec

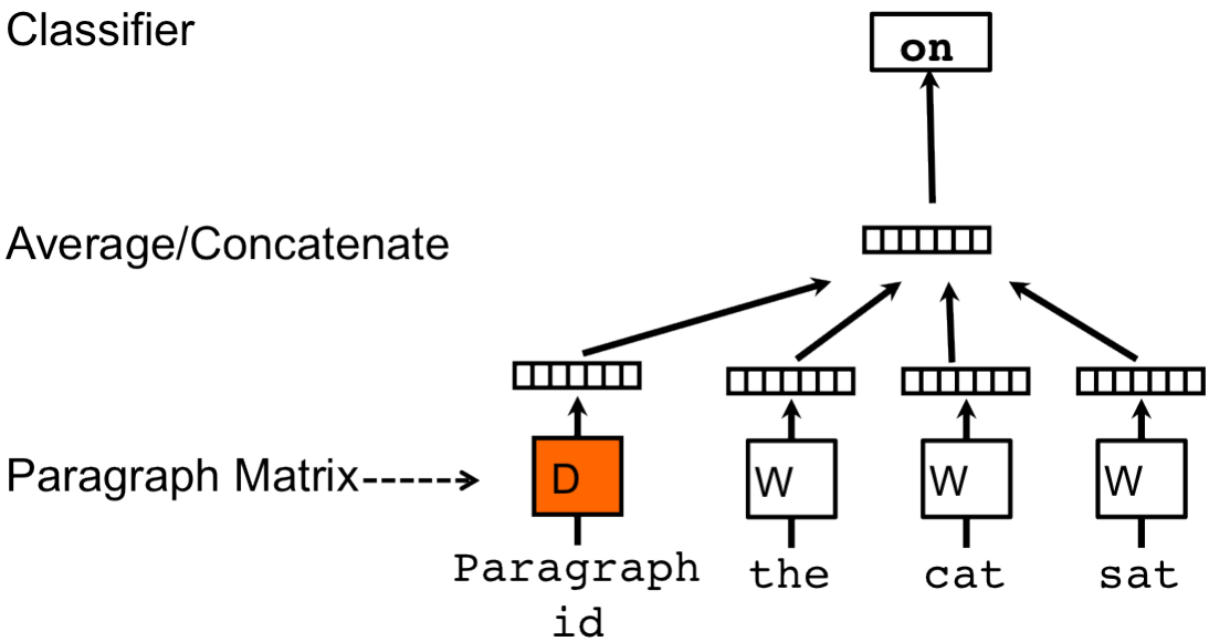
- How to encode a document?
  - Bag of words (TF-IDF), average word2vec, etc.

Doc2Vec

- How to encode a document?
  - Bag of words (TF-IDF), average word2vec, etc.
- Do ***not*** capture the semantic meaning of a doc
  - "*I like final project*" $\neq$ "*Final project likes me*"
- Predictive learning for docs?

Doc2Vec

- How to encode a document?
  - Bag of words (TF-IDF), average word2vec, etc.
- Do ***not*** capture the semantic meaning of a doc
  - "*I like final project*" $\neq$ "*Final project likes me*"
- Predictive learning for docs?
- Doc2vec [7]: to capture the ***context*** not explained by words



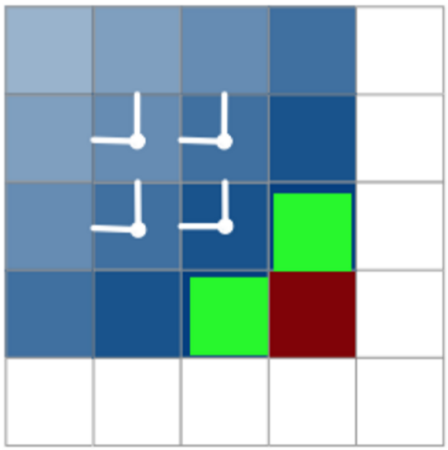
Filling Images

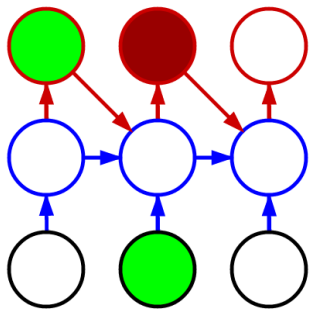
- How?

Filling Images

- How?

- PixelRNN [19]





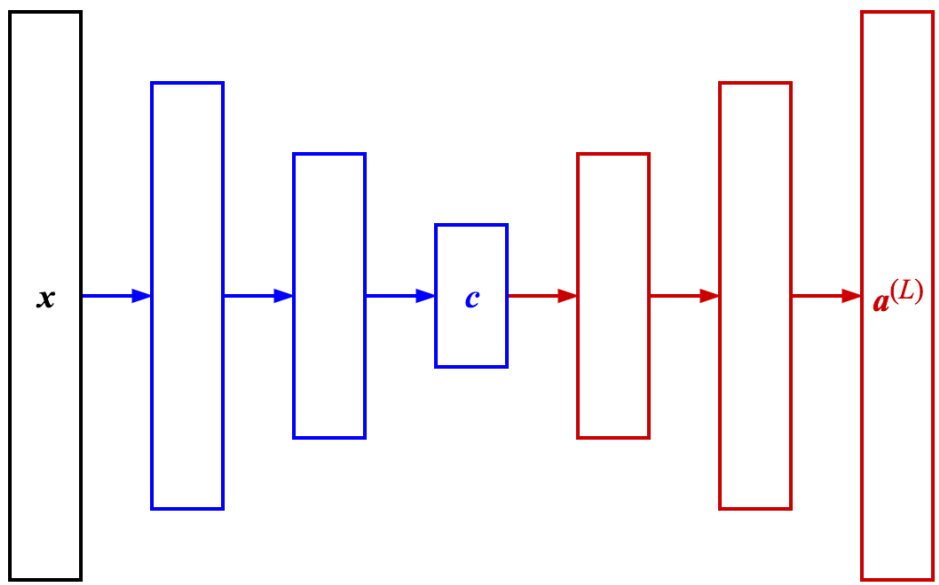
More

- Predicting the future by watching unlabeled videos [6, 21]:

Outline

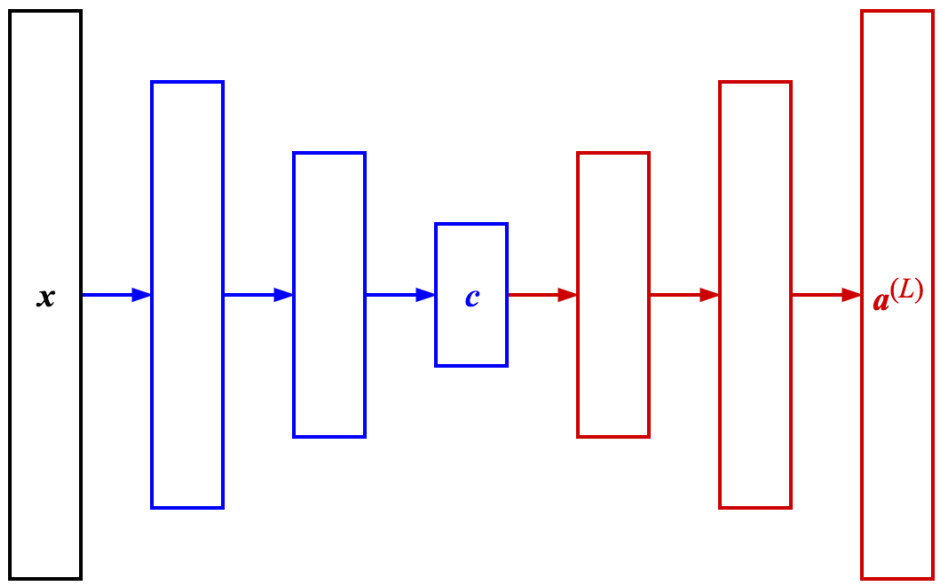
Autoencoders I

- Encoder: to learn a low dimensional representation $\mathbf{c}$ (called ***code***) of input $\mathbf{x}$
- Decoder: to reconstruct $\mathbf{x}$ from $\mathbf{c}$



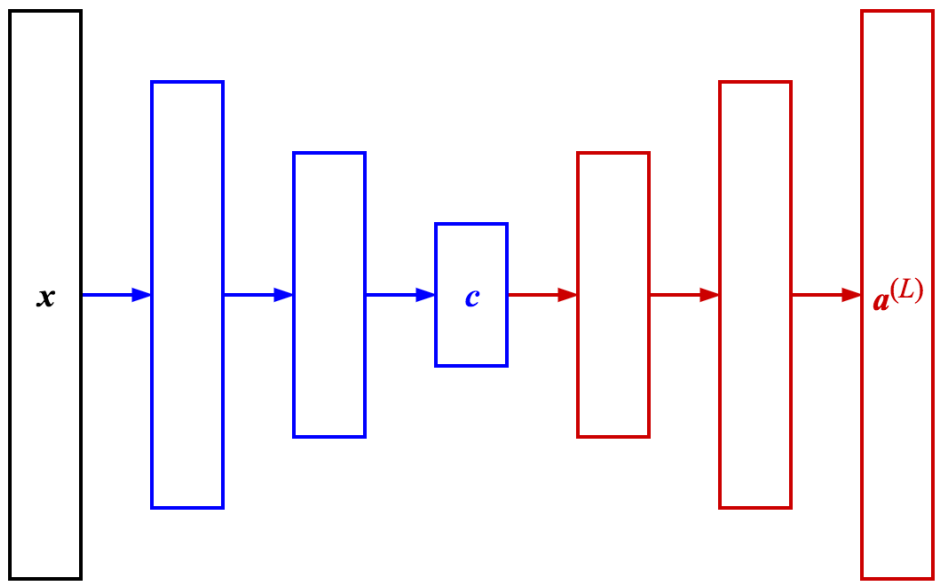
Autoencoders I

- Encoder: to learn a low dimensional representation $\mathbf{c}$ (called ***code***) of input $\mathbf{x}$
- Decoder: to reconstruct $\mathbf{x}$ from $\mathbf{c}$
- Cost function: $\arg\min_{\Theta} -\log \mathrm{P}(\mathbb{X} \mid \Theta) = \arg\min_{\Theta} -\sum_n \log \mathrm{P}(\boldsymbol{x}^{(n)} \mid \Theta)$



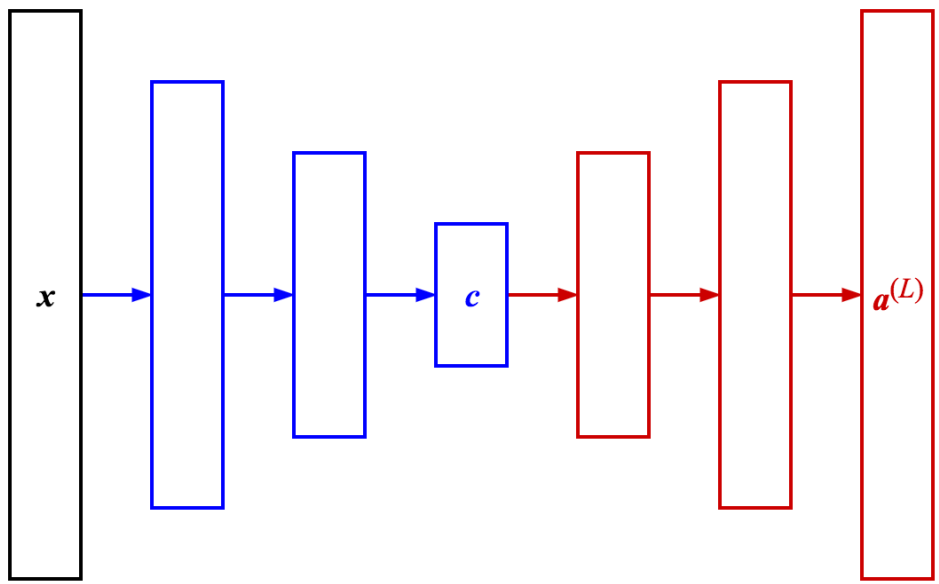
Autoencoders I

- Encoder: to learn a low dimensional representation $\mathbf{c}$ (called ***code***) of input $\mathbf{x}$
- Decoder: to reconstruct $\mathbf{x}$ from $\mathbf{c}$
- Cost function: $\arg\min_{\Theta} -\log \mathrm{P}(\mathbb{X} \mid \Theta) = \arg\min_{\Theta} -\sum_n \log \mathrm{P}(\boldsymbol{x}^{(n)} \mid \Theta)$
- Sigmoid output units $a_j^{(L)} = \hat{\rho}_j$ for $\mathrm{x}_j \sim \mathrm{Bernoulli}(\rho_j)$
  - $\mathrm{P}(x_j^{(n)} \mid \Theta) = (a_j^{(L)})^{x_j^{(n)}} (1 - a_j^{(L)})^{(1 - x_j^{(n)})}$



