Generative Models: GANs, VAEs, Learning Representations

Prof. Manolis Kellis

Slides Credit: Ben Lengerich, Relja Arandjelovic, Fei-Fei Li, Justin Johnson, Serena Yeung
Interpretable Deep Learning

1. Intro to Interpretability
   1a. Interpretability definition: Convert implicit NN information to human-interpretable information
   1b. Motivation: Verify model works as intended; debug classifier; make discoveries; Right to explanation
   1c. Ante-hoc (train interpretable model) vs. Post-hoc (interpret complex model; degree of “locality”)

2. Interpreting Deep Neural Networks
   2a. Interpreting Models (macroscopic, understand internals) vs. decisions (microscopic, practical applications)
   2b. Interpreting Models: Weight visualization, Surrogate model, Activation maximization, Example-based
   2c. Interpreting Decisions:
      - Example-based
      - Attribution Methods: why are gradients noisy?
      - Gradient-based Attribution: SmoothGrad, Interior Gradient
      - Backprop-based Attribution: Deconvolution, Guided Backpropagation

3. Evaluating Attribution Methods
   3a. Qualitative: Coherence: Attributions should highlight discriminative features / objects of interest
   3b. Qualitative: Class Sensitivity: Attributions should be sensitive to class labels
   3c. Quantitative: Sensitivity: Removing feature with high attribution $\Rightarrow$ large decrease in class probability
   3d. Quantitative: ROAR & KAR. Low class prob cuz image unseen $\Rightarrow$ remove pixels, retrain, measure acc. drop

Slides by Beomsu Kim, KAIST
Previously

- We’ve seen ways to model $X \rightarrow Y$ for a fixed dataset
- But that’s not what the world looks like:
  - No $Y$
  - Limited samples
  - Many related datasets
  - Changes over time
- Can we learn the rules which govern how the real world varies?
Learning Representations
Key idea: **Representation learning**

In deep learning, the two tasks are **coupled**:
- the **classification task** “drives” the **feature extraction**
- Extremely powerful and general paradigm
  - **Be creative**! The field is still at its infancy!
  - New application domains (e.g. beyond images) can have **structure** that current architectures **do not capture/exploit**
  - Genomics/biology/neuroscience can help drive development of **new architectures**
Representation learning **without annotations?**

Many ideas are possible (and yours could be even better!):

1. Predict the future: RNNs, Video
2. Pretext tasks: predict self, before/after, missing patch, correct rotation, colorization, up-sampling, multimodal
3. Compression: Autoencoder (predict self, through clamp), representation \( z \)
5. Make latent space parameters \( z \) meaningful, orthogonal, explicit, tuneable
6. Train using a second network: GANs - Improve quality of output images
7. The Sky is the Limit
Representation learning **without annotations?**

### Many ideas are possible (and yours could be even better!):

1. Predict the future: RNNs, Video
2. Compression: Autoencoder (predict self, through clamp), representation $z$
3. Pretext tasks: predict self, before/after, missing patch, correct rotation, colorization, up-sampling, multimodal
5. Make latent space parameters $z$ meaningful, orthogonal, explicit, tuneable
6. Train using a second network: GANs - Improve quality of output images
7. The Sky is the Limit
Pretext Tasks
A tour of pretext tasks

Self-supervised learning
- **Goal**: Learn good representations
- **Means**: Construct a pretext task
  - Don’t care about the pretext task itself
  - Only important it enables learning

Rough pretext task classification
- Inferring structure
- Transformation prediction
- Reconstruction
- Exploiting time
- Multimodal
- Instance classification

Disclaimer
- Rough classification of tasks, some fit multiple categories
- Trying to cover many but inevitably missing many works
- Often have to pick one of multiple concurrent similar methods
- If A comes before B in this presentation, it doesn’t mean A did it first

Slide Credit: Relja Arandjelovic
Inferring structure
Context prediction

Can you guess the spatial configuration for the two pairs of patches?

Question 1:

Question 2:
Context prediction

Can you guess the spatial configuration for the two pairs of patches? Much easier if you recognize the object!

Question 1:

Question 2:

Intuition
- The network should learn to recognize object parts and their spatial relations

Slide Credit: Relja Arandjelovic
Context prediction

\[ X = (\text{cat face}, \text{cat neck}); Y = 3 \]

["Unsupervised visual representation learning by context prediction", Doersh et al. 15]
Context prediction

Pros
- (arguably) The first self-supervised method
- Intuitive task that should enable learning about object parts

Cons
- Assumes training images are photographed with canonical orientations (and canonical orientations exist)
- Training on patches, but trying to learn image representations
- Networks can “cheat” so special care is needed [discussed later]
  - Further gap between train and eval
- Not fine-grained enough due to no negatives from other images
  - e.g. no reason to distinguish cat from dog eyes
- Small output space - 8 cases (positions) to distinguish?

Slide Credit: Relja Arandjelovic
Jigsaw puzzles

Divide image into patches and permute them

Predict the permutation

Network

Classifier

Pros & Cons: Same as for context prediction apart from being harder

["Unsupervised learning of visual representations by solving jigsaw puzzles", Noroozi et al. 17]
Transformation prediction

Slide Credit: Relja Arandjelovic
Rotation prediction

Can you guess how much rotated is applied?

[“Unsupervised representation learning by predicting image rotations”, Gidaris et al. 18]
Rotation prediction

Can you guess how much rotated is applied? Much easier if you recognize the content!

90° rotation  270° rotation  180° rotation  0° rotation
Rotation prediction

Pros
- Very simple to implement and use, while being quite effective

Cons
- Assumes training images are photographed with canonical orientations (and canonical orientations exist)
- Train-eval gap: no rotated images at eval
- Not fine-grained enough due to no negatives from other images
  - e.g. no reason to distinguish cat from dog
- Small output space – 4 cases (rotations) to distinguish [not trivial to increase; see later]
- Some domains are trivial e.g. StreetView ⇒ just recognize sky
Relative transformation prediction

Estimate the transformation between two images. Requires good features

Slide Credit: Relja Arandjelovic
Relative transformation prediction

Pros
- In line with classical computer vision, e.g. SIFT was developed for matching

Cons
- Train-eval gap: no transformed images at eval
- Not fine-grained enough due to no negatives from other images
  - e.g. no reason to distinguish cat from dog
- Questionable importance of semantics vs low-level features (assuming we want semantics)
  - Features are potentially not invariant to transformations

[“AET vs. AED: Unsupervised representation learning by auto-encoding transformations rather than data”, Zhang et al. 19]
Reconstruction

Slide Credit: Relja Arandjelovic
Denoising autoencoders

What is the noise and what is the signal?
Recognizing the digit helps!

