# The future will be unsupervised
# Open problems for Deep Generative models

Alex Dimakis
UT Austin

joint work with
Qi Lei, Shanshan Wu, Ashish Bora, Dave Van Veen Ajil Jalal,and Eric Price, UT Austin

@AlexGDimakis
Data.ece.utexas.edu (**UT Minds group**)

# Outline

- 1. Supervised learning works well*
- 2. Except when it is not
  (adversarial examples, distribution shifts)
- 3. We need unsupervised models to detect/model this.  Deep Generative models.
- 4. Some results and open problems for generative models.

# Supervised learning

Given many features x1,x2, .. $x_n$ predict a label Y.

e.g. predict if a person will develop cardiovascular desease by age 60 given height, weight, blood pressure etc….

Predict the price that an apartment will be sold given its number of bedrooms, no of bathrooms, its location, square footage etc..

Predict the probability that someone will click on an Ad given their browsing history and features of the Ad etc..

Predict from the pixels of an MRI scan if a tumor is malignant or benign.

Predict from the pixels of an image if an elderly person has fallen and needs help

# Supervised learning

| age | height | weight | BP … | Y=label |
|---|---|---|---|---|
| 24 | 175 | 83 | … | 0 |
| 35 | 182 | 123 | .. | 1 |
| … | | | | |

Given many features x1,x2, .. $x_n$ predict a label Y.

# Supervised learning

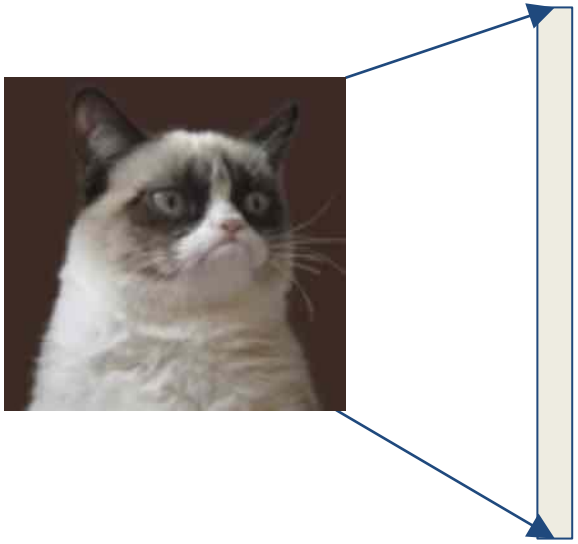
Supervised learning works really well*

# Supervised learning

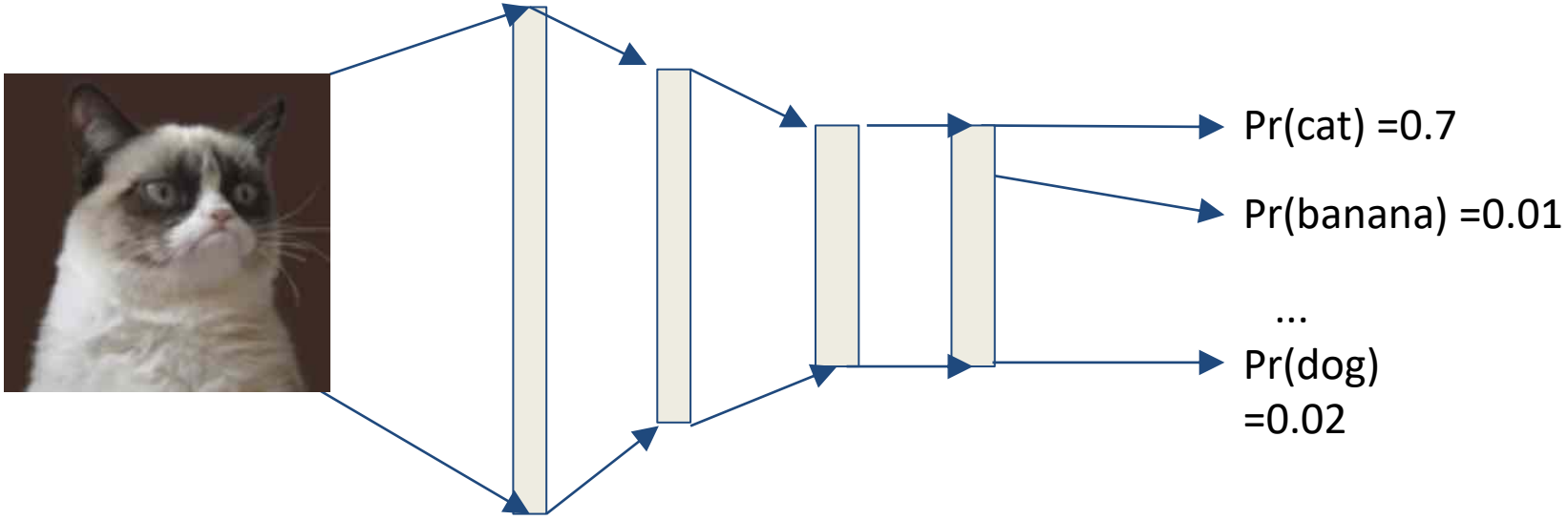
Supervised learning works really well*

If you have enough labeled data
If your data is not changing in distribution
If nobody is manipulating your features
If your task does not change

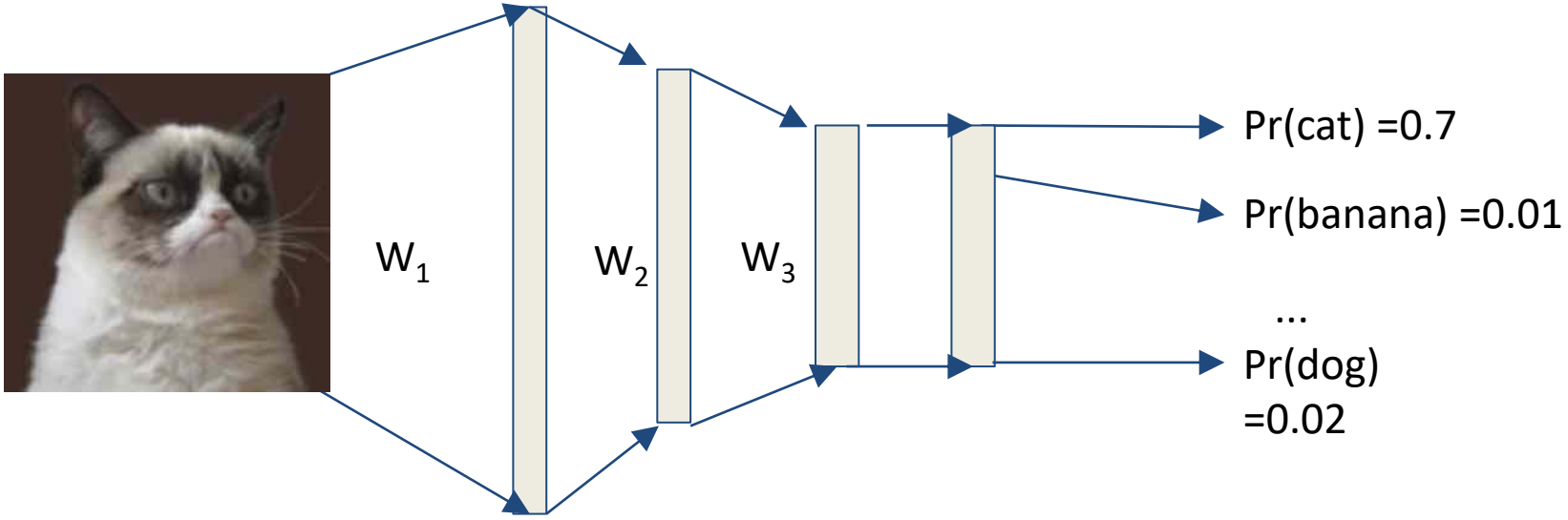
# Deep Neural nets: Classifiers



# Deep Neural nets: Classifiers



# Deep Neural nets: Classifiers

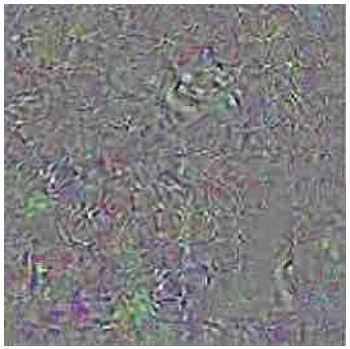



Supervised Learning= needs labeled data

# Adversarial Examples: Neural Nets are very brittle

Clean Image
(cat 99%)

+



Adversarial perturbation (**Not Random**)

=

Perturbed input
(banana 98%)

# Adversarial attacks

Can be targeted or untargeted.