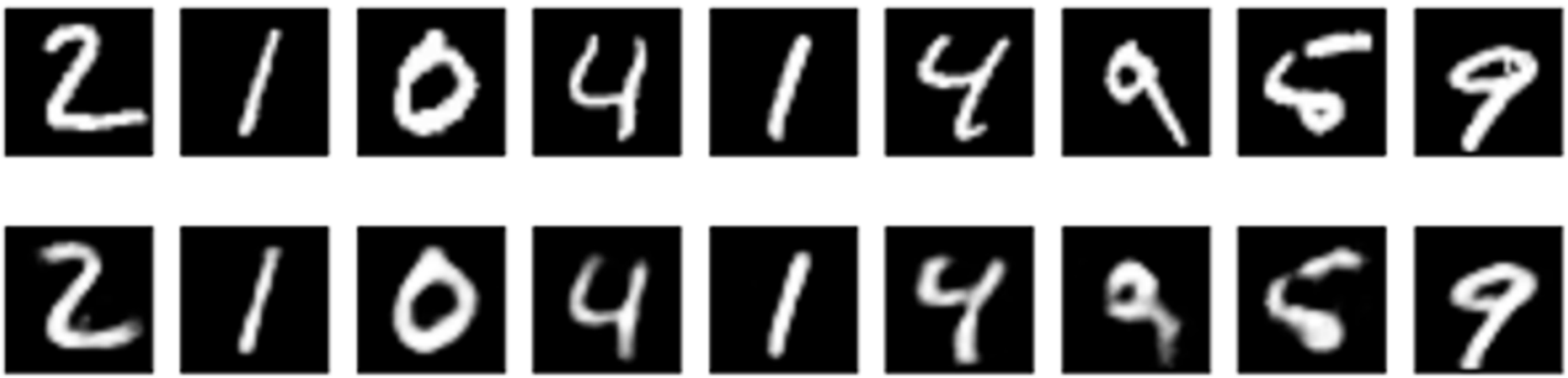
Autoencoders I

- Encoder: to learn a low dimensional representation $\mathbf{c}$ (called ***code***) of input $\mathbf{x}$
- Decoder: to reconstruct $\mathbf{x}$ from $\mathbf{c}$
- Cost function: $\arg\min_\Theta -\log \mathrm{P}(\mathbb{X}\,|\,\Theta) = \arg\min_\Theta -\sum_n \log \mathrm{P}(\boldsymbol{x}^{(n)}\,|\,\Theta)$
- Sigmoid output units $a_j^{(L)} = \hat{\rho}_j$ for $\mathrm{x}_j \sim \mathrm{Bernoulli}(\rho_j)$
  - $\mathrm{P}(x_j^{(n)}\,|\,\Theta) = (a_j^{(L)})^{x_j^{(n)}}(1-a_j^{(L)})^{(1-x_j^{(n)})}$
- Linear output units $\boldsymbol{a}^{(L)} = \boldsymbol{z}^{(L)} = \hat{\mu}$ for $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$
  - $-\log \mathrm{P}(\boldsymbol{x}^{(n)}\,|\,\Theta) = \|\boldsymbol{x}^{(n)} - \boldsymbol{z}^{(L)}\|^2$



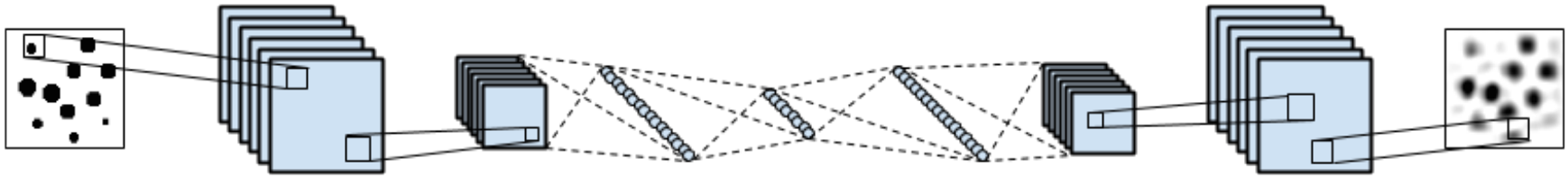
Autoencoders II

- A 32-bit code can roughly represents a 32×32 MNIST image



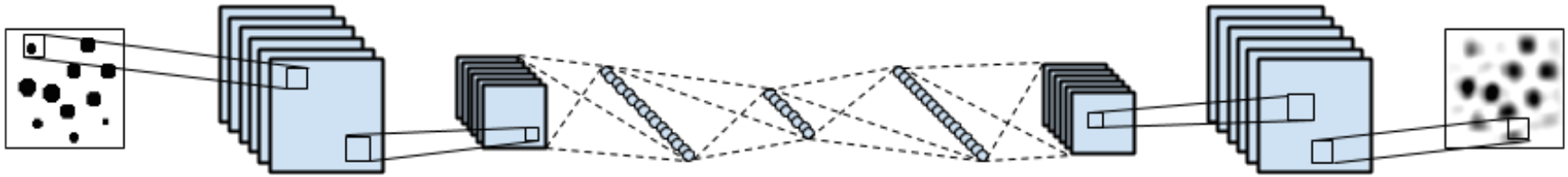
Convolutional Autoencoders

- Convolution + deconvolution:



Convolutional Autoencoders

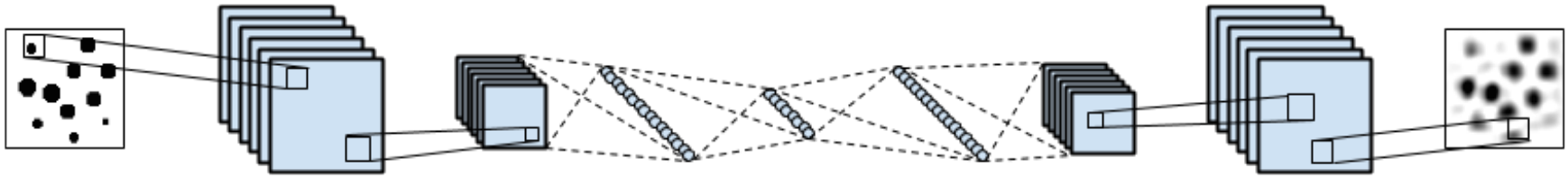
- Convolution + deconvolution:



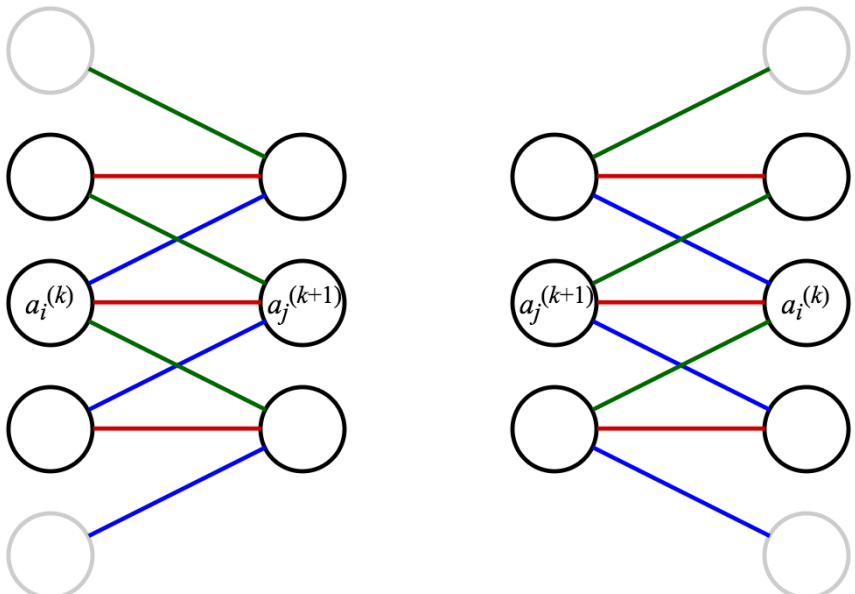
- How to train deconvolution layer?

Convolutional Autoencoders

- Convolution + deconvolution:



- How to train deconvolution layer? Treat it as convolution layer

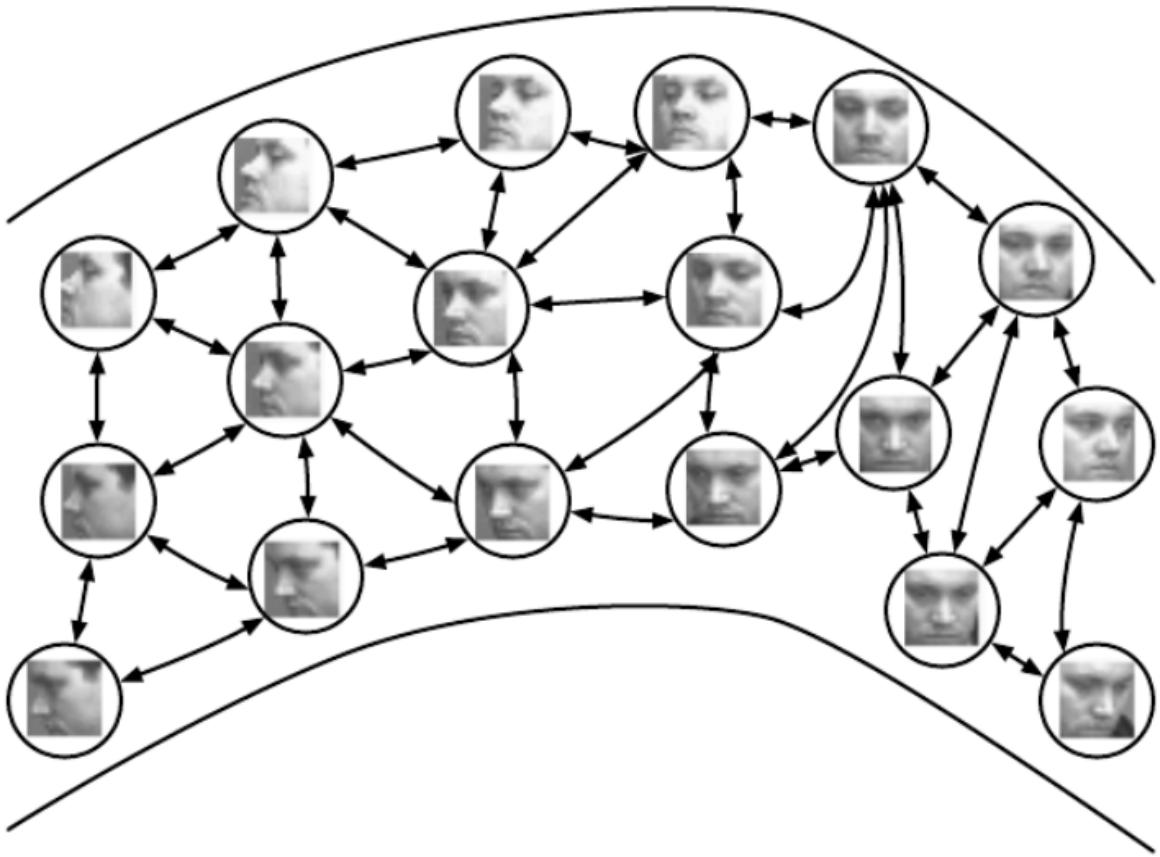


Manifolds I

- In many applications, data concentrate around one or more low-dimensional ***manifolds***

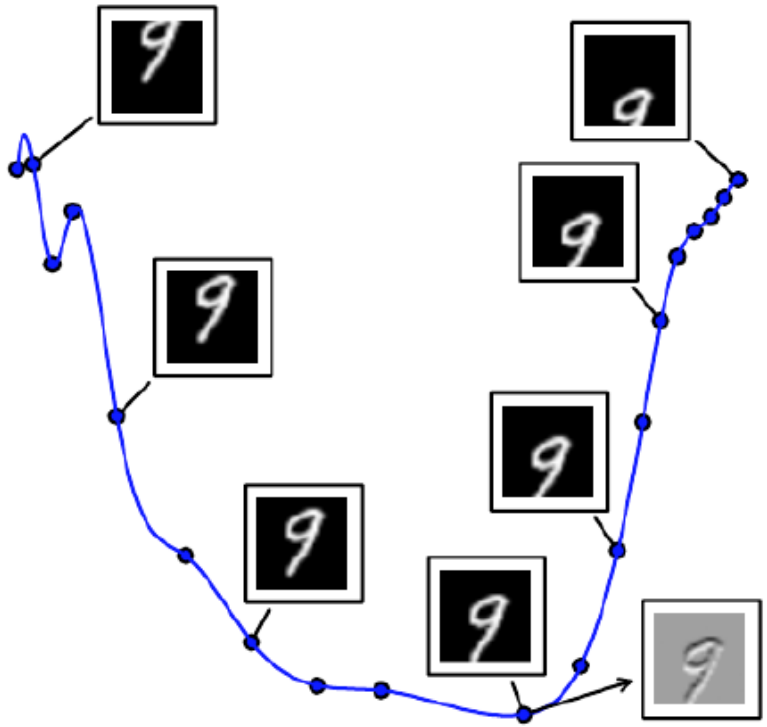
Manifolds I

- In many applications, data concentrate around one or more low-dimensional ***manifolds***
- A manifold is a topological space that are ***linear locally***



Manifolds II

- For each point $\boldsymbol{x}$ on a manifold, we have its ***tangent space*** spanned by ***tangent vectors***
  - Local directions specify how one can change $\boldsymbol{x}$ infinitesimally while staying on the manifold



Learning Manifolds I

- How to learn manifolds with autoencoders?