Pros
- Simple classical method
- Apart from representations, we get a denoiser for free

Cons
- Train-eval gap: training on noisy data
- Too easy, no need for semantics – low level cues are sufficient

[“Extracting and composing robust features with denoising autoencoders”, Vincent et al. 08]
Context encoders

What goes in the middle?

["Context encoders: Feature learning by inpainting", Pathak et al. 16]
Context encoders

What goes in the middle? **Much easier if you recognize the objects!**

Natural language processing (e.g. word2vec, BERT)

All [MASK] have tusks. ⇒ All **elephants** have tusks.

[“Distributed representations of words and phrases and their compositionality”, Mikolov et al. 13]
[“BERT: Pre-training of deep bidirectional transformers for language understanding”, Devlin et al. 18]
Context encoders

Pros
- Requires preservation of fine-grained information

Cons
- Train-eval gap: no masking at eval
- Reconstruction is too hard and ambiguous
- Lots of effort spent on “useless” details: exact colour, good boundary, etc

[“Context encoders: Feature learning by inpainting”, Pathak et al. 16]
Colorization

What is the colour of every pixel? Hard if you don’t recognize the object!

Slide Credit: Relja Arandjelovic
Context encoders

![Diagram of context encoders]

Pros
- Requires preservation of fine-grained information

Cons
- Reconstruction is too hard and ambiguous
- Lots of effort spent on “useless” details: exact colour, good boundary, etc
- Forced to evaluate on greyscale images, losing information
**Context encoders $\Rightarrow$ Split-brain encoders**

![Diagram showing the process of converting grayscale and color channels into predicted grayscale and color channels.](image)

**Pros**
- Requires preservation of fine-grained information

**Cons**
- Reconstruction is too hard and ambiguous
- Lots of effort spent on “useless” details: exact colour, good boundary, etc
- Forced to evaluate on grayscale images, losing information
- Processes different chunks of the input independently
Predicting bag-of-words

Bag-of-words reminder

Extract features:
use a pre-trained self-supervised convnet $\Phi(\cdot)$

image $x$  \quad Pre-trained $\Phi(\cdot)$  \quad feature map

assign features to visual words

$k$-means clustering

vocabulary of visual words

Histogram

Bag-of-Words (BoW)

- Loses low-level details
- Encodes mid/high-level concepts

[“Learning representations by predicting bags of visual words”, Gidaris et al. 20]
Predicting bag-of-words

Inspired by NLP: targets = discrete concepts (words)

Slide Credit: Relja Arandjelovic
Predicting bag-of-words

Pros
- Representations are invariant to desired transformations
- Learn contextual reasoning skills
  - Infer words of missing image regions

Cons
- Requires bootstrapping from another network
  - e.g. hard to learn more fine-grained features
- Pitfalls of BoW
  - (partial) loss of spatial information
  - SpatialBoW not improving

[“Learning representations by predicting bags of visual words”, Gidaris et al. 20]
Instance classification
Exemplar ConvNets

This is a distorted crop extracted from an image, which of these crops has the same source image?

[Image of various crops]

[“Discriminative unsupervised feature learning with exemplar convolutional neural networks”, Dosovitskiy et al. 14]
Exemplar ConvNets

This is a distorted crop extracted from an image, which of these crops has the same source image?

Easy if robust to the desired transformations (geometry and colour)

["Discriminative unsupervised feature learning with exemplar convolutional neural networks", Dosovitskiy et al. 14]
Exemplar ConvNets

Classification into K "classes" (source images)

Pros
- Representations are invariant to desired transformations
- Requires preservation of fine-grained information

Cons
- Choosing the augmentations is important
- Exemplar based: images of the same class or instance are negatives
  - Nothing prevents it from focusing on the background
- Original formulation is not scalable (number of "classes" = dataset size)

["Discriminative unsupervised feature learning with exemplar convolutional neural networks", Dosovitskiy et al.14]
Exemplar ConvNets via metric learning

Exemplar ConvNets are not scalable (number of "classes" = number of training images)

- Reformulate in terms of metric learning
- Traditional losses such as contrastive or triplet ["Multi-task self-supervised visual learning", Doersch and Zisserman 17], ["HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips", Miech et al. 19]
- Recently popular: InfoNCE ["Representation Learning with Contrastive Predictive Coding", van den Oord et al. 18]
  - Used by many recent methods: CPC, AMDIM, SimCLR, MoCo, ..
Noise Contrastive Estimation

InfoNCE loss (a specific popular version)
- For query, positive and negative:
\[ -\log \frac{\exp(q^T p)}{\exp(q^T p) + \sum_{n \in N(q)} \exp(q^T n)} \]
- Like a ranking loss: \((q, p)\) should be close, \((q, n)\) should be far
- An implementation
  \[ \text{logits} = [q^T p, q^T n_1, q^T n_2, ..] = q^T [p, n_1, n_2, ..] \]
  \[ \text{labels} = [1, 0, 0, ..] \]
  \[ \text{InfoNCE} = \text{cross-entropy}(\text{softmax(logits)}, \text{labels}) \]
- Squint and see classification loss
  - Replace \([p, n_1, n_2, ..]\) with \([w_p, w_{n_1}, w_{n_2}, ..]\)
  - Like classification with weight-exemplars
- More details and perspectives in the next part

Classification
\[ [1, 0, 0, ..] \] \[ [1, 0, 0, ..] \] \[ [0, 1, 0, ..] \]

Metric learning

Network

Network

Network

p

q

n \in N(q)
Contrastive predictive coding (CPC)

Roughly: Context Prediction + Exemplar ConvNets

- From a patch, predict representations of other patches below it
- Use InfoNCE loss to contrast the (predictions, correct, negatives)
  - Negatives: other patches from the same image and other images
Contrastive predictive coding (CPC)

Pros
- Generic framework easily applied to images, video, audio, NLP, ..
- Exemplar: Requires preservation of fine-grained information
- Context prediction: Should enable learning about object parts

Cons
- Exemplar based: images of the same class or instance are negatives
- Train-eval gap: training on patches, evaluating on images
- Assumes training images are photographed with canonical orientations (and canonical orientations exist)
- Somewhat slow training due to dividing into patches

["Representation Learning with Contrastive Predictive Coding", van den Oord et al. 18]
["Data-efficient image recognition with contrastive prediction coding", Hénaaff et al. 19]
Exploiting time
Watching objects move

Which pixels will move?