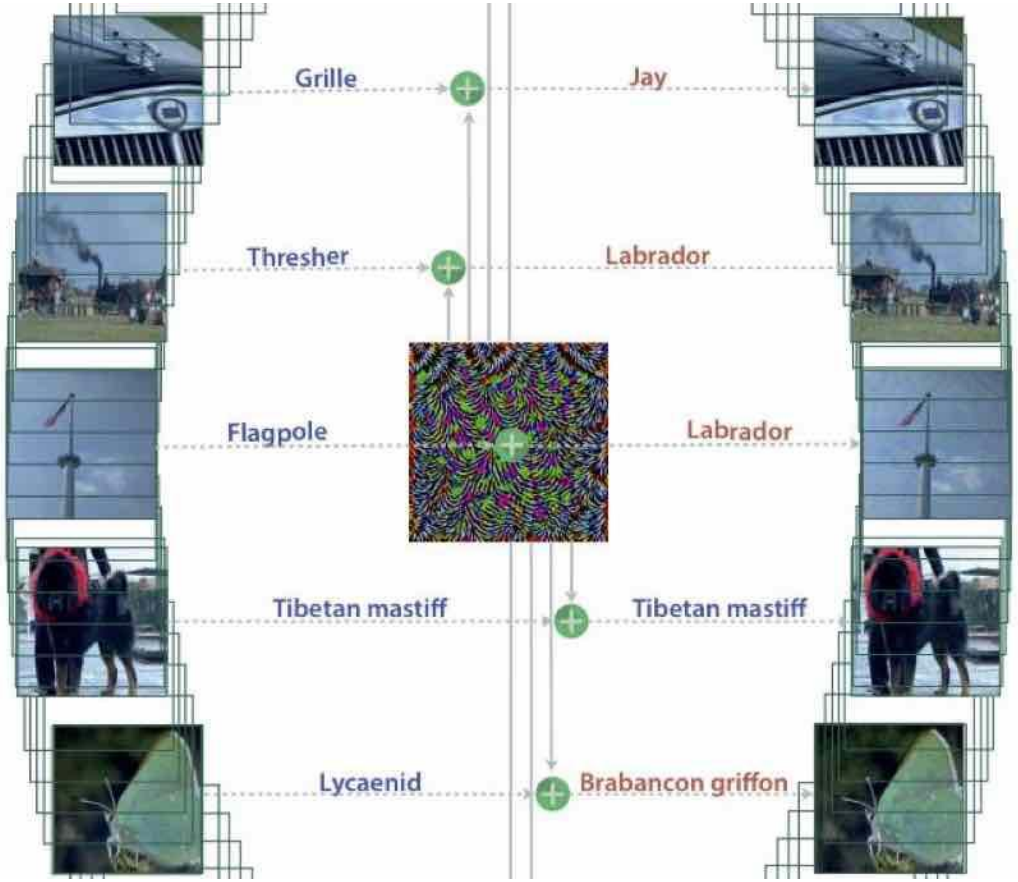
Can be created with white-box or black-box access to a classifier.

Can be **robust** to translations, adding noise, rotations, even work over 3d printing.

Can be **universal** across classes or even across models (transferability).

Can influence image, audio, text classifiers, non-neural networks ..

# Universal attacks




Moosavi-Dezfooli, S. M., Fawzi, A., Fawzi, O., & Frossard, P. (2016). Universal adversarial perturbations. arXiv preprint arXiv:1610.08401.

# 3d printed attack object


https://www.labsix.org/about/

**Synthesizing Robust Adversarial Examples**

*Anish Athalye*, Logan Engstrom*, Andrew Ilyas*, Kevin Kwok* https://arxiv.org/abs/1707.07397

# Traffic sign attack



Both classified as Speed LIMIT signs by CNNs

Evtimov et al. https://arxiv.org/pdf/1707.08945.pdf
Robust Physical-World Attacks on Deep Learning Visual Classification

# Attacking text classifiers: LSTM for sentiment analysis

- Original text:

This Starbucks location is located in the Bally's Grand Bazaar Shops. It's open 24/7 and it is huge. There is plenty of seating. Most of the seating is stadium type seating with benches. They also have an out door patio.

The staff is very friendly and attentive to the guests. I do notice that they are under staffed sometimes when they are busy. They get your drinks out pretty fast though.

Also, this location is not owned by the casino so they don't charge outrageous prices like the location on the Linq promenade does.

Definitely one of my favorite Starbucks stores. Stop by if your on the Strip.

**Model Output: 100% positive review.**

# Attacking text classifiers: LSTM for sentiment analysis

- Attacked text:

This Starbucks location is located in the Bally's Grand Bazaar Shops. It's open 24/7 and it is huge. There is plenty of seating. Most of the seating is stadium type seating with benches. They also have an out door patio.

The staff is very friendly and attentive to the guests. I do notice that they are under staffed sometimes when they are busy. They'll get your drinks pretty fast.

Also, this place is not owned by the property so they do n't charge outrageous prices as a place on an Linq promenade.

Definitely one of my favorite Starbucks stores. Stop by if your on the Strip.

**Model Output: 93% negative review.**

Discrete Adversarial Attacks and Submodular Optimization with Applications to Text Classification.
https://arxiv.org/pdf/1812.00151.pdf
SysML 2019. by Qi et al.

# The future will be unsupervised

- Unguided supervised learning leads to very brittle classifiers
- Distribution change can be due to changing environment, different camera, etc.
- We need to model high-dimensional distributions
- Learn the parameters to generate artificial data and to perform inference.

# Unsupervised learning

| age | height | weight | BP … | Y=label |
|---|---|---|---|---|
| 24 | 175 | 83 | … | 0 |
| 35 | 182 | 123 | .. | 1 |
| … | | | | |

Given many features x1,x2, .. $x_n$ ~~predict a label Y.~~
Learn the structure of the features
Generate artificial data, Clustering
Denoising, Fill-in missing data, find distribution shifts etc

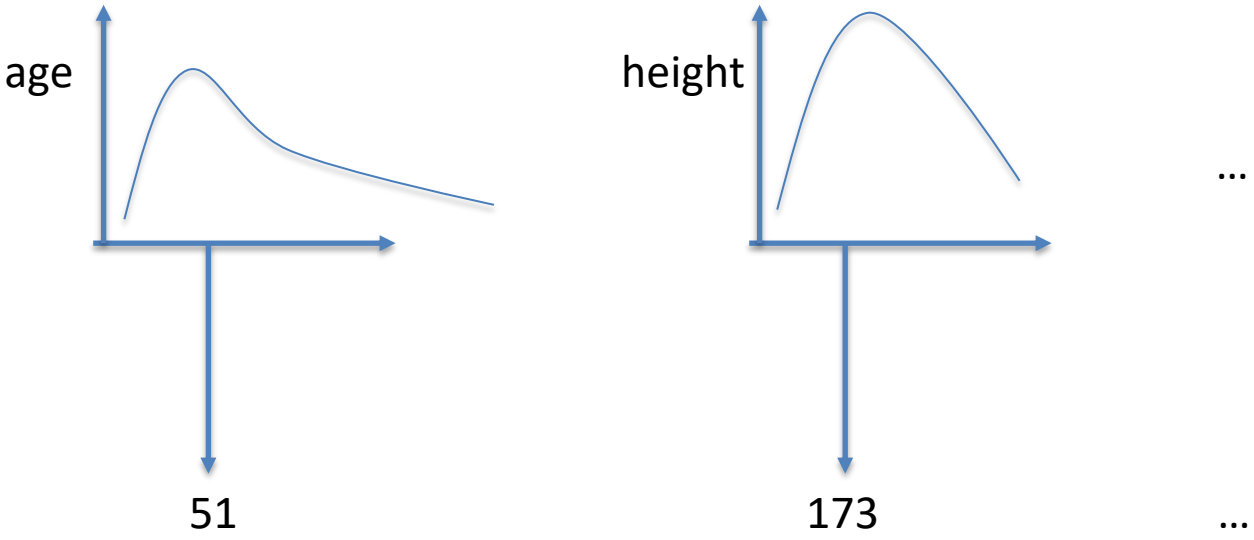
# Unsupervised learning

| age | height | weight | BP … | Y=label |
|---|---|---|---|---|
| 24 | 175 | 83 | … | 0 |
| 35 | 182 | 123 | .. | 1 |
| 44 | 12 | 2817 | … | |

Given many features x1,x2, .. $x_n$ ~~predict a label Y.~~
Learn the structure of the features
Generate artificial data, Clustering
Denoising, Fill-in missing data, find distribution shifts etc

# How to model high-dimensional distributions

Idea 0: Independent: Learn the marginal for each feature and sample independently.



# How to model high-dimensional distributions

Idea 1:  Parametric: Fit a multivariate Gaussian

Idea 2: Use conditional independence (Markov Chains and Graphical models)

Idea 3: Use sparsity in a basis.