Learning Manifolds I

- How to learn manifolds with autoencoders?
- Contractive autoencoder [16]: regularize the code $\mathbf{c}$ such that it is invariant to local changes of $\mathbf{x}$:

$$\Omega(\mathbf{c}) = \sum_n \left\| \frac{\partial \boldsymbol{c}^{(n)}}{\partial \boldsymbol{x}^{(n)}} \right\|_F^2$$

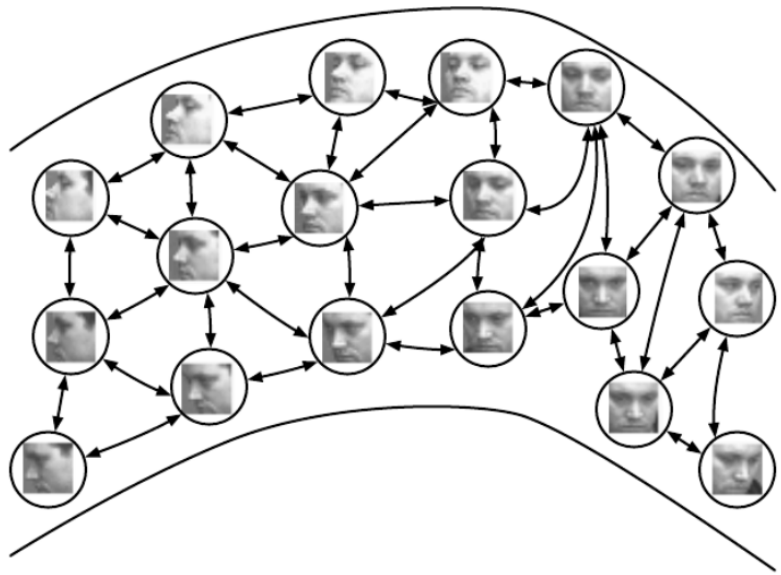
  - $\partial \boldsymbol{c}^{(n)} / \partial \boldsymbol{x}^{(n)}$ is a Jacobian matrix

Learning Manifolds I

- How to learn manifolds with autoencoders?
- Contractive autoencoder [16]: regularize the code $\mathbf{c}$ such that it is invariant to local changes of $\mathbf{x}$:

$$\Omega(\mathbf{c}) = \sum_n \left\| \frac{\partial \boldsymbol{c}^{(n)}}{\partial \boldsymbol{x}^{(n)}} \right\|_F^2$$

  - $\partial \boldsymbol{c}^{(n)} / \partial \boldsymbol{x}^{(n)}$ is a Jacobian matrix
- Encoder preserves local structures in code space



Learning Manifolds II

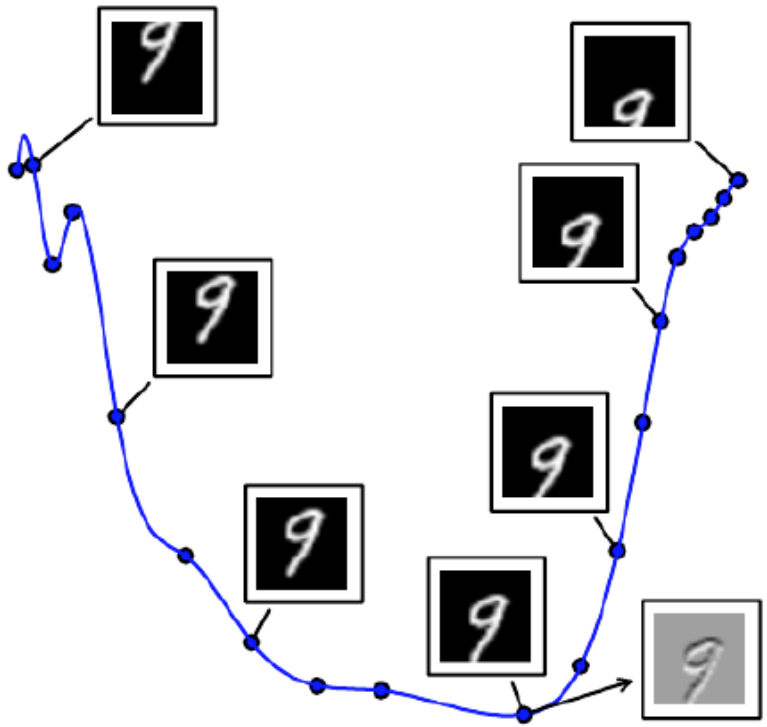
- In practice, it is easier to train a denoising autoencoder [20]:

Learning Manifolds II

- In practice, it is easier to train a denoising autoencoder [20]:
  - Encoder: to encode $\mathbf{x}$ ***with*** random noises
  - Decoder: to reconstruct $\mathbf{x}$ ***without*** noises

Tangent Vectors I

- The code $\mathbf{c}$ represents a coordinate on a low dimensional manifold
  - E.g., the blue line
- How to get the tangent vectors of a given point?



Tangent Vectors II

- Given a point $\boldsymbol{x}$, let $\boldsymbol{c}$ be the code of $\boldsymbol{x}$ and $\boldsymbol{J}(\boldsymbol{x}) = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}$ be the Jacobian matrix of $\boldsymbol{c}$ at $\boldsymbol{x}$

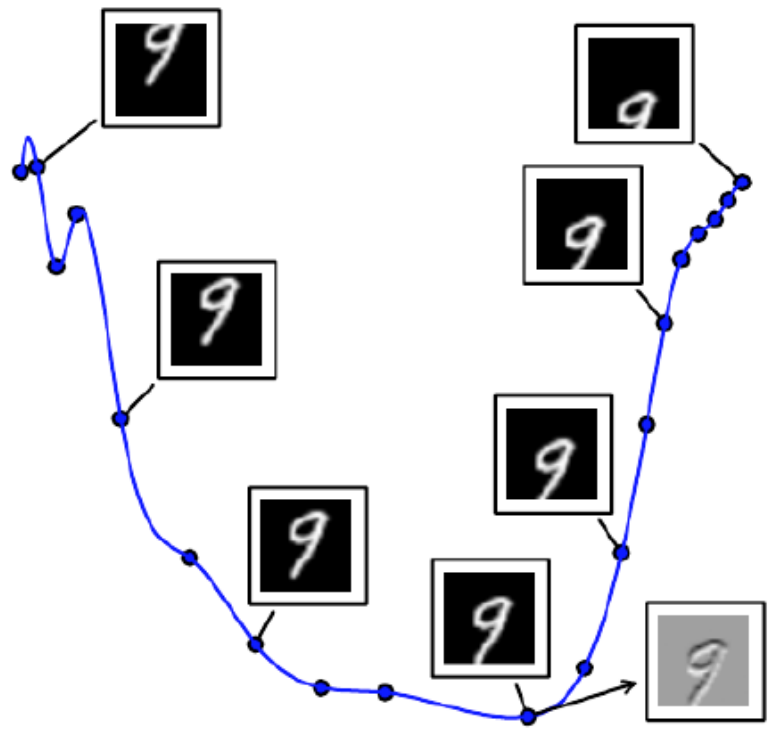
Tangent Vectors II

- Given a point $\boldsymbol{x}$, let $\boldsymbol{c}$ be the code of $\boldsymbol{x}$ and $\boldsymbol{J}(\boldsymbol{x}) = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}$ be the Jacobian matrix of $\boldsymbol{c}$ at $\boldsymbol{x}$
- $\boldsymbol{J}(\boldsymbol{x})$ summarizes how $\boldsymbol{c}$ changes in terms of $\boldsymbol{x}$

Tangent Vectors II

- Given a point $\boldsymbol{x}$, let $\boldsymbol{c}$ be the code of $\boldsymbol{x}$ and $\boldsymbol{J}(\boldsymbol{x}) = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}$ be the Jacobian matrix of $\boldsymbol{c}$ at $\boldsymbol{x}$
- $\boldsymbol{J}(\boldsymbol{x})$ summarizes how $\boldsymbol{c}$ changes in terms of $\boldsymbol{x}$

- Directions in the input space that ***changes $\boldsymbol{c}$ most*** should be tangent vectors

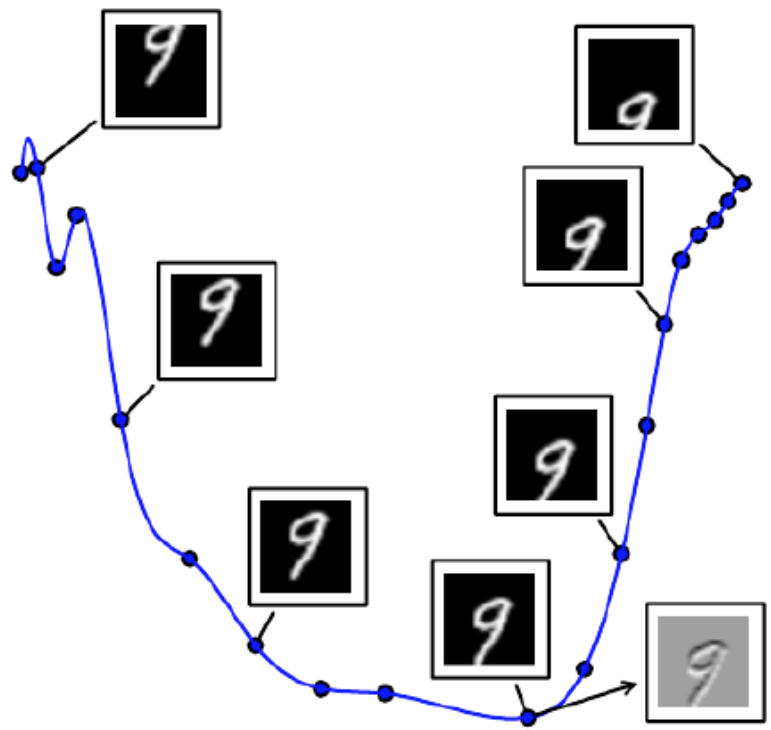


Tangent Vectors II

- Given a point $\boldsymbol{x}$, let $\boldsymbol{c}$ be the code of $\boldsymbol{x}$ and $\boldsymbol{J}(\boldsymbol{x}) = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}$ be the Jacobian matrix of $\boldsymbol{c}$ at $\boldsymbol{x}$
- $\boldsymbol{J}(\boldsymbol{x})$ summarizes how $\boldsymbol{c}$ changes in terms of $\boldsymbol{x}$

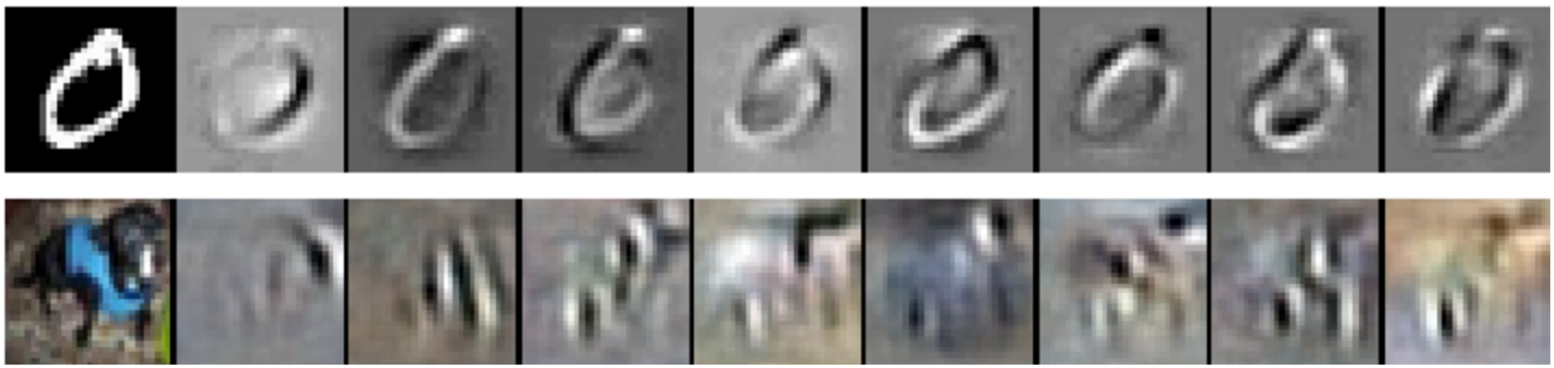
- Directions in the input space that ***changes $\boldsymbol{c}$ most*** should be tangent vectors