["Learning features by watching objects move", Pathak et al. 16]
Watching objects move

Which pixels will move? Easy if we can segment objects!
Watching objects move

Mask prediction (pixel-wise logistic regression)

Network

Pros
- Emerging behaviour: segmentation
- No train-eval gap

Cons
- "Blind spots": stationary objects
- Potential focus on large salient objects
- Depends on an external motion segmentation algorithm
- Cannot be extended to temporal nets (pretext task would be trivial)

["Learning features by watching objects move", Pathak et al. 16]

Slide Credit: Relja Arandjelovic
Tracking by colorization

Given an earlier frame, colourize the new one.
Tracking by colorization

Given an earlier frame, colourize the new one. Easy if everything can be tracked!

"Tracking emerges by colorizing videos", Vondrick et al. 18
Tracking by colorization

Pros
- Emerging behaviour: tracking, matching, optical flow, segmentation

Cons
- Low level cues are effective – less emphasis on semantics
- Forced to evaluate on greyscale frames, losing information

[“Tracking emerges by colorizing videos”, Vondrick et al. 18]
Temporal ordering

Is this sequence of frames correctly ordered?
Temporal ordering

Is this sequence of frames correctly ordered? Easy if we recognize the action and human pose!

[Image of two sequences of frames, one correct and one incorrect, with green checkmark and red X]

["Shuffle and learn: Unsupervised learning using temporal order verification", Misra et al. 16]
Temporal ordering

Pros
- No train-eval gap
- Learns to recognize human pose

Cons
- Mostly focuses on human pose – not always sufficient
- Questionable if it can be extended to temporal nets (potentially task becomes too easy)

Extensions
- N frames with one randomly placed – find it
  ["Self-supervised video representation learning with odd-one-out networks", Fernando et al. 16]
- Ranking loss: embeddings should be similar for frames close in time and dissimilar for far away frames
  ["Time-contrastive networks: Self-supervised learning from video", Sermanet et al. 17]
Audio-visual correspondence

Does the sound go with the image?
Audio-visual correspondence

Does the sound go with the image? Easy if we recognize what is happening in both the frame and the audio

Slide Credit: Relja Arandjelovic
Audio-visual correspondence

[“Look, Listen and Learn”, Arandjelović et al. 17]

Slide Credit: Relja Arandjelovic
Audio-visual correspondence

Pros
- Natural different views of the training data, no need for augmentations
- No train-eval gap
- Representations in both modalities for free

Cons
- “Blind spots”: not everything makes a sound
- Exemplar based: videos of the same class or instance are negatives
- Small output space - two cases (corresponds or not)
  - Can be improved by contrastive approaches

Slide Credit: Relja Arandjelovic
Leveraging narration

Does the narration go with the video?

(Text obtained from automatic speech recognition)
Leveraging narration

Does the narration go with the video? Easy if we recognize what is happening in the video and narrations

(Text obtained from automatic speech recognition)

Complication compared to the audio-visual case:

- Narration and visual content are less aligned
Leveraging narration

Multiple instance learning extension of the NCE loss

Pros
- Natural different views of the training data, no need for augmentations
- No train-eval gap
- Representations in both modalities for free

Cons
- “Blind spots”: not everything is mentioned in narrations
- Exemplar based: videos of the same class or instance are negatives
- Assumes a single language, potentially non-trivial to extend to more languages
Representation learning without annotations?

Many ideas are possible (and yours could be even better!):

1. Predict the future: RNNs, Video
2. Compression: Autoencoder (predict self, through clamp), representation $z$
3. Pretext tasks: predict self, before/after, missing patch, correct rotation, colorization, up-sampling, multimodal
5. Make latent space parameters $z$ meaningful, orthogonal, explicit, tuneable
6. Train using a second network: GANs - Improve quality of output images
7. The Sky is the Limit
Auto-Encoders
Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

\[ z \text{ usually smaller than } x \]  
(dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

\[ \text{Origin} \text{ally: Linear + nonlinearity (sigmoid)} \]
\[ \text{Later: Deep, fully-connected} \]
\[ \text{Later: ReLU CNN} \]
Some background first: Autoencoders

How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself

Reconstructed input data $\hat{x}$
Decoder

Features $z$
Encoder

Input data $x$

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)
Some background first: Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function: $\|x - \hat{x}\|^2$

Reconstructed input data

Doesn't use labels!

Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

Reconstructed data
Some background first: Autoencoders

After training, throw away decoder
Some background first: Autoencoders

Encoder can be used to initialize a **supervised** model

Loss function
(Softmax, etc)

Predicted Label

Classifier

Features

Fine-tune encoder jointly with classifier

Train for final task
(sometimes with small data)

Input data

Encoder

Some examples:
- bird
- plane
- dog
- deer
- truck

Lecture 11 -

dog
deer
truck
bird
plane
Some background first: Autoencoders

Autoencoders can reconstruct data, and can learn features to initialize a supervised model.

Features capture factors of variation in training data. Can we generate new images from an autoencoder?
Representation learning without annotations?

Many ideas are possible (and yours could be even better!):

1. Predict the future: RNNs, Video
2. Compression: Autoencoder (predict self, through clamp), representation $z$
3. Pretext tasks: predict self, before/after, missing patch, correct rotation, colorization, up-sampling, multimodal
5. Make latent space parameters $z$ meaningful, orthogonal, explicit, tuneable
6. Train using a second network: GANs - Improve quality of output images
7. The Sky is the Limit
Variational AutoEncoders (VAEs)
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data \( \{x^{(i)}\}_{i=1}^{N} \) is generated from underlying unobserved (latent) representation \( z \)

Sample from true conditional

\[ p_{\theta^*}(x \mid z^{(i)}) \]

Sample from true prior

\[ p_{\theta^*}(z) \]

Intuition (remember from autoencoders!):
\( x \) is an image, \( z \) is latent factors used to generate \( x \): attributes, orientation, etc.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model.