# How to model high-dimensional distributions

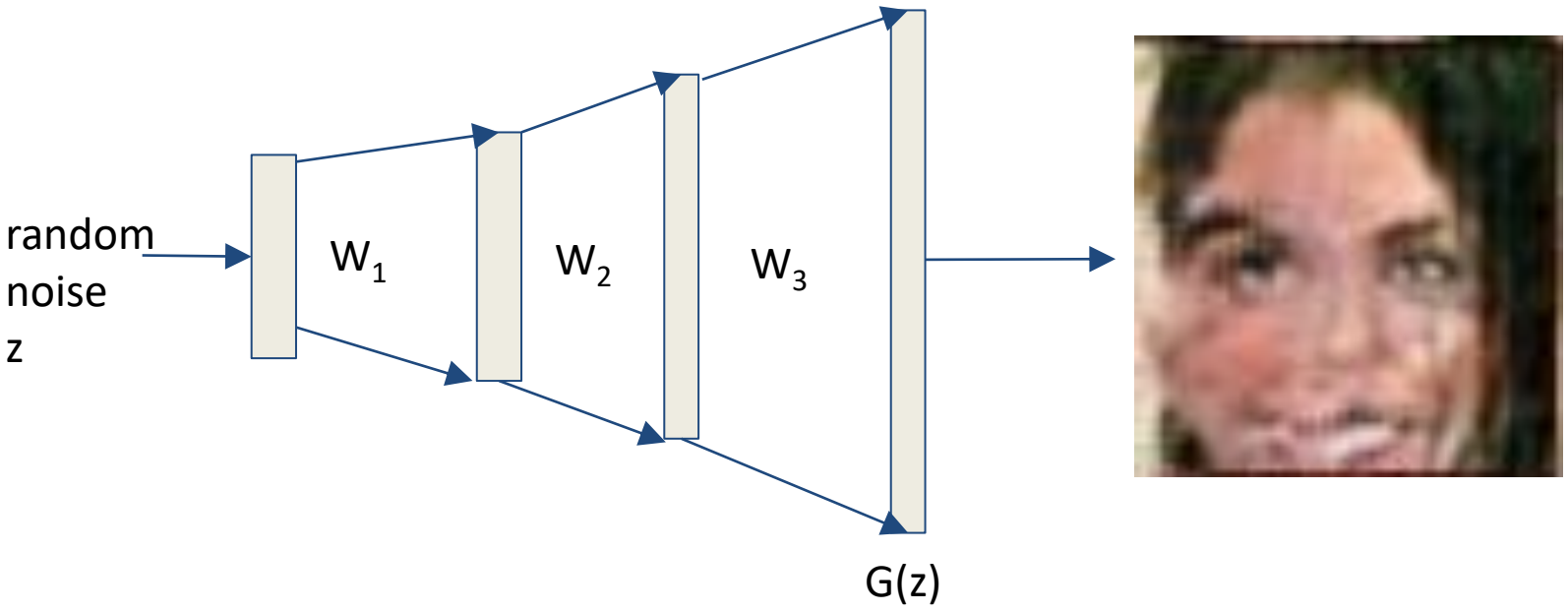
Idea 1:  Parametric: Fit a multivariate Gaussian

Idea 2: Use conditional independence (Markov Chains and Graphical models)

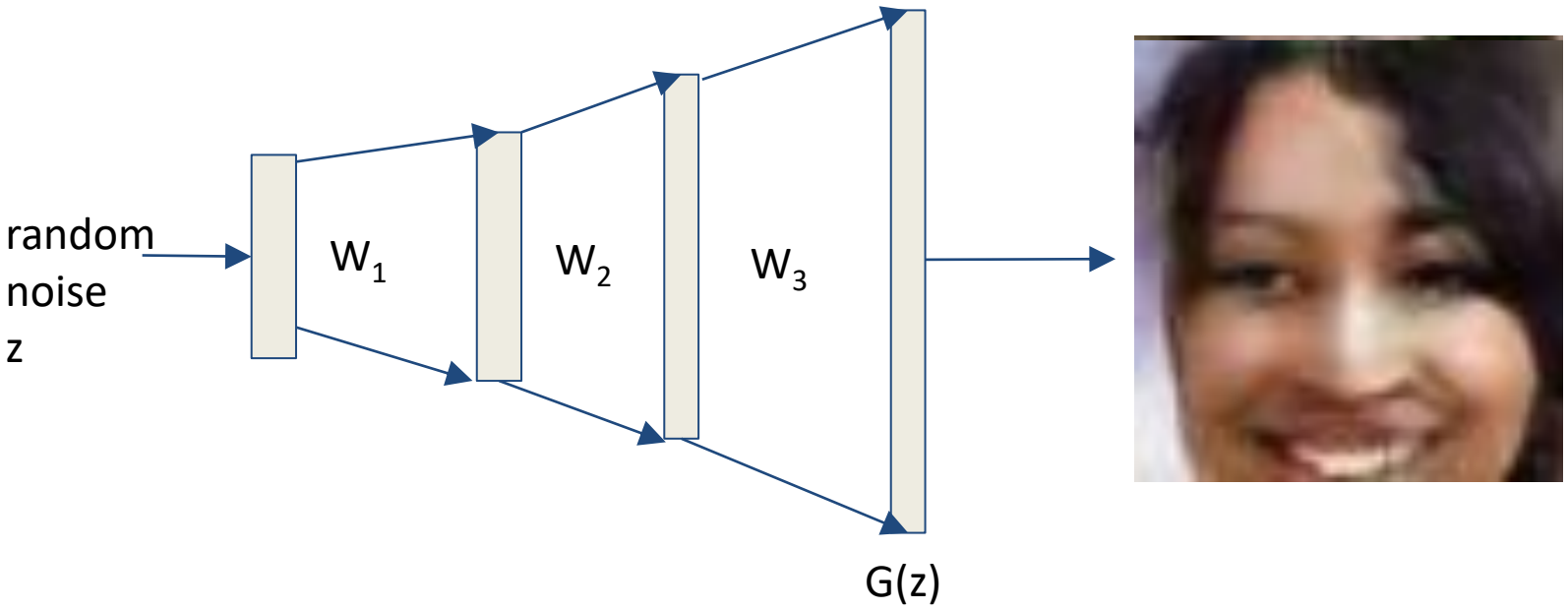
Idea 3: Use sparsity in a basis.