1. Decompose $\boldsymbol{J}(\boldsymbol{x})$ using SVD such that $\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$
2. Let tangent vectors be rows of $\boldsymbol{V}$ corresponding to the largest singular values in $\boldsymbol{D}$



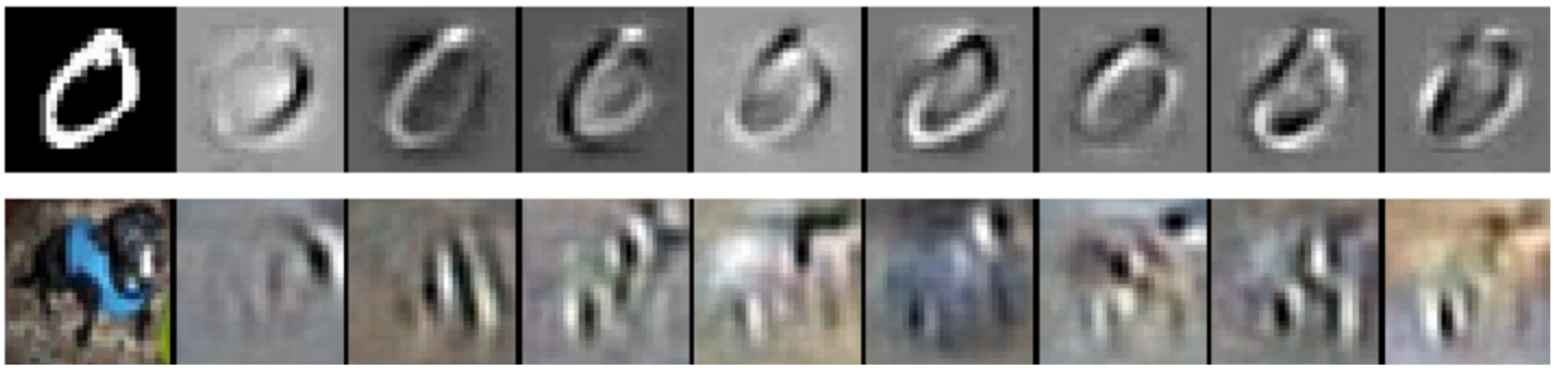
Tangent Vectors III

- In practice, $\boldsymbol{J}(\boldsymbol{x})$ usually has few large singular values
- Tangent vectors found by contractive/denoising autoencoders:



Tangent Vectors III

- In practice, $\boldsymbol{J}(\boldsymbol{x})$ usually has few large singular values
- Tangent vectors found by contractive/denoising autoencoders:

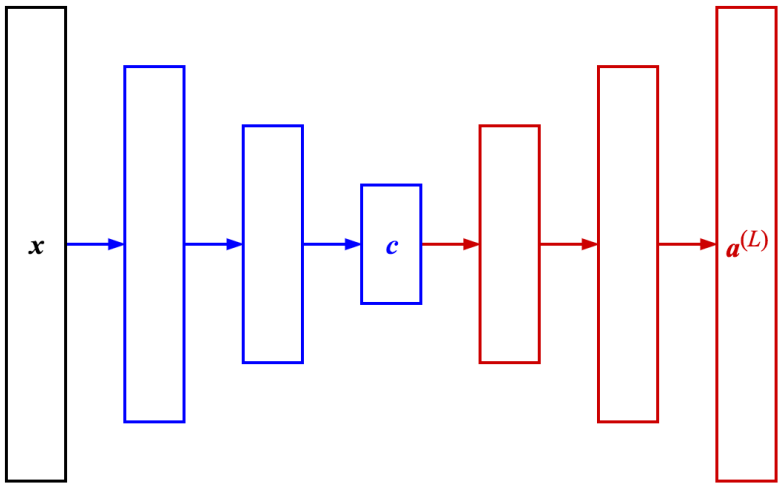


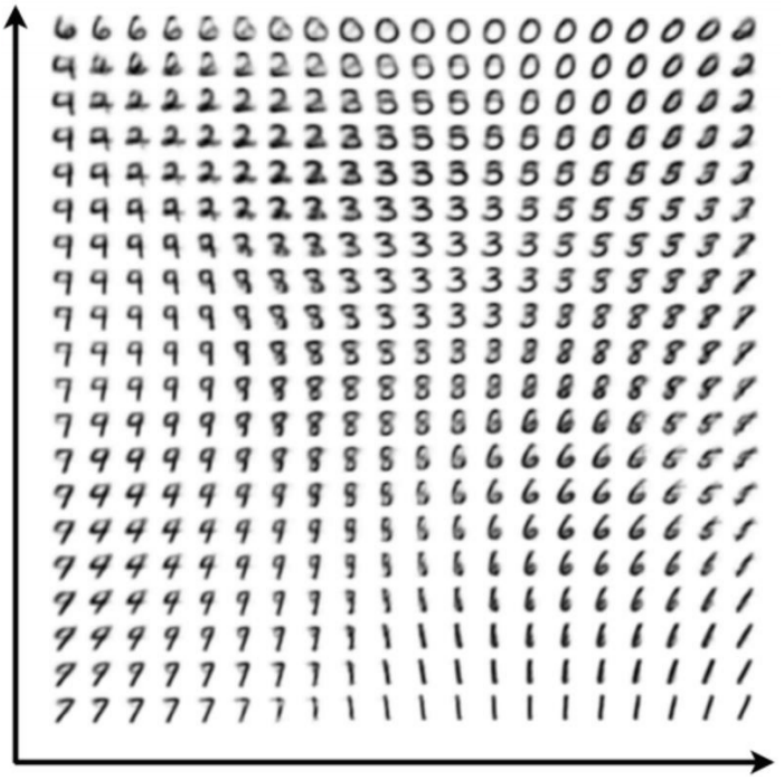
- Can be used by Tangent Prop [18]:
- Let $\{\boldsymbol{v}^{(i,j)}\}_j$ be tangent vectors of each example $\boldsymbol{x}^{(i)}$
- Trains an NN classifier f with cost penalty: $\Omega[f] = \sum_{i,j} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(i)})^\top \boldsymbol{v}^{(i,j)}$
  - Points in the same manifold share the same label

Outline

Decoder as Data Generator

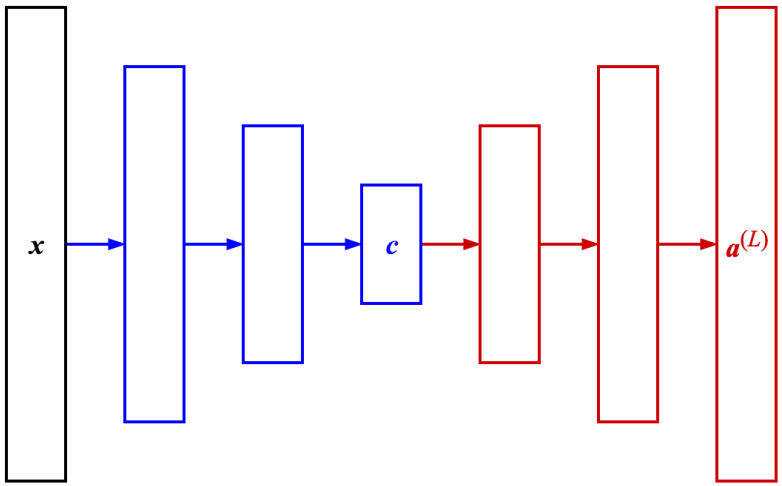
- Decoder can generate data points even with ***synthetic codes***

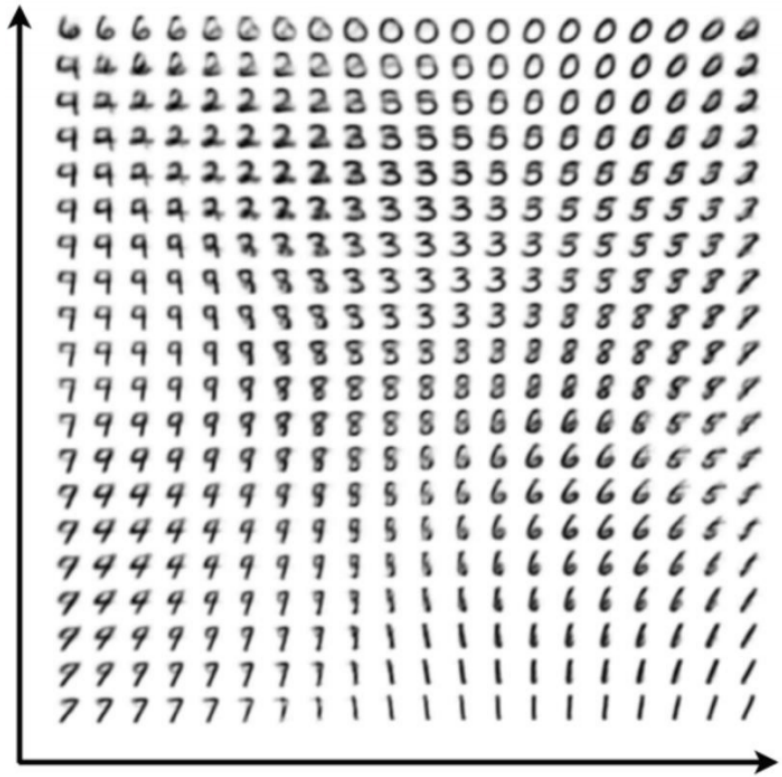




Decoder as Data Generator

- Decoder can generate data points even with ***synthetic codes***

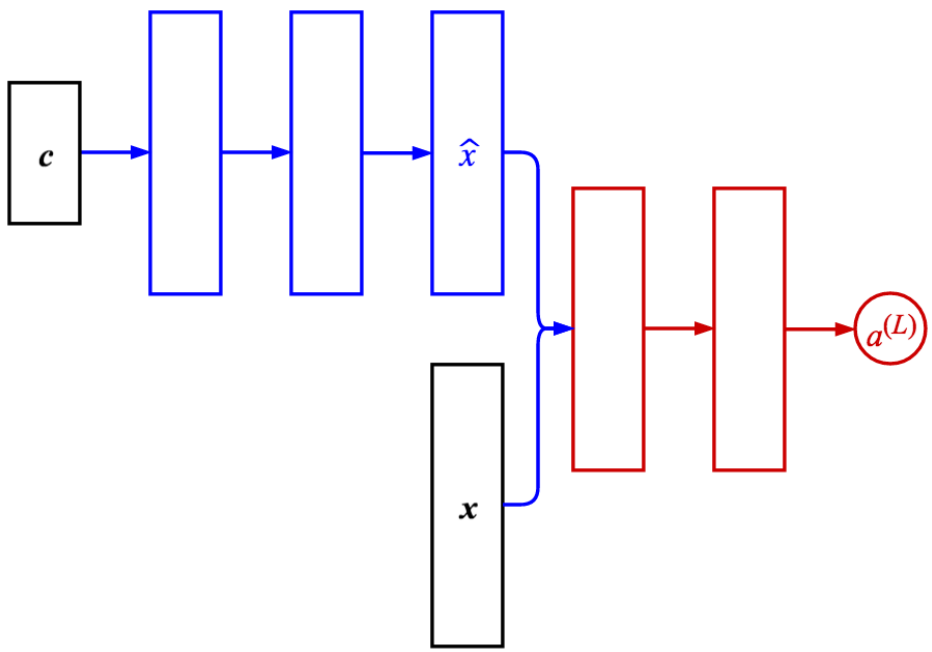




- However, decoder just "remembers" the samples it has seen
  - Generated data are just combinations of training examples

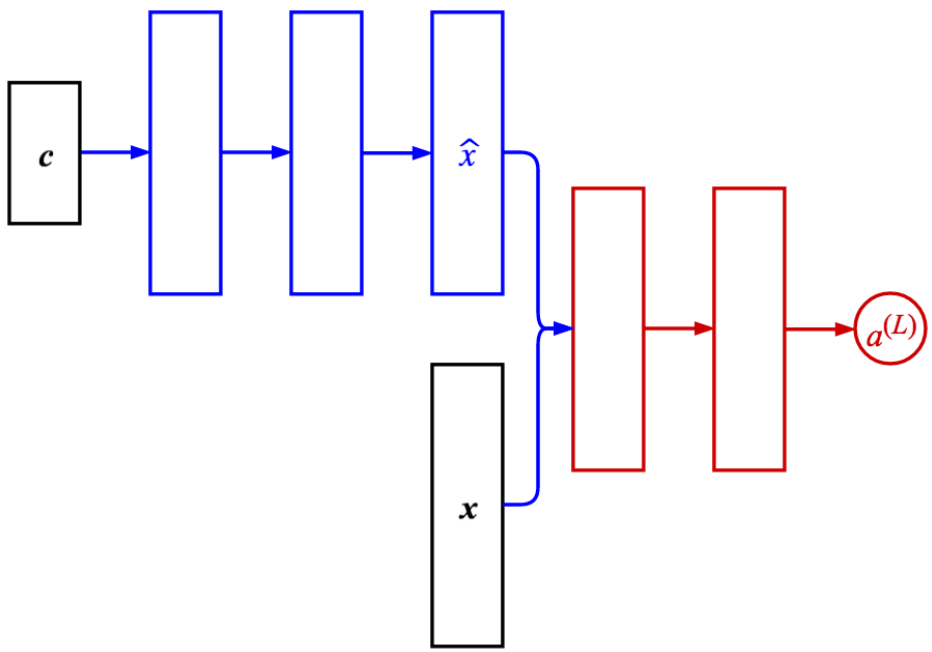
Generative Adversarial Networks (GANs)