**How should we represent this model?**

Choose prior $p(z)$ to be simple, e.g. Gaussian.

Conditional $p(x|z)$ is complex (generates image) => represent with neural network

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Sample from true conditional
$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior
$p_{\theta^*}(z)$

Decoder network

$x$

$z$

We want to estimate the true parameters $\theta^*$ of this generative model.

How to train the model?

Remember strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x \mid z)dz$$

Now with latent $z$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model.

How to train the model?

Remember strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

Q: What is the problem with this?

Intractable!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Since we’re modeling probabilistic generation of data, encoder and decoder networks are probabilistic Mean and

(diagonal) covariance of $z \mid x$

Encoder network

$q_\phi(z \mid x)$

(parameters $\phi$)

$\mathcal{X}$

Mean and (diagonal) covariance of $x \mid z$

Decoder network

$p_\theta(x \mid z)$

(parameters $\theta$)

$\mathcal{Z}$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic.

Sample \( z \mid x \sim \mathcal{N}(\mu_z \mid x, \Sigma_z \mid x) \)

Encoder network

\( q_\phi(z \mid x) \) (parameters \( \phi \))

\( x \)

Sample \( x \mid z \sim \mathcal{N}(\mu_x \mid z, \Sigma_x \mid z) \)

Decoder network

\( p_\theta(x \mid z) \) (parameters \( \theta \))

\( z \)

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Since we’re modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Sample $z$ from $z \mid x \sim \mathcal{N}(\mu_z \mid x, \Sigma_z \mid x)$

Encoder network
$q_\phi(z \mid x)$
(parameters $\phi$)

$x$

Decoder network
$p_\theta(x \mid z)$
(parameters $\theta$)

Sample $x \mid z$ from $x \mid z \sim \mathcal{N}(\mu_x \mid z, \Sigma_x \mid z)$

Encoder and decoder networks also called “recognition”/“inference” and “generation” networks

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}$$

$$= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}$$

$$= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}$$

$$= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \ || \ p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) \ || \ p_\theta(z | x^{(i)}))$$

The expectation wrt. $z$ (using encoder network) let us write nice KL terms
Variational Autoencoders

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right]
\]

\( (p_\theta(x^{(i)}) \) Does not depend on \( z \) \)

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)}{p_\theta(z | x^{(i)})} q_\phi(z | x^{(i)}) \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))
\]

Decoder network gives \( p_\theta(x|z) \), can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and \( z \) prior) has nice closed-form solution!

\( p_\theta(z|x) \) intractable (saw earlier), can’t compute this KL term :( But we know KL divergence always \( >= 0 \).
Variational Autoencoders

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad \text{(}\ p_\theta(x^{(i)}) \text{ } \text{Does not depend on } z\text{)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)})) + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right]
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi) \geq 0
\]

We want to maximize the data likelihood

Tractable lower bound which we can take gradient of and optimize! (\ p_\theta(x|z) \text{ differentiable, KL term differentiable)
Variational Autoencoders

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z) q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi) = \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \text{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z | x^{(i)})) + \text{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z | x^{(i)})) > 0
\]

\[
\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)
\]

Variational lower bound (“ELBO”)

Reconstruct the input data

Make approximate posterior distribution close to prior

Training: Maximize lower bound

\[\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)\]
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

\[ \mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_{\theta}(z)) \]

\[ \mathcal{L}(x^{(i)}, \theta, \phi) \]

Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!

Maximize likelihood of original input being reconstructed

Sample \( x|z \) from \( x|z \sim \mathcal{N}(\mu_x|z, \Sigma_x|z) \)

Decoder network \( p_{\theta}(x|z) \)

Sample \( z \) from \( z|x \sim \mathcal{N}(\mu_z|x, \Sigma_z|x) \)

Encoder network \( q_{\phi}(z|x) \)

Input Data \( x \)
Variational Autoencoders: Generating Data!

Use decoder network. Now sample $z$ from prior!

$\hat{x}$

Sample $x|z$ from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

$\mu_{x|z}$

$\Sigma_{x|z}$

Decoder network $p_\theta(x|z)$

Sample $z$ from $z \sim \mathcal{N}(0, I)$

Data manifold for 2-d $z$

Vary $z_1$

Vary $z_2$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Slides: Fei-Fei Li, Justin Johnson, Serena Yeung

May 9, 2019
Variational Autoencoders: Generating Data!

Diagonal prior on $\mathbf{z}$
$\Rightarrow$ independent latent variables

Different dimensions of $\mathbf{z}$ encode interpretable factors of variation

Also good feature representation that can be computed using $q_\phi(z|x)$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Generating Data!

32x32 CIFAR-10

Labeled Faces in the Wild

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a (variational) lower bound

Pros:
- Principled approach to generative models
- Allows inference of q(z|x), can be useful feature representation for other tasks

Cons:
- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active areas of research:
- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs)
- Incorporating structure in latent variables, e.g., Categorical Distributions
Representation learning **without annotations?**

Many ideas are possible (and yours could be even better!):

1. Predict the future: RNNs, Video
2. Compression: Autoencoder (predict self, through clamp), representation \( z \)
3. Pretext tasks: predict self, before/after, missing patch, correct rotation, colorization, up-sampling, multimodal
5. Make latent space parameters \( z \) meaningful, orthogonal, explicit, tuneable
6. Train using a second network: GANs - Improve quality of output images
7. The Sky is the Limit
GANs:
Generative Adversarial Networks
Generative Adversarial Networks

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

Input: Random noise

Output: Sample from training distribution

Generator Network

| z |

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Random noise $z$ is fed into the generator network, which creates fake images that are then fed into the discriminator network. The discriminator tries to determine whether the images are real or fake. If the discriminator is able to distinguish between real and fake images, the generator network learns to create more realistic images.

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images  
**Discriminator network**: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]

- Discriminator ($\theta_d$) wants to **maximize objective** such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- Generator ($\theta_g$) wants to **minimize objective** such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

   $$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_data} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_data} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

   $$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.
Training GANs: Two-player game

Putting it together: GAN training algorithm

for number of training iterations do
  for k steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
      $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by ascending its stochastic gradient (improved objective):
    $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$
end for

Some find $k=1$ more stable, others use $k > 1$, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

After training, use generator network to generate new images

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Mean Squared Error Can Ignore Small but Task-Relevant Features

Figure 15.5

The ping pong ball vanishes because it is not large enough to significantly affect the mean squared error.
Adversarial Losses Preserve Any Features with Highly Structured Patterns

Ground Truth    MSE    Adversarial

Figure 15.6
Mean squared error loses the ear because it causes a small change in few pixels. Adversarial loss preserves the ear because it is easy to notice its absence.