**Idea 4: Use deep generative models:**
**Transform a multivariate gaussian using a neural network.**

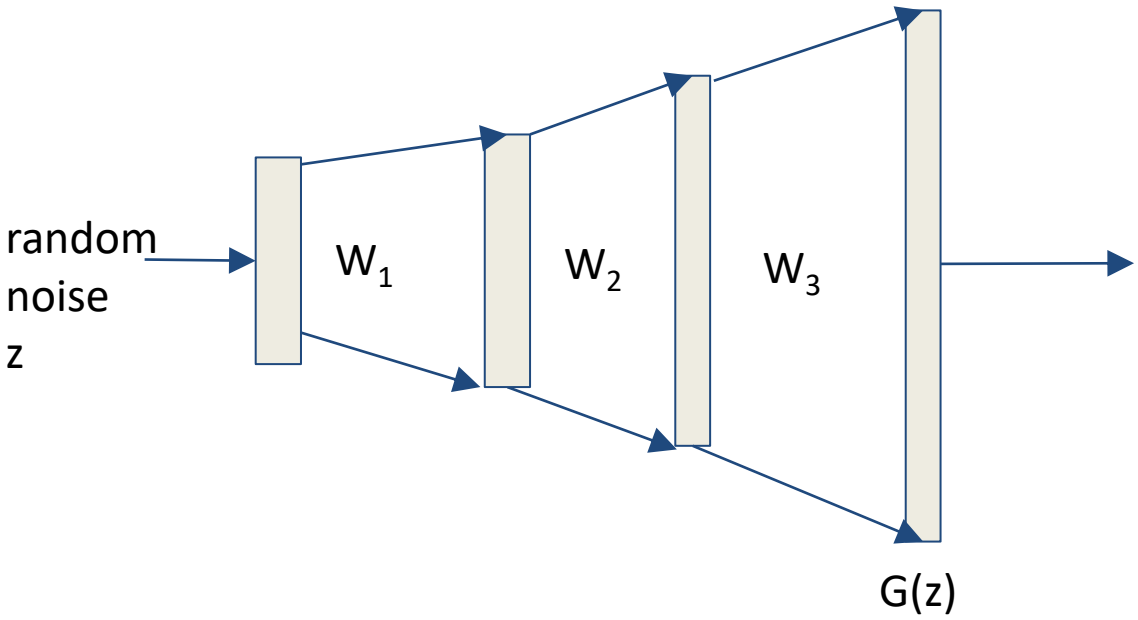
# How training a GAN looks like



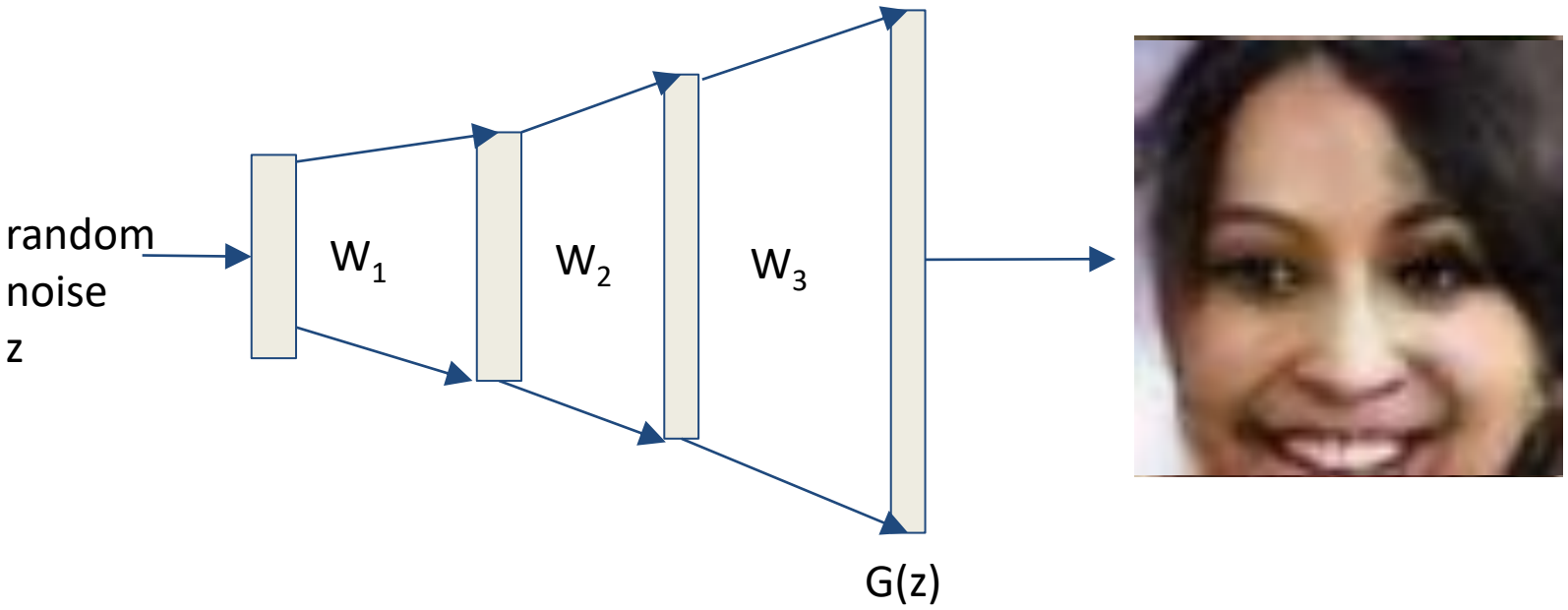
# How training a GAN looks like



# How training a GAN looks like

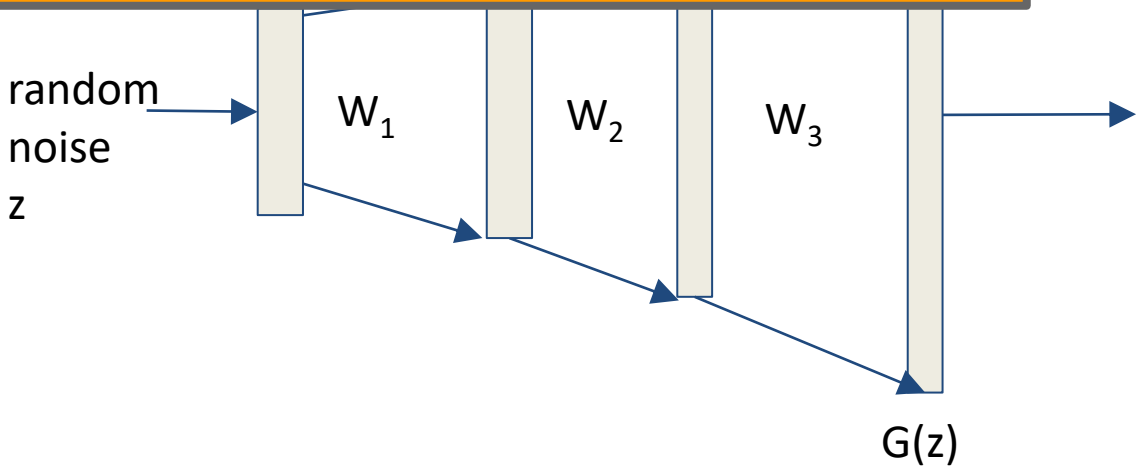



# How training a GAN looks like

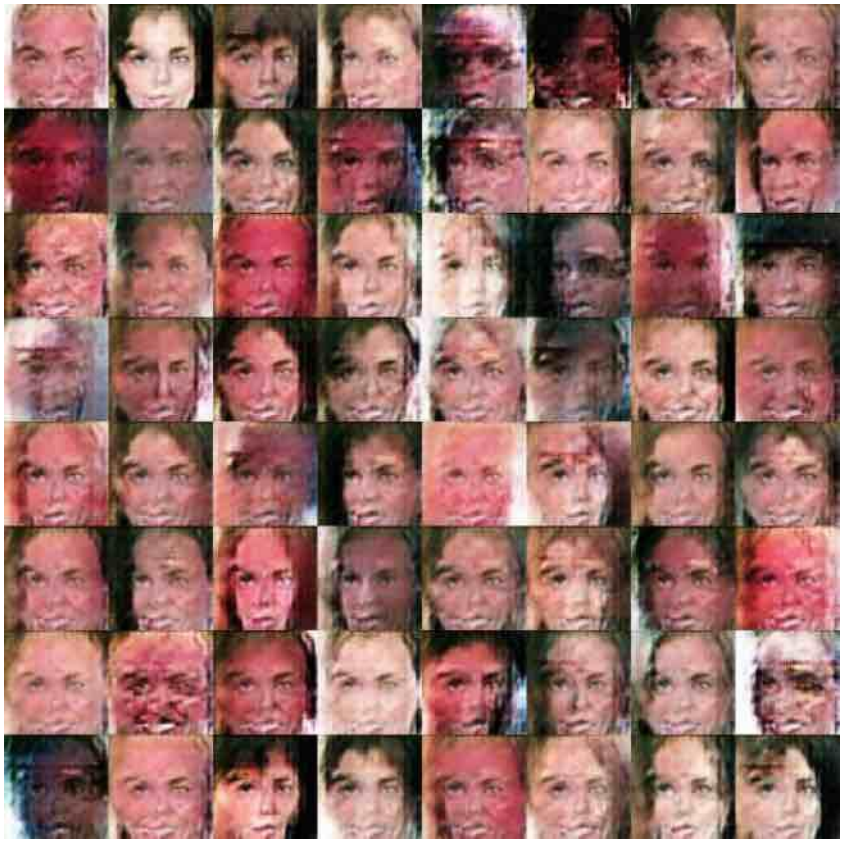


# N looks like

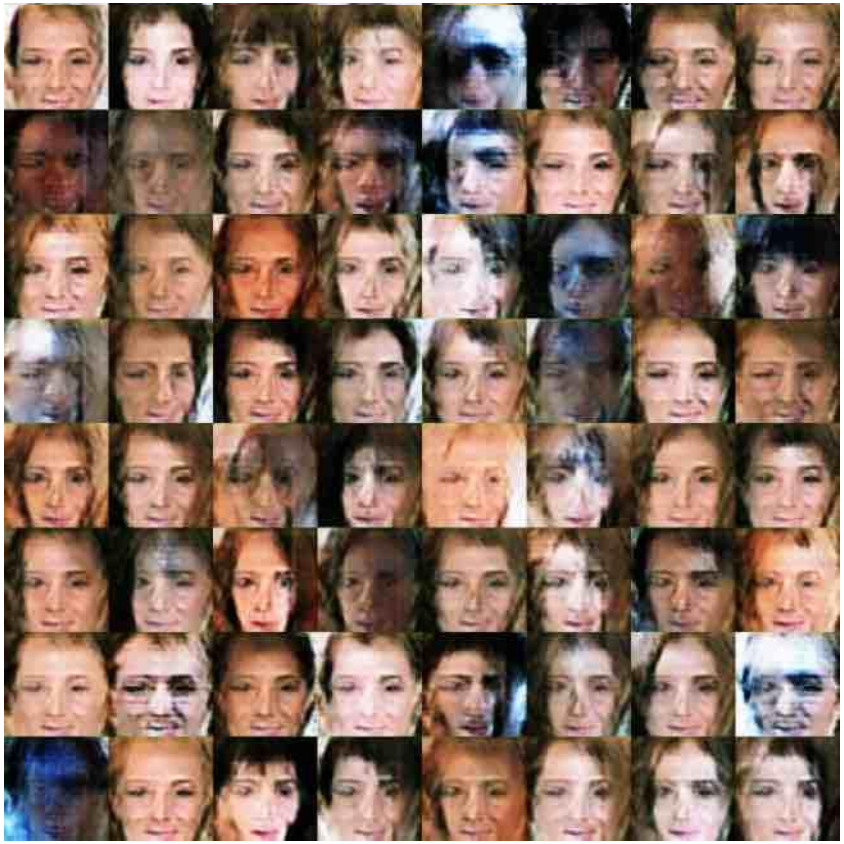
Any Resemblance
to Actual Persons,
Living or Dead,
is Purely Coincidental