- *Generative adversarial network* (*GAN*) [4]:
- Generator: to generate data points from random codes
- Discriminator: to separate generated points from true ones



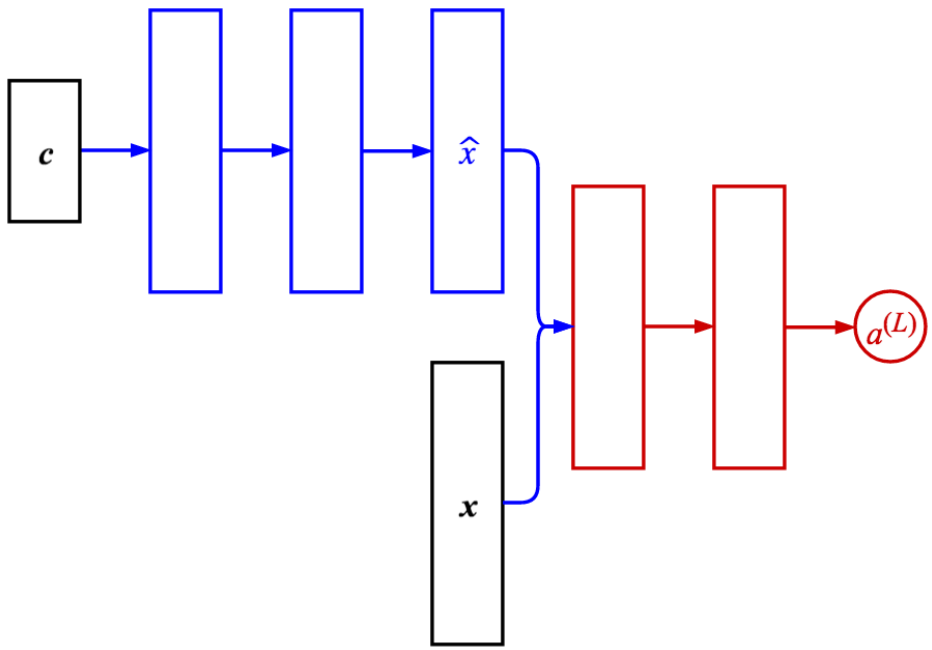
Generative Adversarial Networks (GANs)

- *Generative adversarial network* (*GAN*) [4]:
- Generator: to generate data points from random codes
- Discriminator: to separate generated points from true ones
  - Sigmoid output unit $a^{(L)} = \hat{\rho}$ for $\mathrm{P}(\mathrm{y} = \text{true point} \,|\, \mathbf{x}) \sim \mathrm{Bernoulli}(\rho)$



Generative Adversarial Networks (GANs)

- *Generative adversarial network* (*GAN*) [4]:
- Generator: to generate data points from random codes
- Discriminator: to separate generated points from true ones
  - Sigmoid output unit $a^{(L)} = \hat{\rho}$ for $\mathrm{P}(\mathrm{y} = \text{true point} \,|\, \mathbf{x}) \sim \mathrm{Bernoulli}(\rho)$
  - Weights for $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are tied

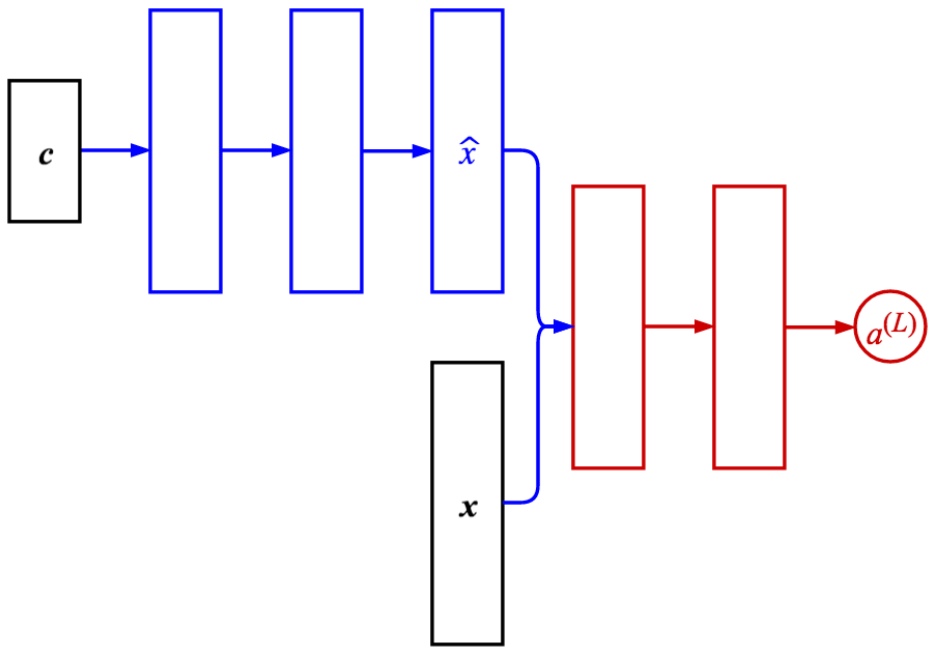


Cost Function

- In normal binary classification, we have the log likelihood
  $\log \mathrm{P}(\mathbb{X}|\Theta) = \sum_n \log \mathrm{P}(y^{(n)}|\boldsymbol{x}^{(n)}, \Theta) = \sum_n \log \left[(\hat{\rho}^{(n)})^{y^{(n)}} (1-\hat{\rho}^{(n)})^{(1-y^{(n)})} \right]$
- Cost function (N true points, M generated points):

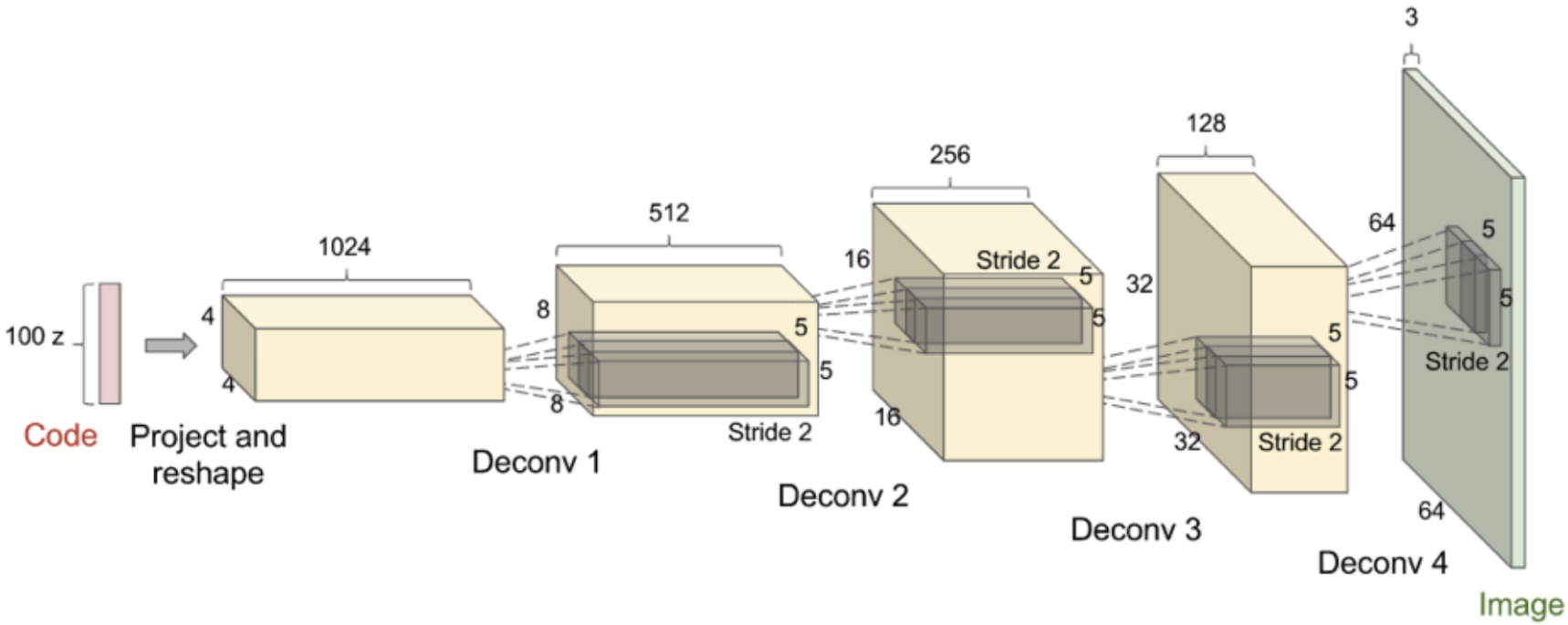
$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1-\hat{\rho}^{(m)})$$

- $\hat{\rho}^{(n)}$ depends on Θ_{dis} only
- $\hat{\rho}^{(m)}$ depends on both Θ_{dis} and Θ_{gen}



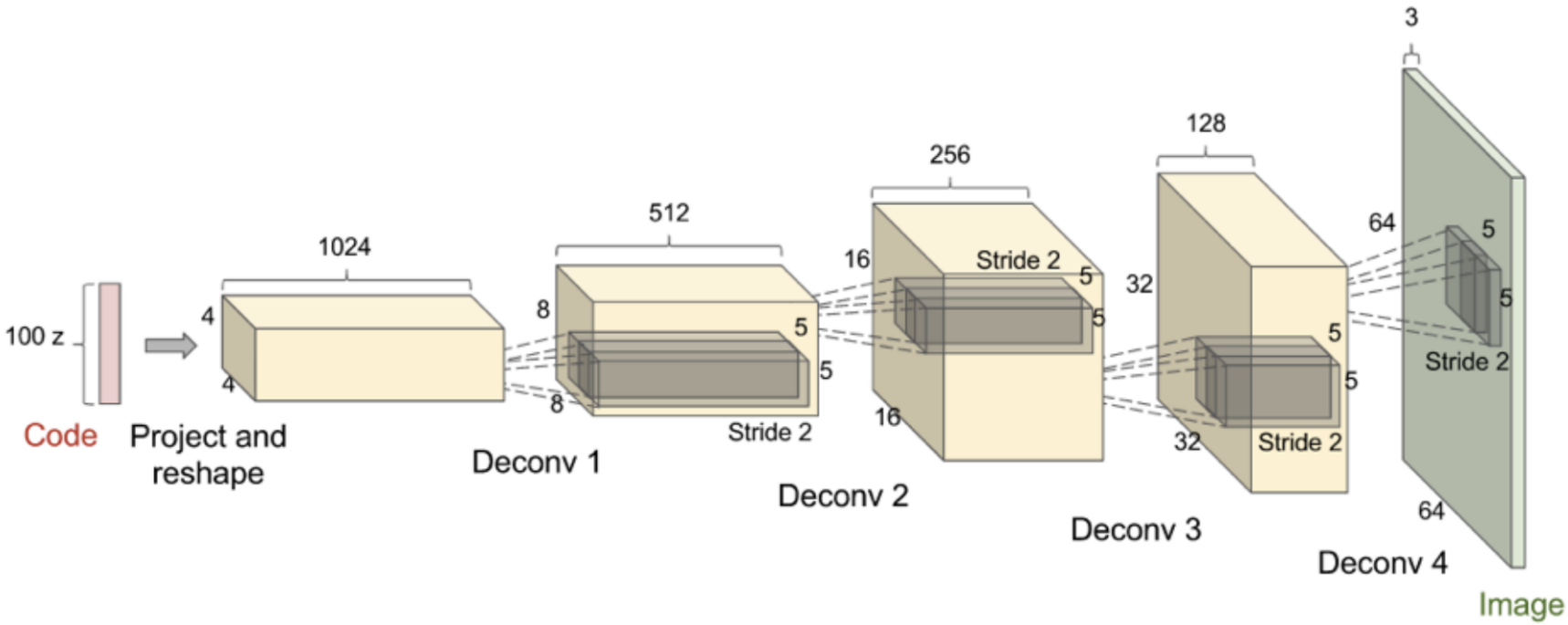
Convolutional Generators

- DC-GAN [14] for generating images:



Convolutional Generators

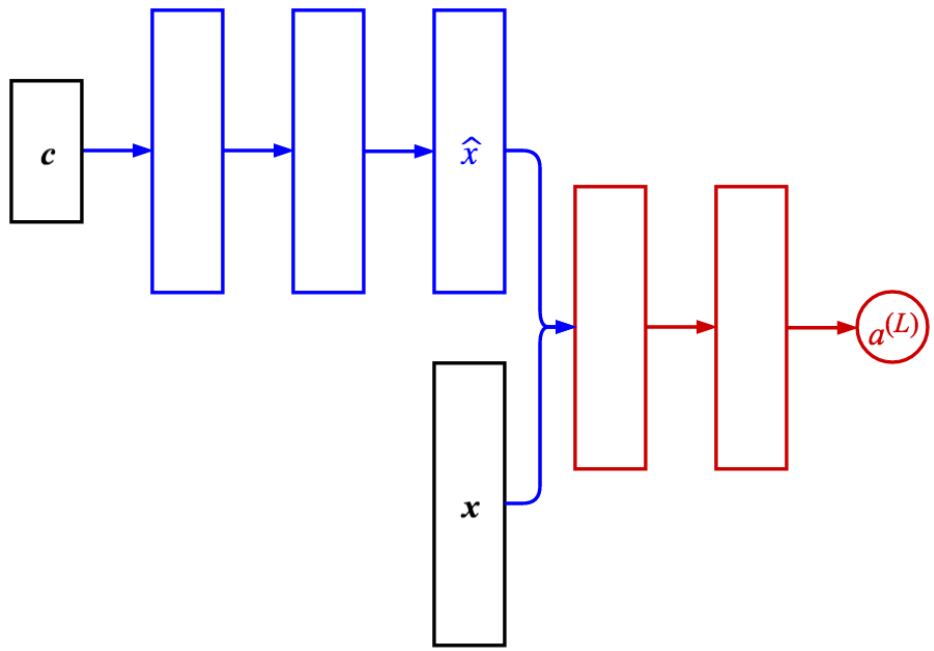
- DC-GAN [14] for generating images:



Training: Alternative SGD

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

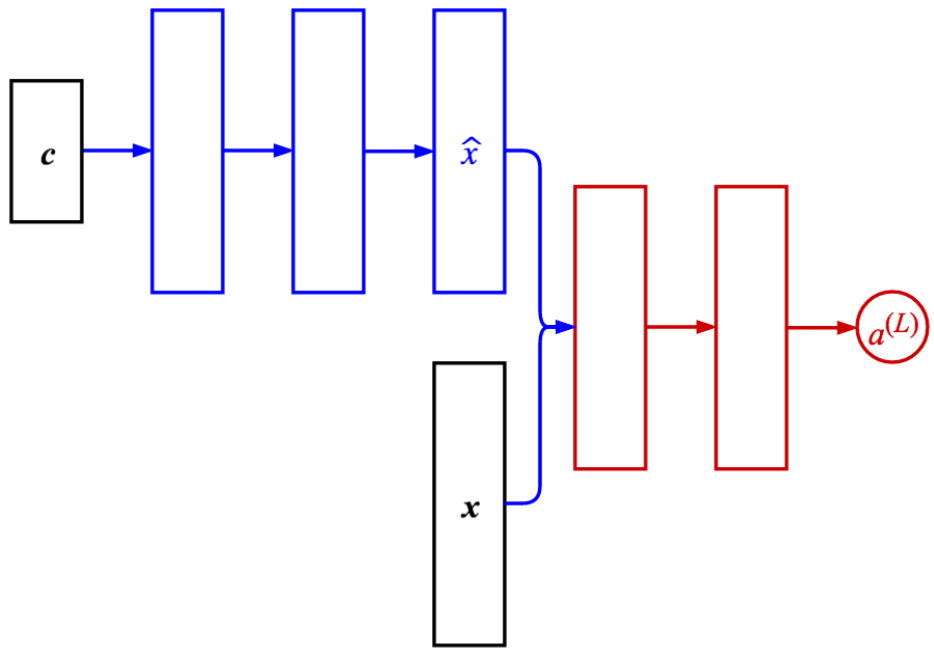
- Alternate SGD:



Training: Alternative SGD

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

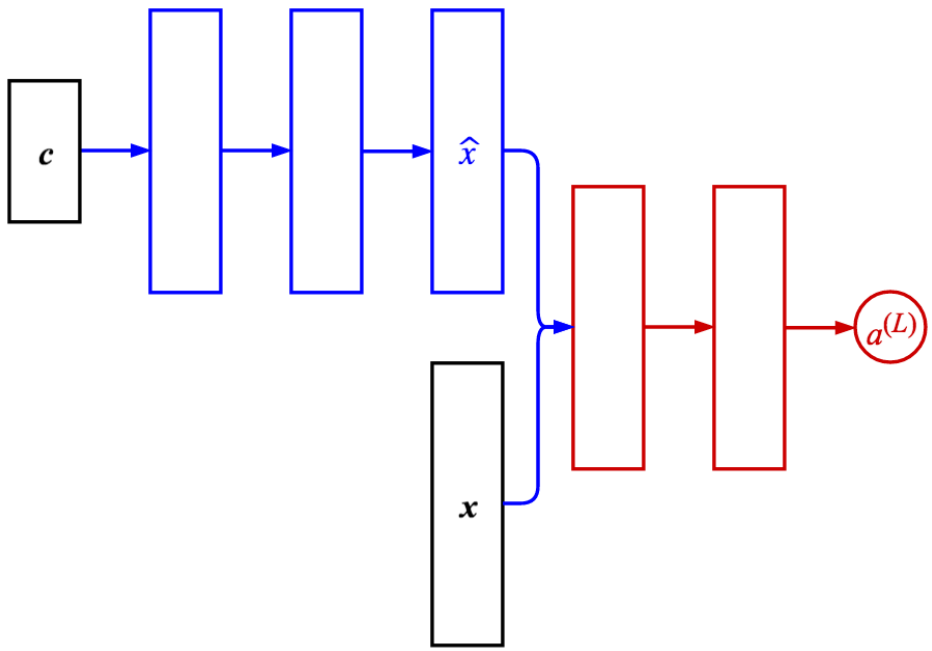
- Alternate SGD:
- Fix Θ_{gen}, apply a stochastic gradient ascent step on Θ_{dis}
  - $\nabla_{\Theta_{\text{dis}}} C$ involves both the first and second term



Training: Alternative SGD

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

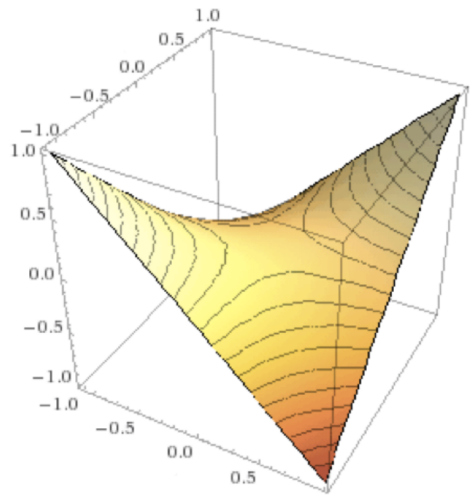
- Alternate SGD:
- Fix Θ_{gen}, apply a stochastic gradient ascent step on Θ_{dis}
  - $\nabla_{\Theta_{\text{dis}}} C$ involves both the first and second term
- Fix Θ_{dis}, apply a stochastic gradient ascent step on Θ_{gen}
  - $\nabla_{\Theta_{\text{gen}}} C$ involves only the second term



Training: Challenges I

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- The goal is to find a saddle point

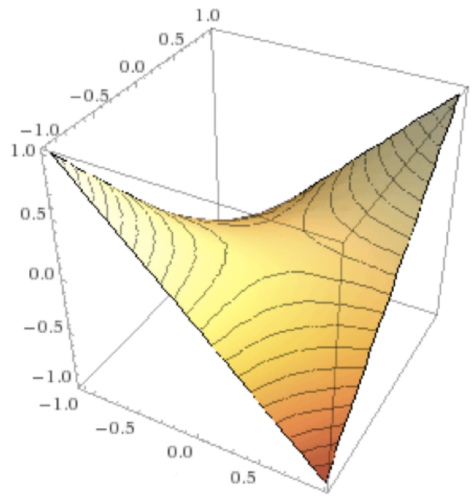


- However, most optimization algorithms are design to find minima

Training: Challenges I

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- The goal is to find a saddle point



- However, most optimization algorithms are design to find minima
- Avoid momentum in training algorithm

Training: Challenges II

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

Training: Challenges II

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- Alternate SGD does not distinguish between $\min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}}$ and $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$

Training: Challenges II

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- Alternate SGD does not distinguish between $\min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}}$ and $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$
- Mode collapsing: in $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator maps every code to a single point that discriminator believes is most likely to be real

Training: Challenges II

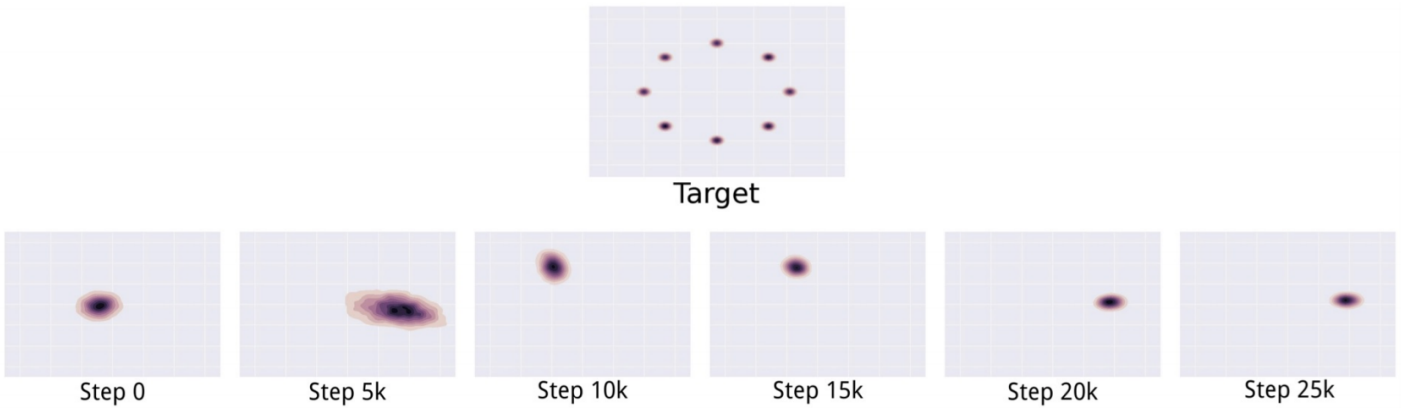
$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- Alternate SGD does not distinguish between $\min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}}$ and $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$
- Mode collapsing: in $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator maps every code to a single point that discriminator believes is most likely to be real
  - Then, discriminator can spot fake points by excluding the point

Training: Challenges II

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- Alternate SGD does not distinguish between $\min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}}$ and $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$
- Mode collapsing: in $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator maps every code to a single point that discriminator believes is most likely to be real
  - Then, discriminator can spot fake points by excluding the point
- Discriminator and generator cancel each other's progress during training steps



Unrolled GANs

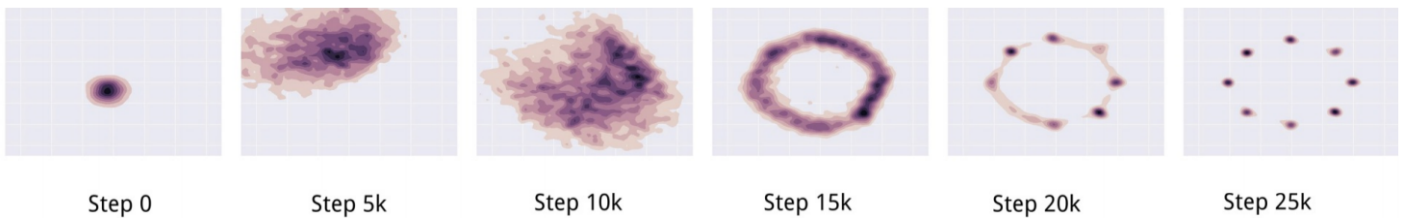
$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- SGD ignores max operation when computing $\nabla_{\Theta_{\text{gen}}} C$

Unrolled GANs

$$\arg \min_{\Theta_{\text{gen}}} \max_{\Theta_{\text{dis}}} \sum_n \log \hat{\rho}^{(n)} + \sum_m \log(1 - \hat{\rho}^{(m)})$$

- SGD ignores max operation when computing $\nabla_{\Theta_{\text{gen}}} C$
- Unrolled GANs [11]: to back-propagate through ***several*** max ***steps*** when computing $\nabla_{\Theta_{\text{gen}}} C$



Minibatch Discrimination

- In $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator collapses because $\nabla_{\Theta_{\text{dis}}} C$ are computed independently for each example