Slide Credit: Ian Goodfellow (Goodfellow 2017)
Generative Adversarial Nets

Generated samples

Nearest neighbor from training set

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.
Generative Adversarial Nets

Generated samples (CIFAR-10)

Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
GANs + CNNs: Convolutional Architectures for Generative Adversarial Networks
Generative Adversarial Nets: Convolutional Architectures

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Generative Adversarial Nets: Convolutional Architectures

Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

Radford et al, ICLR 2016
Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space

Radford et al, ICLR 2016
Generative Adversarial Nets: Interpretable Vector Math

Smiling woman   Neutral woman   Neutral man

Samples from the model

Radford et al, ICLR 2016
Generative Adversarial Nets: Interpretable Vector Math

Samples from the model

Average Z vectors, do arithmetic

Radford et al, ICLR 2016
Samples from the model

Average Z vectors, do arithmetic

Smiling woman Neutral woman Neutral man

Smiling Man

Radford et al, ICLR 2016
Generative Adversarial Nets: Interpretable Vector Math

Glasses man  No glasses man  No glasses woman

Radford et al, ICLR 2016
Generative Adversarial Nets: Interpretable Vector Math

Glasses man  No glasses man  No glasses woman  

Radford et al, ICLR 2016

Woman with glasses

Radford et al, ICLR 2016
Next-Generation GANs
2017: Explosion of GANs

“The GAN Zoo”

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Networks
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-image Translation using Cycle-Consistent Adversarial Networks
- DNT - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAIWNN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Inversive Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- ICGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo
2017: Explosion of GANs

"The GAN Zoo"

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired image-to-image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWIN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Inretrospective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-posing Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo
2017: Explosion of GANs

Better training and generation


Improved Wasserstein GAN, Gulrajani 2017.

Progressive GAN, Karras 2018.
2017: Explosion of GANs
Source->Target domain transfer


Text -> Image Synthesis
this small bird has a pink breast and crown, and black primaries and secondaries. this magnificent fellow is almost all black with a red crest, and white cheek patch.

Reed et al. 2017.
Many GAN applications

2019: BigGAN

Brock et al., 2019
Style GANs

Figure 8. The effect of truncation trick as a function of style scale $\psi$. When we fade $\psi \rightarrow 0$, all faces converge to the “mean” face of FFHQ. This face is similar for all trained networks, and the interpolation towards it never seems to cause artifacts. By applying negative scaling to styles, we get the corresponding opposite or “anti-face”. It is interesting that various high-level attributes often flip between the opposites, including viewpoint, glasses, age, coloring, hair length, and often gender.

Figure 3. Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A. Copying the styles corresponding to coarse spatial resolutions ($4^2$ – $8^2$) brings high-level aspects such as pose, general hair style, face shape, and eyeglasses from source B, while all colors (eyes, hair, lighting) and finer facial features resemble A. If we instead copy the styles of middle resolutions ($16^2$ – $32^2$) from B, we inherit smaller scale facial features, hair style, eyes open/closed from B, while the pose, general face shape, and eyeglasses from A are preserved. Finally, copying the fine styles ($64^2$ – $1024^2$) from B brings mainly the color scheme and microstructure.
Representation learning **without annotations?**

Many ideas are possible (and yours could be even better!):

1. Predict the future: RNNs, Video
2. Compression: Autoencoder (predict self, through clamp), representation \( z \)
3. Pretext tasks: predict self, before/after, missing patch, correct rotation, colorization, up-sampling, multimodal
5. Make latent space parameters \( z \) meaningful, orthogonal, explicit, tuneable
6. Train using a second network: GANs - Improve quality of output images
7. The Sky is the Limit
Taxonomy of Generative Models

- Generative models
  - Explicit density
  - Implicit density
    - Markov Chain
  - Approximate density
    - Variational
      - Variational Autoencoder
    - Markov Chain
      - Boltzmann Machine
  - Fully Visible Belief Nets
    - NADE
    - MADE
    - PixelRNN/CNN
    - NICE / RealNVP
    - Glow
    - Fjord

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
GANs $\leftrightarrow$ VAEs
Let’s look at VAE and GAN more closely…

From Eric Xing’s slides for CMU 10-708
Based on “On Unifying Deep Generative Models”
Variational Autoencoders (VAEs)

- [Kingma & Welling, 2014]
- Use variational inference with an inference model
  - Enjoy similar applicability with wake-sleep algorithm
- Generative model $p_{\theta}(x|z)$, and prior $p(z)$
  - Joint distribution $p_{\theta}(x,z) = p_{\theta}(x|z)p(z)$
- Inference model $q_{\phi}(z|x)$

Figure courtesy: Kingma & Welling, 2014
Generative Adversarial Nets (GANs)

- [Goodfellow et al., 2014]
- Generative model $x = G_\theta(z), \ z \sim p(z)$
  - Map noise variable $z$ to data space $x$
  - Define an implicit distribution over $x$: $p_{g_\theta}(x)$
    - a stochastic process to simulate data $x$
    - Intractable to evaluate likelihood
- Discriminator $D_\phi(x)$
  - Output the probability that $x$ came from the data rather than the generator
  - No explicit inference model
  - No obvious connection to previous models with inference networks like VAEs
    - We will build formal connections between GANs and VAEs later
A unified view of deep generative models

- Literatures have viewed these DGM approaches as distinct model training paradigms
  - GANs: achieve an equilibrium between generator and discriminator
  - VAEs: maximize lower bound of the data likelihood

- Let's study a new formulation for DGMs
  - Connects GANs, VAEs, and other variants, under a unified view
  - Links them back to inference and learning of Graphical Models, and the wake-sleep heuristic that approximates this
  - Provides a tool to analyze many GAN-/VAE-based algorithms
  - Encourages mutual exchange of ideas from each individual class of models
Generative Adversarial Nets (GANs):