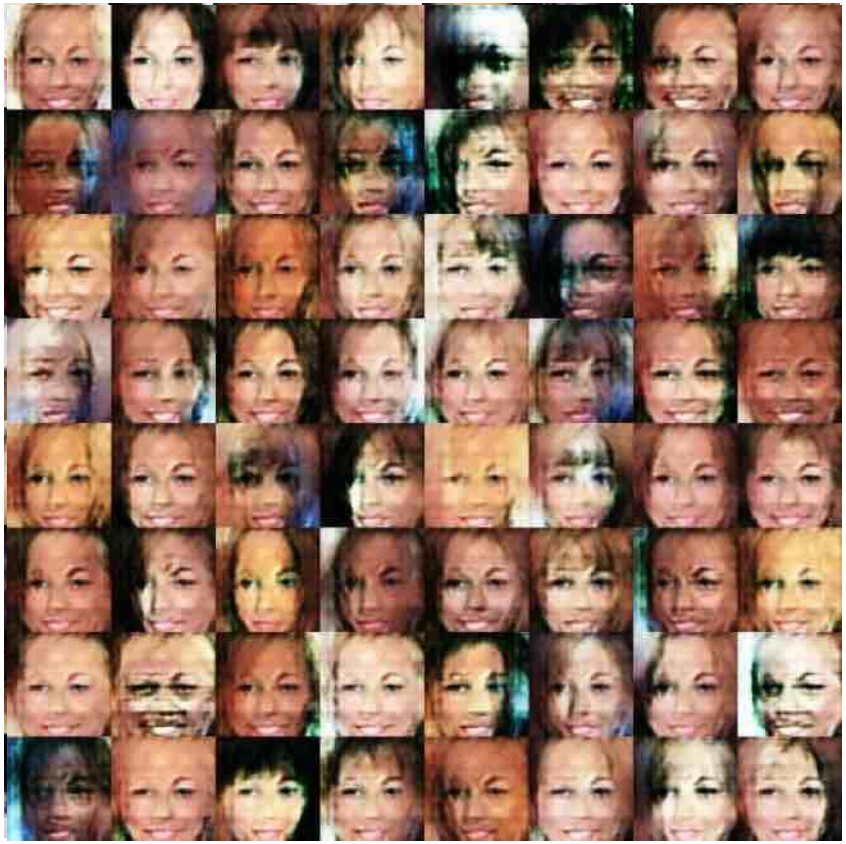

# Adversarial Training



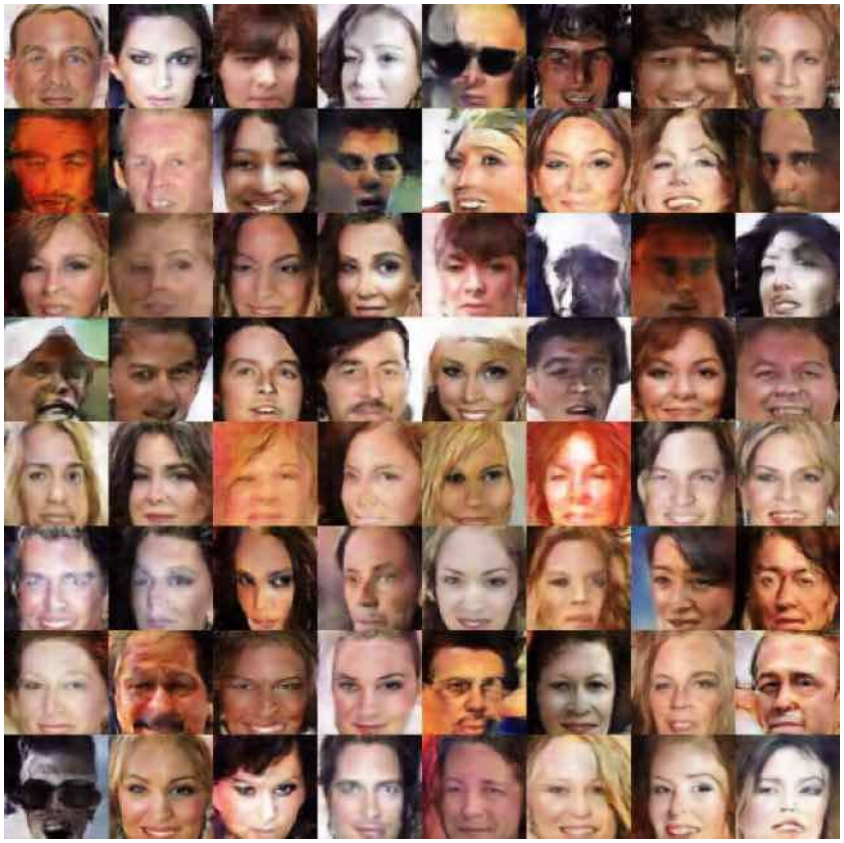
# Adversarial Training



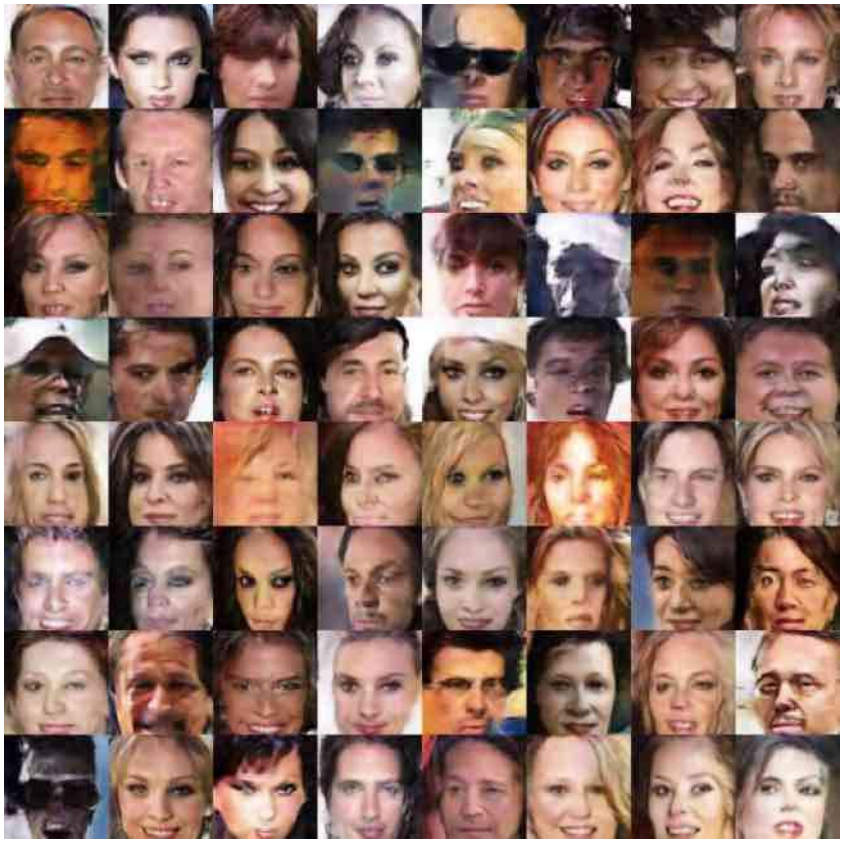
# Adversarial Training



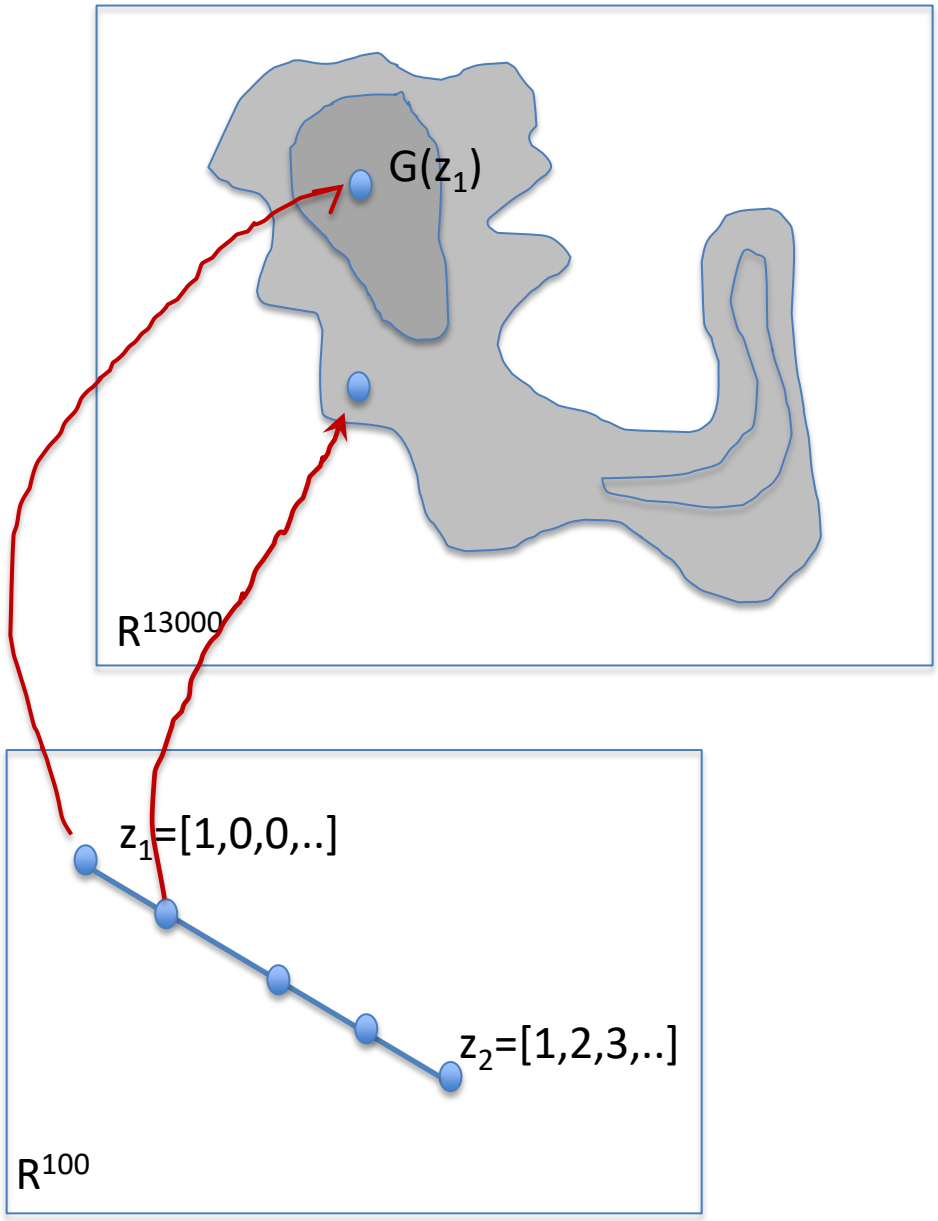
# Adversarial Training



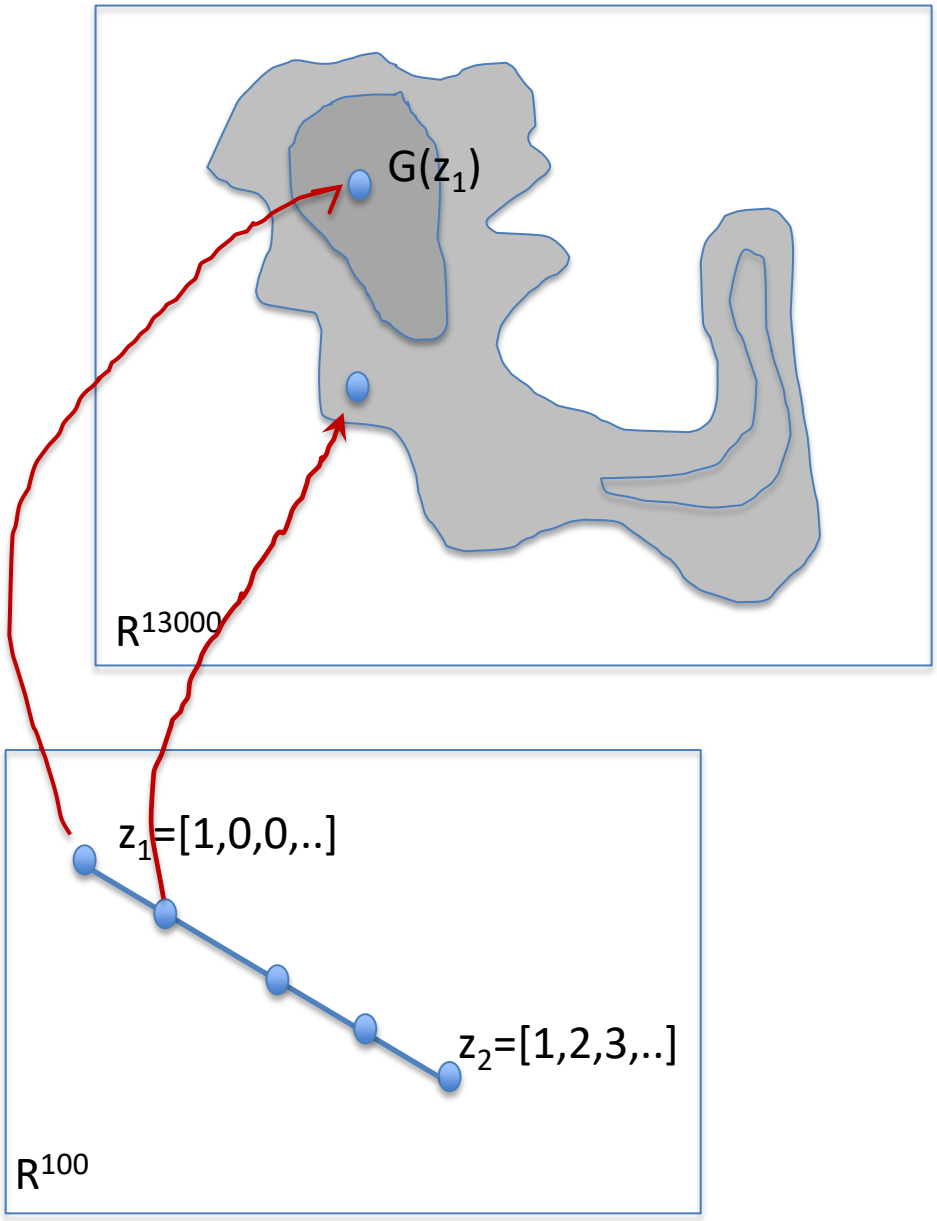
# Adversarial Training



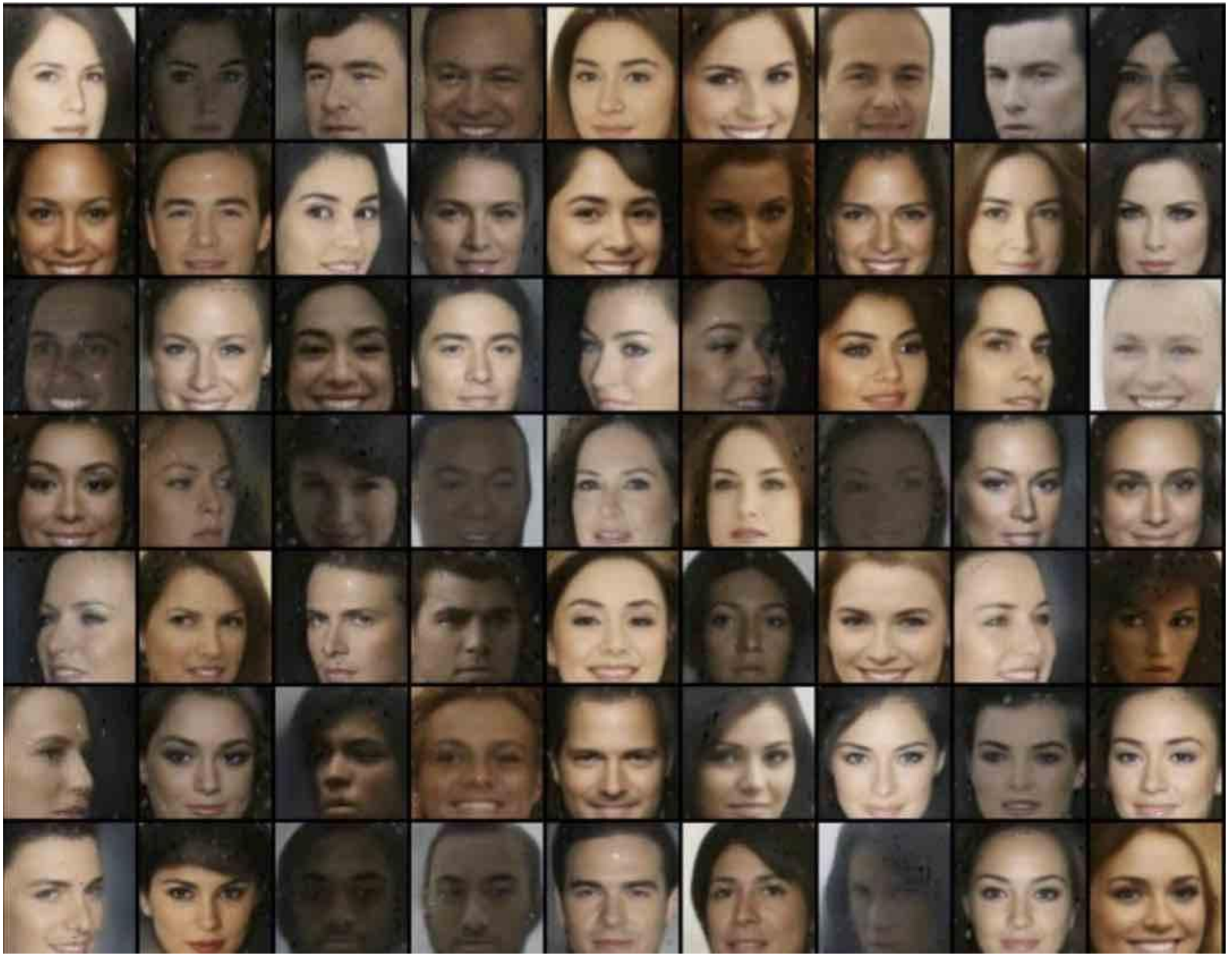