Minibatch Discrimination

- In $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator collapses because $\nabla_{\Theta_{\text{dis}}} C$ are computed independently for each example
- Generator cannot know if discriminator is excluding a single region
  - Cannot learn to generate dissimilar points

Minibatch Discrimination

- In $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator collapses because $\nabla_{\Theta_{\text{dis}}} C$ are computed independently for each example
- Generator cannot know if discriminator is excluding a single region
  - Cannot learn to generate dissimilar points
- Minibatch discrimination [17]: to let discriminator ***look at multiple points*** when making predictions

Minibatch Discrimination

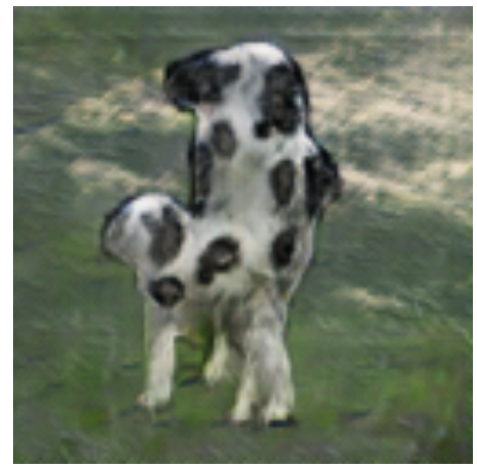
- In $\max_{\Theta_{\text{dis}}} \min_{\Theta_{\text{gen}}}$, generator collapses because $\nabla_{\Theta_{\text{dis}}} C$ are computed independently for each example
- Generator cannot know if discriminator is excluding a single region
  - Cannot learn to generate dissimilar points
- Minibatch discrimination [17]: to let discriminator ***look at multiple points*** when making predictions
- Without vs. with minibatch discrimination:

Other Difficulties

Counting

Perspective

Global structure

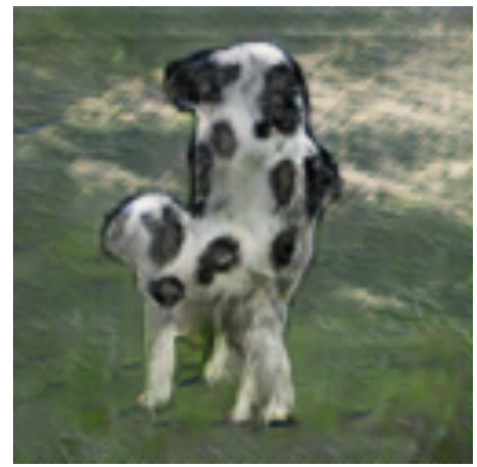


Other Difficulties

Counting

Perspective

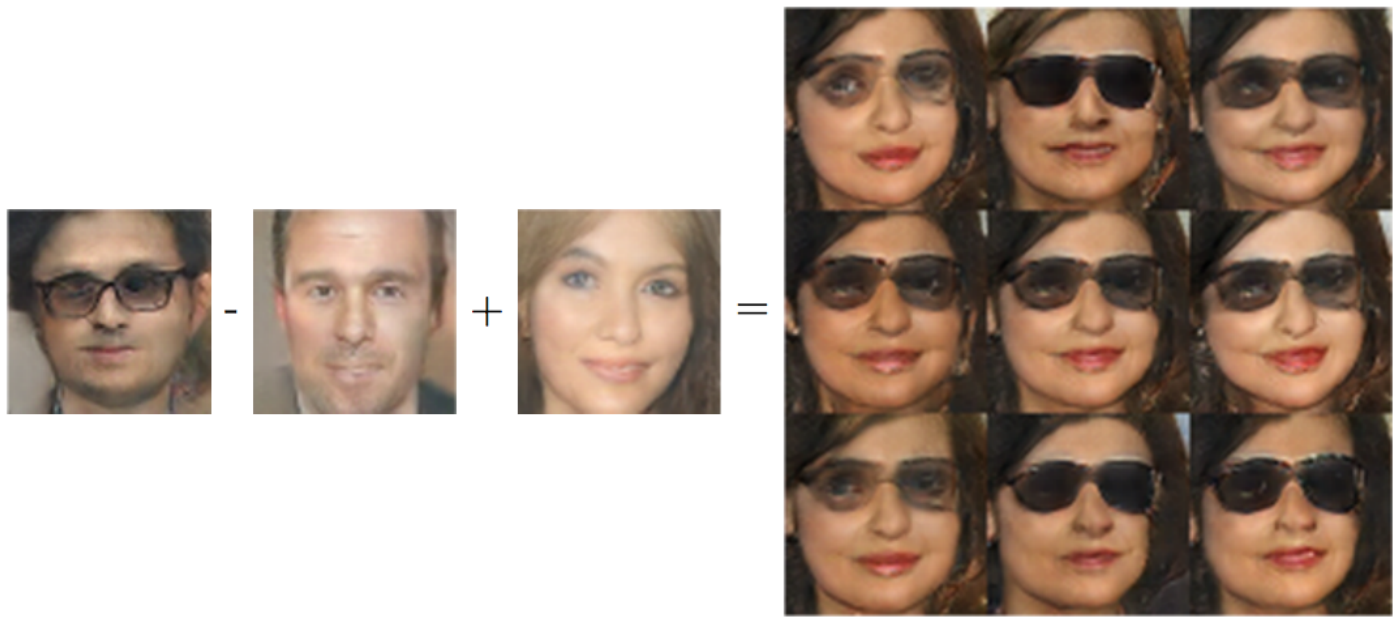
Global structure



- Still are research problems

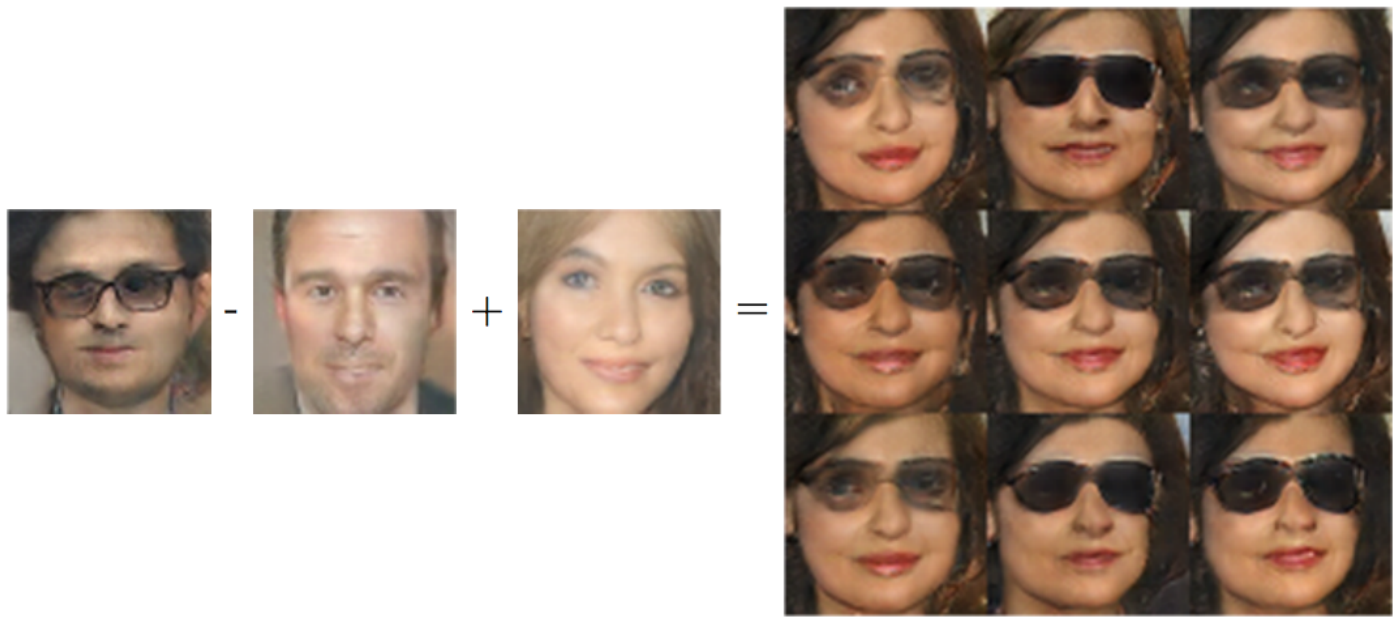
Code Space Arithmetics

- DC-GAN [14] can learn to use codes in meaningful ways:



Code Space Arithmetics

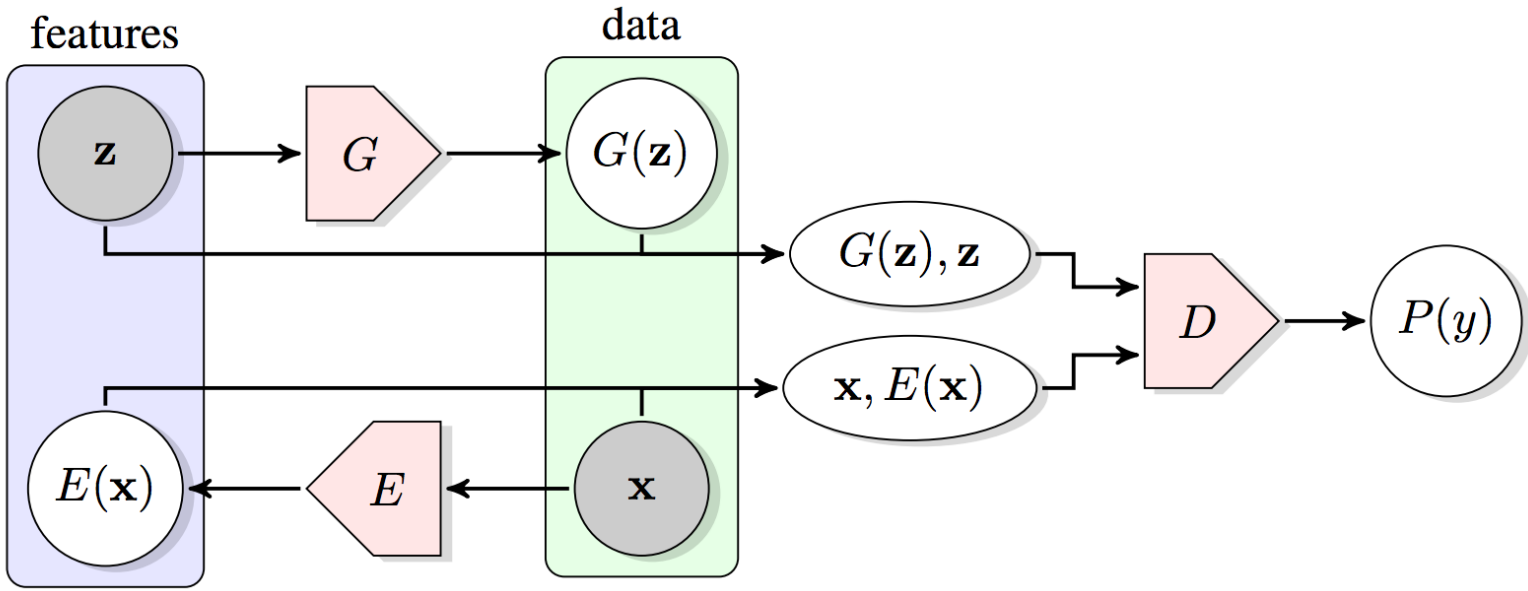
- DC-GAN [14] can learn to use codes in meaningful ways:



- Finding codes for images with constraints [22, 1] ▸ Demo 1 ▸ Demo 2

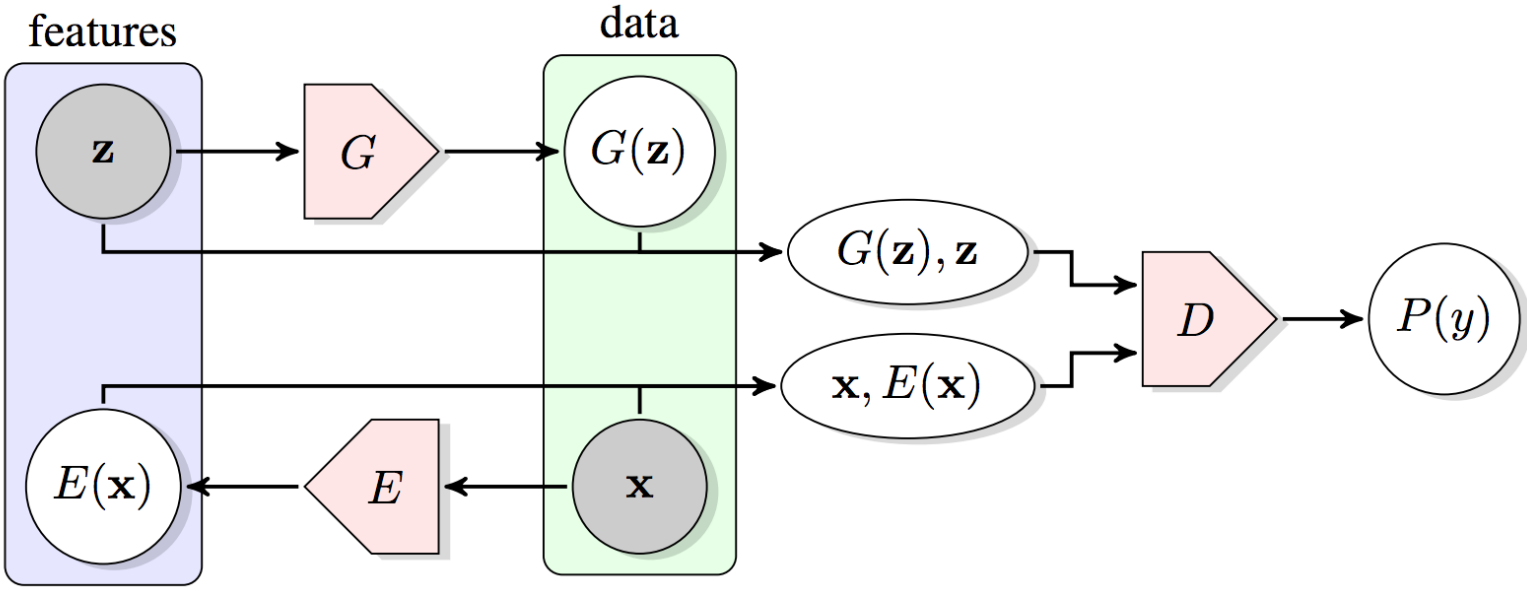
Learn to Encode in GANs

- Adversarial feature learning [2, 3]:



Learn to Encode in GANs

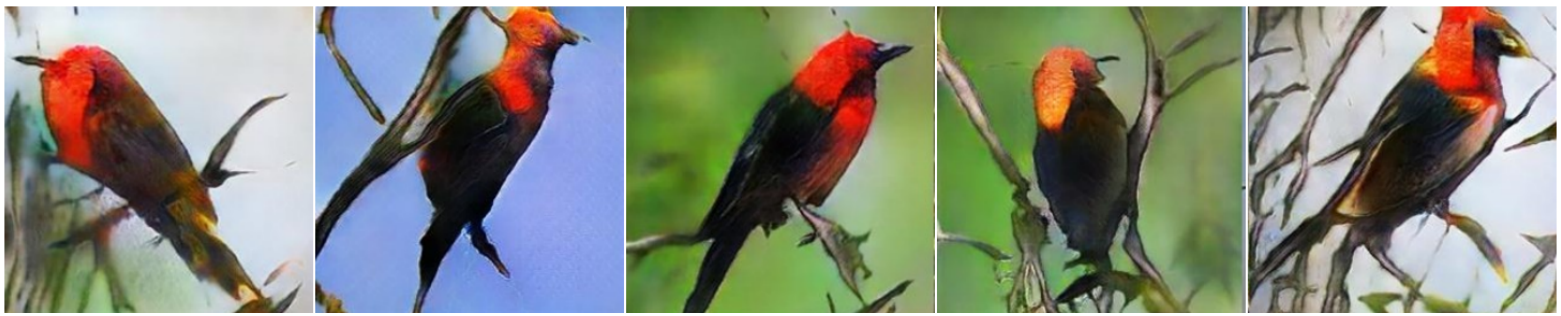
- Adversarial feature learning [2, 3]:



Conditional GAN I

- How?

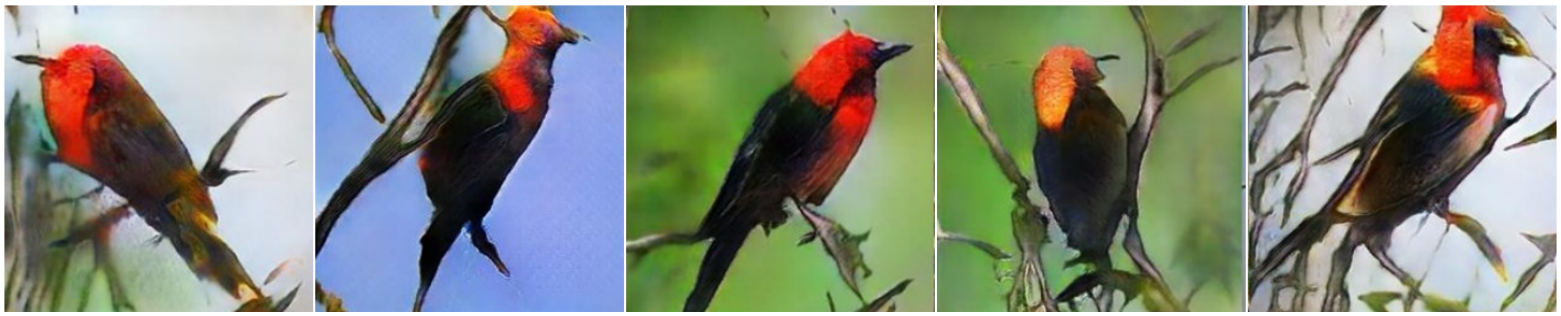
"*This bird is completely red with black wings and pointy beak.*"



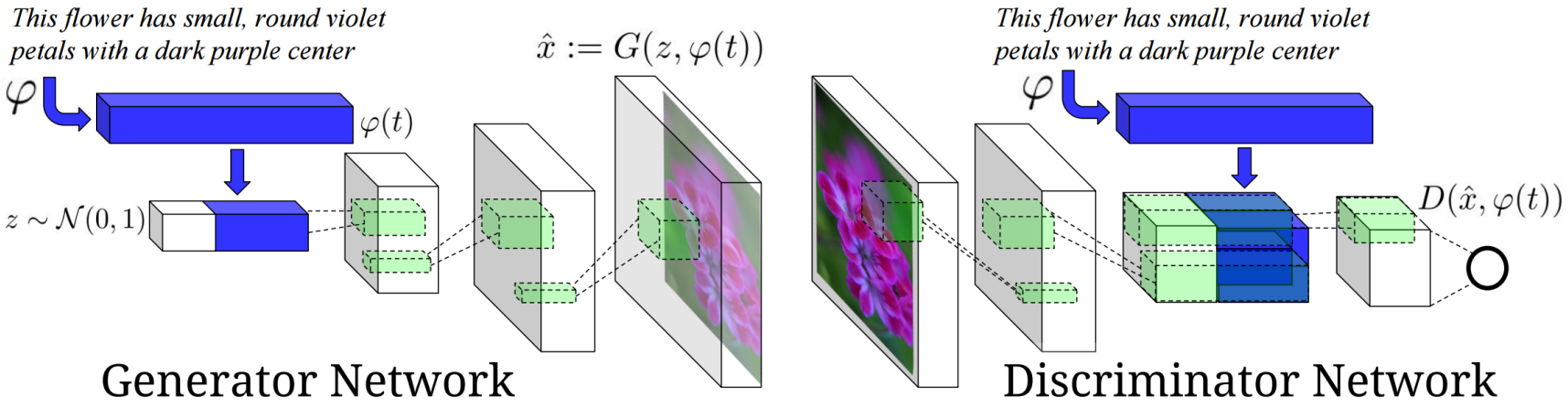
Conditional GAN I

- How?

  "*This bird is completely red with black wings and pointy beak.*"

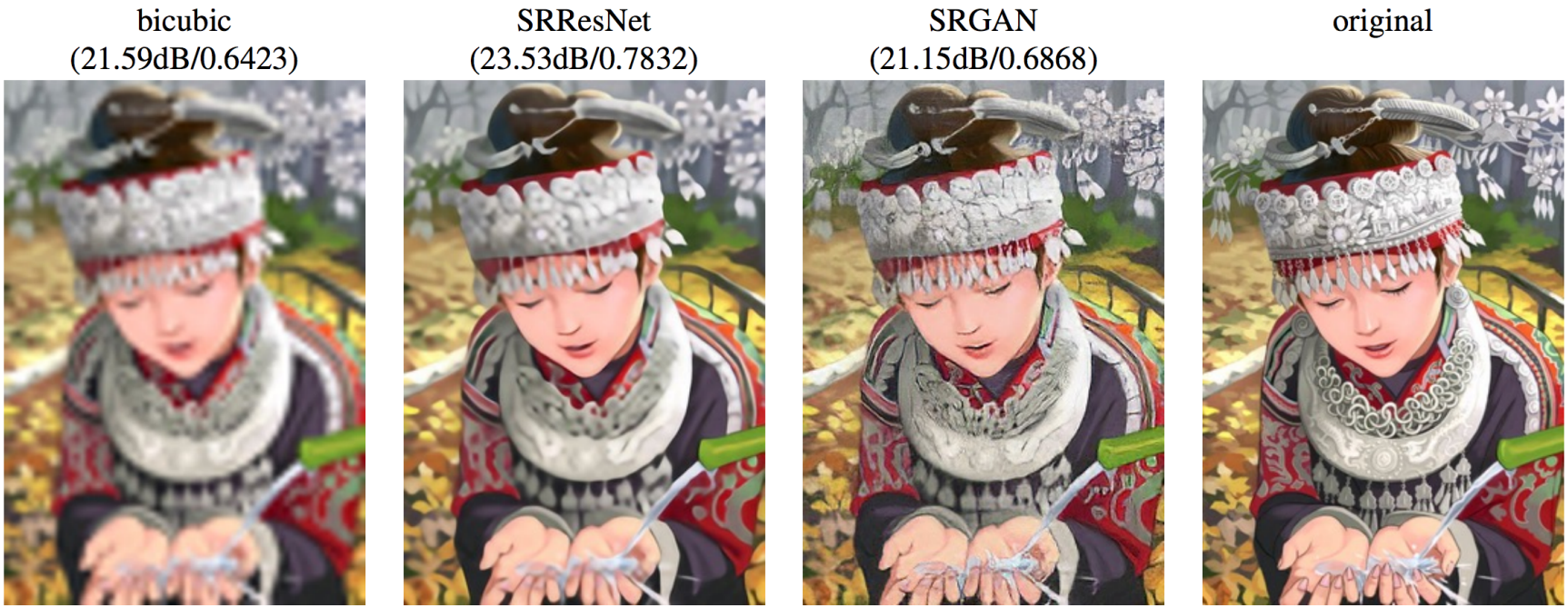


- Text to image [15]:



More GANs I

- Super resolution [8]:



More GANs II

- Image-to-image translation [5]:

Reference I

[1] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston.
Neural photo editing with introspective adversarial networks.
arXiv preprint arXiv:1609.07093, 2016.

[2] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell.
Adversarial feature learning.
arXiv preprint arXiv:1605.09782, 2016.

[3] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville.
Adversarially learned inference.
arXiv preprint arXiv:1606.00704, 2016.

Reference II

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.
Generative adversarial nets.
In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros.
Image-to-image translation with conditional adversarial networks.
arXiv preprint arXiv:1611.07004, 2016.

[6] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu.
Video pixel networks.
arXiv preprint arXiv:1610.00527, 2016.

[7] Quoc V Le and Tomas Mikolov.
Distributed representations of sentences and documents.
In *ICML*, volume 14, pages 1188–1196, 2014.

Reference III

[8] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al.
Photo-realistic single image super-resolution using a generative adversarial network.
arXiv preprint arXiv:1609.04802, 2016.

[9] Daniel D Lee and H Sebastian Seung.
Learning the parts of objects by non-negative matrix factorization.
Nature, 401(6755):788–791, 1999.

[10] Daniel D Lee and H Sebastian Seung.
Algorithms for non-negative matrix factorization.
In *Advances in neural information processing systems*, pages 556–562, 2001.

Reference IV

[11] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein.
Unrolled generative adversarial networks.
arXiv preprint arXiv:1611.02163, 2016.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781, 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
Distributed representations of words and phrases and their compositionality.
In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Reference V

[14] Alec Radford, Luke Metz, and Soumith Chintala.
Unsupervised representation learning with deep convolutional generative adversarial networks.
arXiv preprint arXiv:1511.06434, 2015.

[15] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee.
Generative adversarial text to image synthesis.
arXiv preprint arXiv:1605.05396, 2016.

[16] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio.
Contractive auto-encoders: Explicit invariance during feature extraction.
In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

Reference VI

[17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen.
Improved techniques for training gans.
In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[18] Patrice Simard, Bernard Victorri, Yann LeCun, and John S Denker.
Tangent prop-a formalism for specifying selected invariances in an adaptive network.
In *NIPS*, volume 91, pages 895–903, 1991.

[19] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al.
Conditional image generation with pixelcnn decoders.
In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

Reference VII

[20] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol.
Extracting and composing robust features with denoising autoencoders.
In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[21] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba.
Anticipating the future by watching unlabeled video.
arXiv preprint arXiv:1504.08023, 2015.

[22] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros.
Generative visual manipulation on the natural image manifold.
In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.