- Implicit distribution over $x \sim p_\theta(x|y)$
  
  \[ p_\theta(x|y) = \begin{cases} 
  p_{g_\theta}(x) & y = 0 \\
  p_{data}(x) & y = 1. 
  \end{cases} \]
  
  (distribution of generated images)
  (distribution of real images)

- $x \sim p_{g_\theta}(x) \iff x = G_\theta(z), \; z \sim p(z|y = 0)$

- $x \sim p_{data}(x)$
  - the code space of $z$ is degenerated
  - sample directly from data
A new formulation

- Rewrite GAN objectives in the "variational-EM" format

Recap: conventional formulation:

$$\begin{align*}
\max_\phi \mathcal{L}_\phi &= \mathbb{E}_{x = G_\theta(z), z \sim p(z|y=0)} [\log (1 - D_\phi(x))] + \mathbb{E}_{x \sim p_{data}(x)} [\log D_\phi(x)] \\
\max_\theta \mathcal{L}_\theta &= \mathbb{E}_{x = G_\theta(z), z \sim p(z|y=0)} [\log D_\phi(x)] + \mathbb{E}_{x \sim p_{data}(x)} [\log (1 - D_\phi(x))] \\
&= \mathbb{E}_{x = G_\theta(z), z \sim p(z|y=0)} [\log D_\phi(x)]
\end{align*}$$

- Rewrite in the new form
  - Implicit distribution over $x \sim p_\theta(x|y)$
    
      $x = G_\theta(z), \ z \sim p(z|y)$

  - Discriminator distribution $q_\phi(y|x)$
    
    $q_\phi^r(y|x) = q_\phi(1 - y|x)$ (reverse)

$$\begin{align*}
\max_\phi \mathcal{L}_\phi &= \mathbb{E}_{p_\theta(x|y)p(y)} [\log q_\phi(y|x)] \\
\max_\theta \mathcal{L}_\theta &= \mathbb{E}_{p_\theta(x|y)p(y)} [\log q_\phi^r(y|x)]
\end{align*}$$
GANs vs. Variational EM

**Variational EM**
- Objectives
  - \( \max_{\phi} \mathcal{L}_{\phi,\theta} = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x)||p(z)) \)
  - \( \max_{\theta} \mathcal{L}_{\phi,\theta} = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x)||p(z)) \)
  - Single objective for both \( \theta \) and \( \phi \)
  - Extra prior regularization by \( p(z) \)
  - **The reconstruction term:** maximize the conditional log-likelihood of \( x \) with the generative distribution \( p_{\theta}(x|z) \) conditioning on the latent code \( z \) inferred by \( q_{\phi}(z|x) \)
  - \( p_{\theta}(x|z) \) is the generative model
  - \( q_{\phi}(z|x) \) is the inference model

**GAN**
- Objectives
  - \( \max_{\phi} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(x|y)p(y)}[\log q_{\phi}(y|x)] \)
  - \( \max_{\theta} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(x|y)p(y)}[\log q_{\phi}(y|x)] \)
  - Two objectives
  - Have global optimal state in the game theoretic view
  - The objectives: maximize the conditional log-likelihood of \( y \) (or \( 1 - y \)) with the distribution \( q_{\phi}(y|x) \) conditioning on data/generation \( x \) inferred by \( p_{\theta}(x|y) \)
  - Interpret \( q_{\phi}(y|x) \) as the generative model
  - Interpret \( p_{\theta}(x|y) \) as the inference model
GANs vs. Variational EM

Variational EM
- Objectives

\[
\max_{\phi} \mathcal{L}_{\phi, \theta} = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) \| p(z))
\]
\[
\max_{\theta} \mathcal{L}_{\phi, \theta} = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) \| p(z))
\]
- Single objective for both \(\theta\) and \(\phi\)
- Extra prior regularization by \(p(z)\)
- The reconstruction term: maximize the conditional log-likelihood of \(x\) with the generative distribution \(p_{\theta}(x|z)\) conditioning on the latent code \(z\) inferred by \(q_{\phi}(z|x)\)
- \(p_{\theta}(x|z)\) is the generative model
- \(q_{\phi}(z|x)\) is the inference model

GAN
- Objectives

\[
\max_{\phi} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(x|y)p(y)}[\log q_{\phi}(y|x)]
\]
\[
\max_{\theta} \mathcal{L}_{\theta} = \mathbb{E}_{p_{\theta}(x|y)p(y)}[\log q_{\phi}^*(y|x)]
\]
- Two objectives
- Have global optimal state in the game theoretic view
- The objectives: maximize the conditional log-likelihood of \(y\) (or \(1 - y\)) with the distribution \(q_{\phi}(y|x)\) conditioning on data/generation \(x\) inferred by \(p_{\theta}(x|y)\)
- Interpret \(q_{\phi}(y|x)\) as the generative model
- Interpret \(p_{\theta}(x|y)\) as the inference model
GANs: minimizing KLD

- As in Variational EM, we can further rewrite in the form of minimizing KLD to reveal more insights into the optimization problem.
- For each optimization step of $p_\theta(x|y)$ at point ($\theta = \theta_0, \phi = \phi_0$), let
  - $p(y)$: uniform prior distribution
  - $p_{\theta=\theta_0}(x) = E_{p(y)}[p_{\theta=\theta_0}(x|y)]$
  - $q^r(x|y) \propto q^{r=\phi_0}(y|x)p_{\theta=\theta_0}(x)$

**Lemma 1:** The updates of $\theta$ at $\theta_0$ have

$$\nabla_\theta \left[ - E_{p_\theta(x|y)p(y)} \left[ \log q^{r=\phi_0}(y|x) \right] \right]_{\theta=\theta_0} =$$

$$\nabla_\theta \left[ E_{p(y)} \left[ KL \left( p_\theta(x|y) \| q^r(x|y) \right) \right] - JSD \left( p_\theta(x|y=0) \| p_\theta(x|y=1) \right) \right]_{\theta=\theta_0}.$$

- KL: KL divergence
- JSD: Jensen-Shannon divergence
GANs: minimizing KLD

- **Lemma 1:** The updates of $\theta$ at $\theta_0$ have
  \[
  \nabla_\theta \left[ - \mathbb{E}_{p_\theta(x|y)p(y)} \left[ \log q_{\phi=\theta_0}^r(y|x) \right] \right]_{\theta=\theta_0} = \\
  \nabla_\theta \left[ \mathbb{E}_{p(y)} \left[ \text{KL} (p_\theta(x|y) \| q^r(x|y)) \right] - \text{JSD} (p_\theta(x|y=0) \| p_\theta(x|y=1)) \right]_{\theta=\theta_0}
  \]