# You can travel in z space too



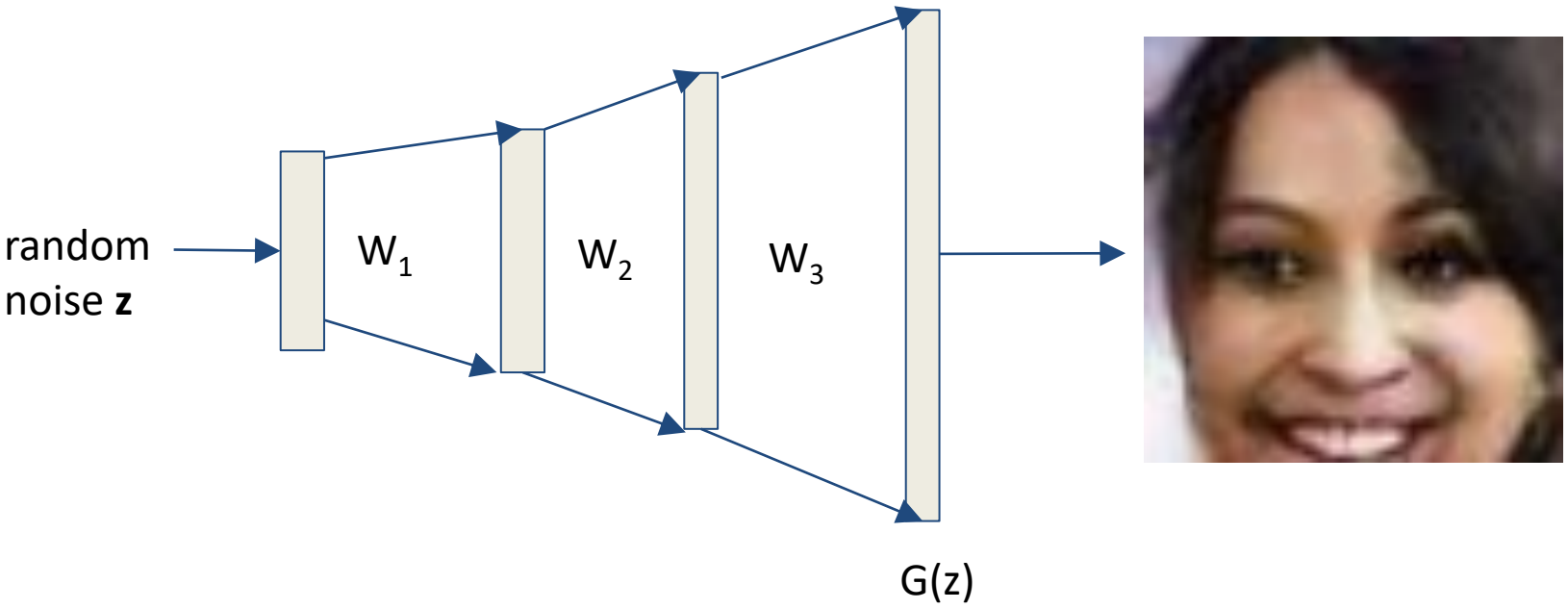
# You can travel in z space too




# Empirically, GANs produce amazing images



# Deep generative models



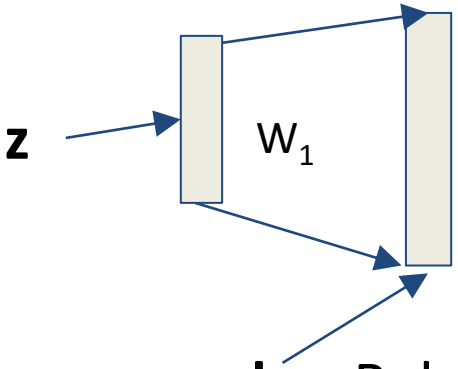



Matrices $W_1$, $W_2$, .. $W_D$ trained from data
(Two standard ways: GANs and VAEs)