- **Connection to variational inference**
  - See $x$ as latent variables, $y$ as visible
  - $p_{\theta=\theta_0}(x)$: prior distribution
  - $q^r(x|y) \propto q_{\phi=\theta_0}^r(y|x)p_{\theta=\theta_0}(x)$: posterior distribution
  - $p_\theta(x|y)$: variational distribution
    - Amortized inference: updates model parameter $\theta$

- **Suggests relations to VAEs, as we will explore shortly**
  
  In VEM, we minimize the following:
  \[
  F(\theta, \phi; x) = -\log p(x) + \text{KL} \left( q_\phi(z|x) \| p_\theta(z|x) \right)
  \]
  \[
  \Rightarrow \text{KL (inference model \| posterior)}
  \]

© Eric Xing @ CMU, 2005-2020
GANs: minimizing KLD

$p_{\theta=\theta_0}(x|y=1) = p_{\text{data}}(x)$
$p_{\theta=\theta_0}(x|y=0) = p_{g_{\theta}=\theta_0}(x)$

$q^r(x|y=0)$

$q^r(x|y=0) \propto q_{\phi=\phi_0}^r(y=0|x)p_{\theta=\theta_0}(x)$

Minimizing the KLD drives $p_{g_{\theta}}(x)$ to $p_{\text{data}}(x)$

- By definition: $p_{\theta=\theta_0}(x) = E_{p(y)}[p_{\theta=\theta_0}(x|y)] = \left(p_{g_{\theta}=\theta_0}(x) + p_{\text{data}}(x)\right) / 2$
- KL($p_{\theta}(x|y=1)||q^r(x|y=1)$) = KL($p_{\text{data}}(x)||q^r(x|y=1)$) : constant, no free parameters
- KL($p_{\theta}(x|y=0)||q^r(x|y=0)$) = KL($p_{g_{\theta}}(x)||q^r(x|y=0)$) : parameter $\theta$ to optimize

- seen as a mixture of $p_{g_{\theta}=\theta_0}(x)$ and $p_{\text{data}}(x)$
- mixing weights induced from $q_{\phi=\phi_0}^r(y=0|x)$
- Drives $p_{g_{\theta}}(x|y)$ to mixture of $p_{g_{\theta}=\theta_0}(x)$ and $p_{\text{data}}(x)$
  $\Rightarrow$ Drives $p_{g_{\theta}}(x)$ to $p_{\text{data}}(x)$
GANs: minimizing KLD

\[ p_{\theta=\theta_0}(x|y = 1) = p_{\text{data}}(x) \quad p_{\theta=\theta_0}(x|y = 0) = p_{g_{\theta=\theta_0}}(x) \]

\[ q^r(x|y = 0) \]

\[ p_{\theta=\theta_{\text{new}}}(x|y = 0) = p_{g_{\theta=\theta_{\text{new}}}}(x) \]

• Missing mode phenomena of GANs
  • Asymmetry of KLD
    • Concentrates \( p_\theta(x|y = 0) \) to large modes of \( q^r(x|y) \)
      \[ \Rightarrow p_{g_\theta}(x) \text{ misses modes of } p_{\text{data}}(x) \]
  • Symmetry of JSD
    • Does not affect the behavior of mode missing

\[
\text{KL} \left( p_\theta(x) \| q^r(x|y = 0) \right) \\
= \int p_\theta(x) \log \frac{p_\theta(x)}{q^r(x|y = 0)} \, dx
\]

• Large positive contribution to the KLD in the regions of \( x \) space where \( q^r(x|y = 0) \) is small, unless \( p_{g_\theta}(x) \) is also small

\[ \Rightarrow p_{g_\theta}(x) \text{ tends to avoid regions where } q^r(x|y = 0) \text{ is small} \]
Recap: conventional formulation of VAEs

Objective:

$$\max_{\theta, \eta} \mathcal{L}_{\theta, \eta}^{\text{vae}} = \mathbb{E}_{p_{\text{data}}(x)} \left[ \mathbb{E}_{\tilde{q}_\eta(z|x)} \left[ \log \tilde{p}_\theta(x|z) \right] - \text{KL}(\tilde{q}_\eta(z|x)||\bar{p}(z)) \right]$$

- $\bar{p}(z)$: prior over $z$
- $\tilde{p}_\theta(x|z)$: generative model
- $\tilde{q}_\eta(z|x)$: inference model
- Only uses real examples from $p_{\text{data}}(x)$, lacks adversarial mechanism

To align with GANs, let’s introduce the real/fake indicator $y$ and adversarial discriminator
VAEs: new formulation

- Assume a *perfect* discriminator $q_*(y|x)$
  - $q_*(y = 1|x) = 1$ if $x$ is real examples
  - $q_*(y = 0|x) = 1$ if $x$ is generated samples
  - $q_*(y|x) := q_*(1 - y|x)$
- Generative distribution
  $$p_\theta(x|z, y) = \begin{cases} p_\theta(x|z) & y = 0 \\ p_{data}(x) & y = 1. \end{cases}$$

- Let $p_\theta(z, y|x) \propto p_\theta(x|z, y)p(z|y)p(y)$
- Lemma 2
  $$L_{\theta, \eta}^{\text{vae}} = 2 \cdot \mathbb{E}_{p_\theta_0(x)} \left[ \mathbb{E}_{q_\eta(z|x, y)}q_*(y|x) \left[ \log p_\theta(x|z, y) \right] - \text{KL}(q_\eta(z|x, y)q_*(y|x) \| p(z|y)p(y)) \right]$$
  $$= 2 \cdot \mathbb{E}_{p_\theta_0(x)} \left[ -\text{KL}(q_\eta(z|x, y)q_*(y|x) \| p_\theta(z, y|x)) \right].$$
## GANs vs VAEs side by side