# Outline

- 1. Supervised learning works well*
- 2. Except when it is not (adversarial examples, distribution shifts)
- 3. We need unsupervised models to detect/model this. Deep Generative models.
- 4. Some results and open problems for generative models.

# Warmup: **One layer** generative models




Forward pass: Find $\mathbf{h_1}$ given $W_1$ and **z**

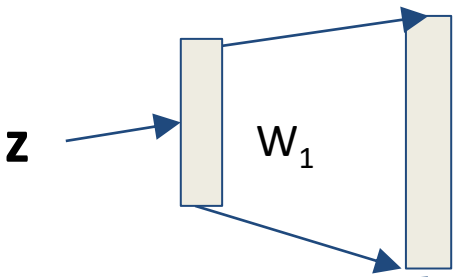
Say $W_1$ =

| | |
|---|---|
| 1 | -1 |
| 2 | -2 |
| 0 | 1 |

**z** =

| |
|---|
| 0 |
| 2 |

# One layer generative models



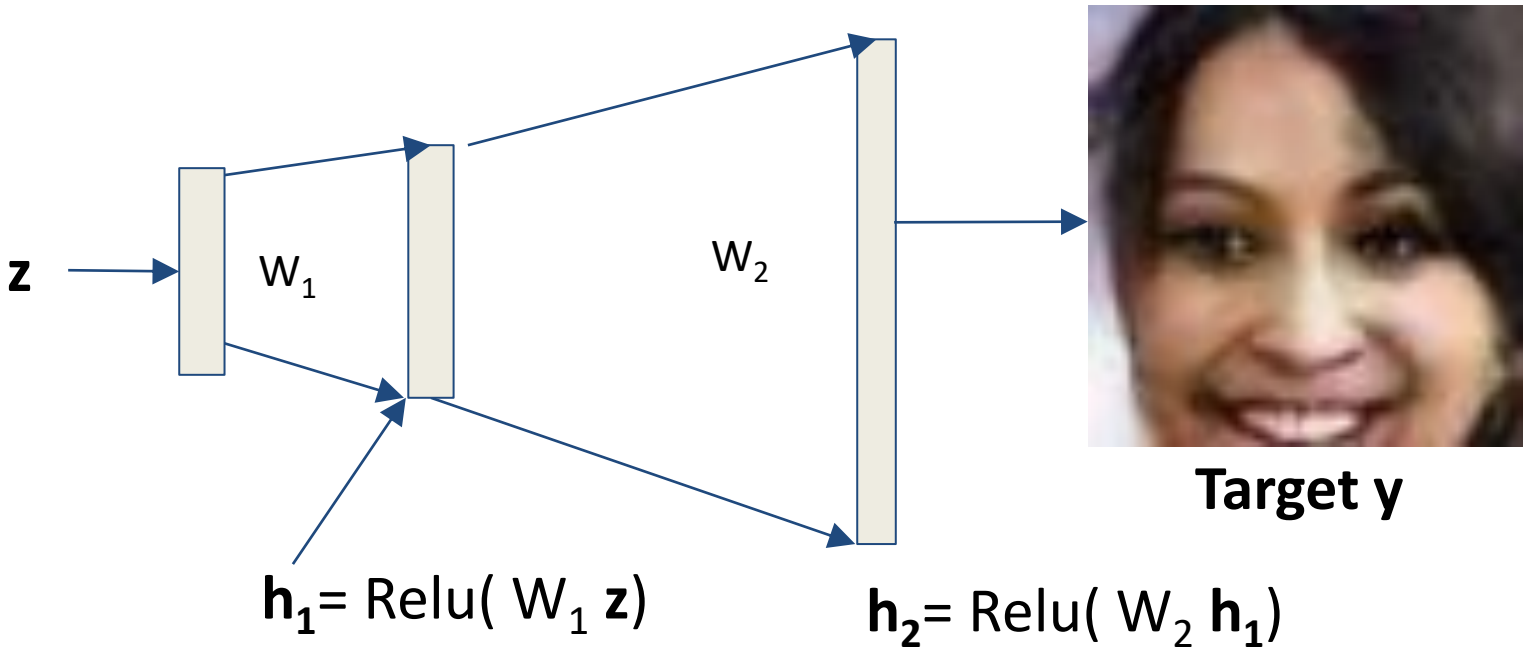

$\mathbf{h_1}$ **=Relu(** $\begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ **)=Relu(** $\begin{bmatrix} -2 \\ -4 \\ 2 \end{bmatrix}$ **)=** $\begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$

$\mathbf{h_1}$= Relu( $W_1$ **z**)

Say $W_1$ **=** $\begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 0 & 1 \end{bmatrix}$

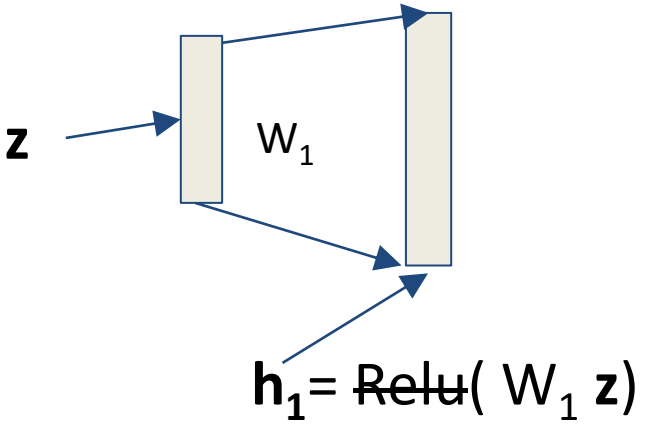
**z =** $\begin{bmatrix} 0 \\ 2 \end{bmatrix}$

# Key problem: Inversion




**Inverting a generative model:**
**Given target observation vector y**
**Find z such that $\mathbf{h_2}$= y**
(or as close as possible in the noisy version)
Weights $W_1$, $W_2$, .. Known.

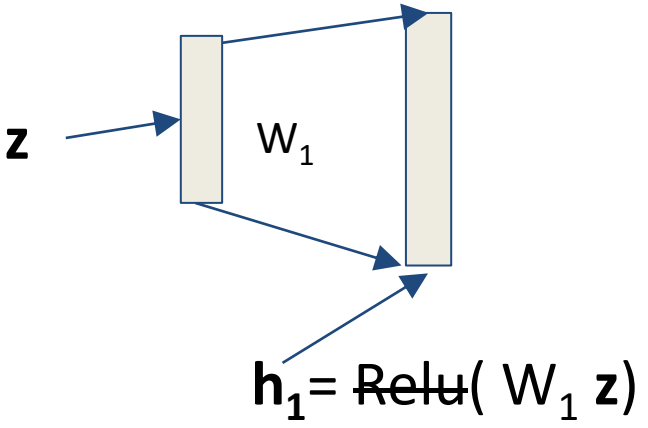
# One layer generative models




Realizable=Noiseless setting, one layer:
Given $\mathbf{h_1}$ = Relu ($W_1$ **z***)
Find z*

If there is no Relu
Given $\mathbf{h_1}$ = $W_1$ **z*** , Find z*

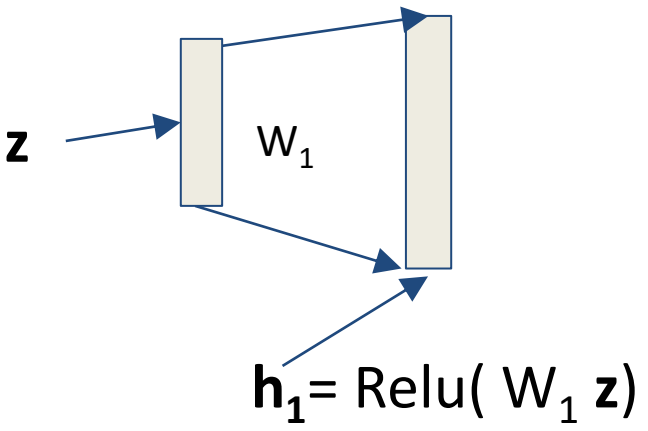
# One layer generative models




Realizable=Noiseless setting, one layer:
Given $\mathbf{h_1}$ = Relu ($W_1$ **z***)
Find z*

If there is no Relu
Given $\mathbf{h_1}$ = $W_1$ **z*** , Find z*
**Ok this is linear equations. Gaussian elimination.**

# One layer generative models



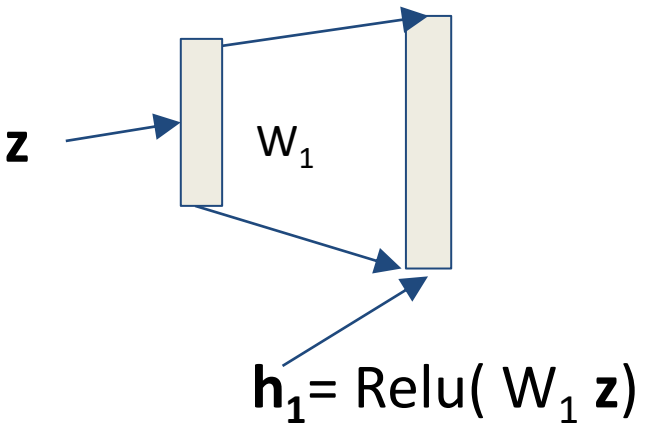

Realizable=Noiseless setting, one layer:
Given $\mathbf{h_1}$ = Relu ($W_1$ **z***)
Find z*

Observation: This is a linear program for one layer.



Inverting Deep Generative models, One layer at a time, Qi et al. NeurIPS 2019.
https://arxiv.org/pdf/1906.07437.pdf

# One layer generative models



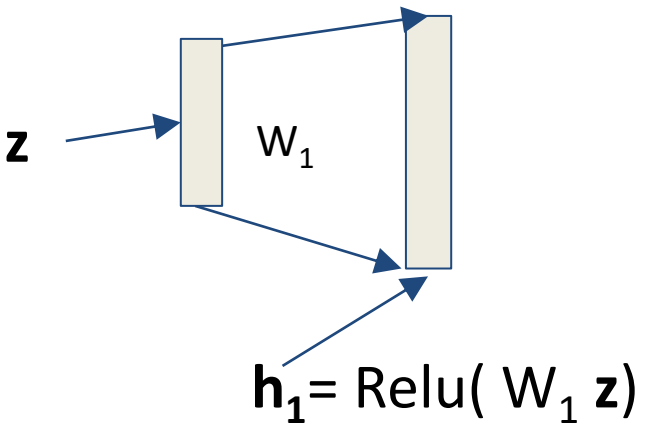

Realizable=Noiseless setting, one layer:
Given $\mathbf{h_1}$ = Relu ($W_1$ **z***)
Find z*

Observation: This is a linear program for one layer.
**Thm:** For two layers or more, inversion is NP-hard.



Inverting Deep Generative models, One layer at a time, Qi et al. NeurIPS 2019.
https://arxiv.org/pdf/1906.07437.pdf

# One layer generative models




Realizable=Noiseless setting, one layer:
Given $\mathbf{h_1} = \text{Relu}(W_1\ \mathbf{z^*})$
Find z*

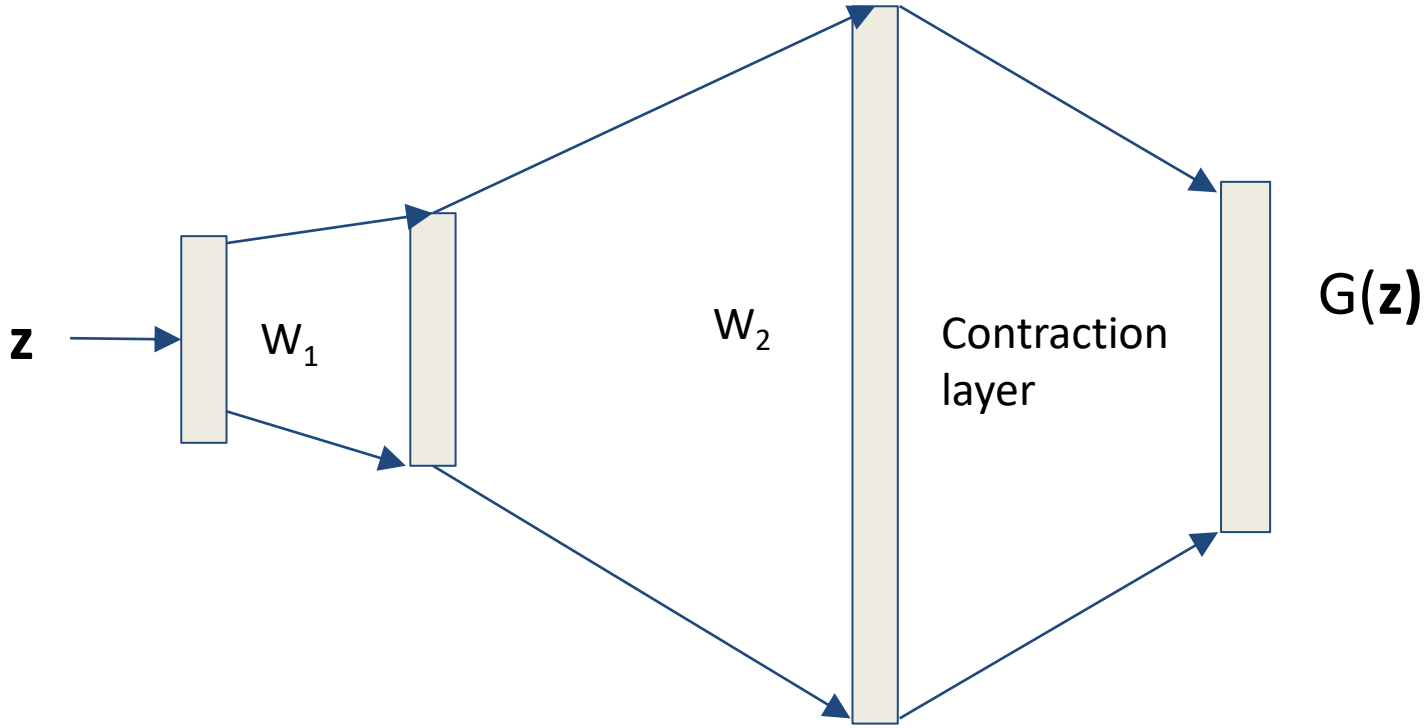
Observation: This is a linear program for one layer.
**Thm:** For two layers or more, inversion is NP-hard.

**Thm:** If all weights are random iid gaussian and each layer is sufficiently expanding, inversion is possible whp in polynomial time.



Inverting Deep Generative models, One layer at a time, Qi et al. NeurIPS 2019.
https://arxiv.org/pdf/1906.07437.pdf

# Deep Generative models




Unfortunately, real deep generative models have some contraction layers (where output dimension < input dimension. So our theory does not apply.

Inversion is done by gradient descent: $\text{Min}_z \; \| G(z) - y \|_2$

In practice, it works for small GANs but not for bigger ones. (Remains open)

# Problem 2: Learning the weights of a deep generative model

- Provably learning a one-layer generative model
- Given samples from **y**= Relu(W **z**) , learn W
- S. Wu, A.G. Dimakis, S. Sanghavi, Learning Distributions Generated by One-Layer ReLU Networks, NeurIPS19.

Without a ReLU, W**z** is gaussian with covariance $WW^T$ So the problem becomes learning a multivariate gaussian.

With ReLUs it becomes a truncated Gaussian. We can learn $WW^T$ using $O( d^2/ \varepsilon^2)$ samples, where d is the ambient dimension to estimate $WW^T$ with error less then $\varepsilon ||W||_F$

# Problem 3: How to evaluate a generator

- Standard method: generate samples, look at them.

G. Daras et al. Your Local GAN: Designing Two Dimensional Local Attention Mechanisms for Generative Models. https://arxiv.org/abs/1911.12287

# Problem 3: How to evaluate a generator

- Standard method: generate samples, look at them.
- Problematic: could be memorizing some subset of images. Cannot be used for categorical data
- Standard methods: FID score, Inception score, try to quantify how natural the generated images are.
- Empirical, heuristic and problematic method.

# A method of evaluating generators

- Given a natural image show me the closest reconstruction.
- This is the most important performance metric for inverse problems. Check if the generator spans the whole real space.

G. Daras et al. Your Local GAN: Designing Two Dimensional Local Attention Mechanisms for Generative Models. https://arxiv.org/abs/1911.12287

# Conclusions

- Supervised learning works if the data distributions do not change

- We need ways to model data distributions. Deep generative models are currently the state of the art empirically.

- We need provable algorithms for:
- 1. Inverting generative models
- 2. Fitting generative models to data.
- 3. Evaluating if generators fit a real data distribution
- Other problems: Conditional sampling, Marginalization, testing distribution changes. …

- Papers, code and pre-trained models:
- http://users.ece.utexas.edu/~dimakis
- https://github.com/AshishBora/csgm
- Twitter: @AlexGDimakis
- Data.ece.utexas.edu (**UT Minds group**)

# Modeling high-dimensional distributions

- Historically there have been two big families of models for high-dimensional distributions. **Model what is natural.**
- Use two different key properties.

- 1.**Sparsity** in a basis (DCT, Wavelet, etc)
- Used in source coding. Gave us mp3, jpeg, mpeg, etc..

- 2. **Conditional independence**: **Graphical models** (Markov Chains, Bayes nets, Factor graphs, etc)
- Used in channel coding. Gave us LDPCs, fountain codes,..

- 3. **Deep Generative models**. Pass simple randomness through a learnable differentiable function (matrix multiplications+relus).
- All training and inference tasks use gradient descent
- All are poorly understood.

fin