<table>
<thead>
<tr>
<th></th>
<th>GANs (InfoGAN)</th>
<th>VAEs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generative distribution</strong></td>
<td>( p_\theta(x</td>
<td>y) = \begin{cases} p_{g_\theta}(x) &amp; y = 0 \ p_{\text{data}}(x) &amp; y = 1. \end{cases} )</td>
</tr>
<tr>
<td><strong>Discriminator distribution</strong></td>
<td>( q_\phi(y</td>
<td>x) )</td>
</tr>
<tr>
<td><strong>z-inference model</strong></td>
<td>( q_\eta(z</td>
<td>x, y) ) of InfoGAN</td>
</tr>
<tr>
<td><strong>KLD to minimize</strong></td>
<td>( \min_\theta \text{KL}(p_\theta(x</td>
<td>y) \parallel q^r(x</td>
</tr>
</tbody>
</table>
## GANs vs VAEs side by side

<table>
<thead>
<tr>
<th></th>
<th>GANs (InfoGAN)</th>
<th>VAEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLD to minimize</td>
<td>$\min_{\theta} \text{KL} (p_{\theta}(x</td>
<td>y) \parallel q^r(x</td>
</tr>
<tr>
<td></td>
<td>$\sim \min_{\theta} \text{KL}(P_{\theta} \parallel Q)$</td>
<td>$\sim \min_{\theta} \text{KL}(Q \parallel P_{\theta})$</td>
</tr>
</tbody>
</table>

- Asymmetry of KLDs inspires combination of GANs and VAEs
  - GANs: $\min_{\theta} \text{KL}(P_{\theta} \parallel Q)$ tends to missing mode
  - VAEs: $\min_{\theta} \text{KL}(Q \parallel P_{\theta})$ tends to cover regions with small values of $p_{\text{data}}$
Discriminative vs. Generative: Blurring the Distinction
Generative vs. Discriminative Classifiers

Training classifiers involves estimating $f: X \rightarrow Y$, or $P(Y|X)$

Generative classifiers:

• Assume some functional form for $P(X|Y)$, $P(X)$
• Estimate parameters of $P(X|Y)$, $P(X)$ directly from training data
• Use Bayes rule to calculate $P(Y|X=x_i)$

Discriminative classifiers:

1. Assume some functional form for $P(Y|X)$
2. Estimate parameters of $P(Y|X)$ directly from training data

Slide Credit: Tom Mitchell
• Consider learning $f: X \rightarrow Y$, where
  
  • $X$ is a vector of real-valued features, $<X_1 \ldots X_n>$
  
  • $Y$ is boolean
  
  • assume all $X_i$ are conditionally independent given $Y$
  
  • model $P(X_i | Y = y_k)$ as Gaussian $\mathcal{N}(\mu_{ik}, \sigma)$
  
  • model $P(Y)$ as binomial $(p)$

• What does that imply about the form of $P(Y|X)$?

$$P(Y = 1 | X = <x_1, \ldots x_n>) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

Slide Credit: Tom Mitchell
Logistic regression

- Logistic regression represents the probability of category $i$ using a linear function of the input variables:

$$P(Y = i \mid X = x) = g(w_{i0} + w_{i1}x_1 + \ldots + w_{id}x_d)$$

where for $i<k$

$$g(z_i) = \frac{e^{z_i}}{1 + \sum_{j=1}^{K-1} e^{z_j}}$$

and for $k$

$$g(z_k) = \frac{1}{1 + \sum_{j=1}^{K-1} e^{z_j}}$$

Slide Credit: Tom Mitchell
Generative-Discriminative Pairs

Example: assume \( Y \) boolean, \( X = \langle X_1, X_2, \ldots, X_n \rangle \), where \( x_i \) are boolean, perhaps dependent on \( Y \), conditionally independent given \( Y \)

Generative model: naïve Bayes:

\[
\hat{p}(x_i = 1|y = b) = \frac{s\{x_i = 1, y = b\} + l}{s\{y = b\} + 2l}
\]

\[
\hat{p}(y = b) = \frac{s\{y = b\}}{\sum_j s\{y = j\}}
\]

Classify new example \( x \) based on ratio

\[
\frac{\hat{p}(y = T|x)}{\hat{p}(y = F|x)} = \frac{\hat{p}(y = T) \prod_{i=1}^{n} \hat{p}(x_i|y = T)}{\hat{p}(y = F) \prod_{i=1}^{n} \hat{p}(x_i|y = F)}
\]

Equivalently, based on sign of log of this ratio

Slide Credit: Tom Mitchell
Generative-Discriminative Pairs

Example: assume Y boolean, \( X = \langle x_1, x_2, \ldots, x_n \rangle \), where \( x_i \) are boolean, perhaps dependent on \( Y \), conditionally independent given \( Y \)

Generative model: naïve Bayes:

\[
\hat{p}(x_i = 1 | y = b) = \frac{s\{x_i = 1, y = b\} + l}{s\{y = b\} + 2l}
\]

\[
\hat{p}(y = b) = \frac{s\{y = b\}}{\sum_j s\{y = j\}}
\]

Classify new example \( x \) based on ratio

\[
\frac{\hat{p}(y = T | x)}{\hat{p}(y = F | x)} = \frac{\hat{p}(y = T) \prod_{i=1}^{n} \hat{p}(x_i | y = T)}{\hat{p}(y = F) \prod_{i=1}^{n} \hat{p}(x_i | y = F)}
\]

Discriminative model: logistic regression

\[
\hat{p}(y = T | x; \beta, \theta) = \frac{1}{1 + \exp(- \sum_{i=1}^{n} \beta_i x_i - \theta)}
\]

Note both learn linear decision surface over \( X \) in this case
What is the difference asymptotically?

Notation: let $\epsilon(h_{A,m})$ denote error of hypothesis learned via algorithm A, from $m$ examples

- If assumed model correct (e.g., naïve Bayes model), and finite number of parameters, then
  \[\epsilon(h_{Dis,\infty}) = \epsilon(h_{Gen,\infty})\]

- If assumed model incorrect
  \[\epsilon(h_{Dis,\infty}) \leq \epsilon(h_{Gen,\infty})\]

Note assumed discriminative model can be correct even when generative model incorrect, but not vice versa

Slide Credit: Tom Mitchell
Some experiments from UCI data sets

Figure 1: Results of 15 experiments on datasets from the UCI Machine Learning repository. Plots are of generalization error vs. \( m \) (averaged over 1000 random train/test splits). Dashed line is logistic regression; solid line is naïve Bayes.