



BST 261: Data Science II

Lecture 15

**Advanced Topics:
Generative Deep Learning**

**Heather Mattie
Harvard T.H. Chan School of Public Health
Spring 2 2019**





Neural Style Transfer

Neural Style Transfer

- ◎ Another major development in deep-learning-driven image modification
- ◎ Introduced by [Leon Gatys et al.](#) in 2015
- ◎ Variations have been introduced, some even as smartphone apps
- ◎ Neural style transfer consists of **applying the style** of a reference image to a target image **while conserving the content** of the target image

Content target



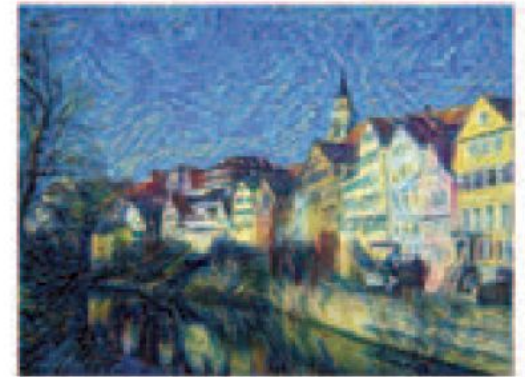
+

Style reference



=

Combination image



Neural Style Transfer

- ◎ Style
 - textures, colors, and visual patterns in the image, at various spatial scales
- ◎ Content
 - higher-level macrostructure of the image
- ◎ Like other neural nets, we need to define and minimize a loss function
 - We want to conserve the content of the original image, while adopting the style of the reference image
 - If we can mathematically define content and style, our loss function would be:

$$\text{Loss} = \text{distance}(\text{style}(\text{ref_image}) - \text{style}(\text{generated_image})) + \text{distance}(\text{content}(\text{original_image}) - \text{content}(\text{generated_image}))$$

Neural Style Transfer

- ◎ Deep CNNs offer a way to define this loss function mathematically
- ◎ Recall:
 - Activations from earlier layers in a network contain local information about the image
 - Activations from higher layers contain increasingly global, abstract information
- ◎ Content loss
 - The content of an image is more global and abstract and should be captured by the representations of later layers
 - Loss is the L2 norm between the activations of an upper layer in a pre-trained CNN, computed over the target image, and the activations of the same layer computed over the generated image
 - Ensures the generated image will look similar to the original target image

Neural Style Transfer

◎ Style loss

- Uses multiple layers of the CNN
- Try to capture the appearance of the style reference image at all spatial scales extracted by the CNN, not just a single scale
- Use the Gram matrix of a layer's activations: the inner product of the feature maps of a given layer
- This inner product can be understood as representing a map of the correlations between the layer's features
- These feature correlations capture the statistics of the patterns of a particular spatial scale, which empirically correspond to the appearance of the textures found at this scale
- Aims to preserve similar internal correlations within the activations of different layers, across the style-reference image and the generated image
- Guarantees that the **textures found at different spatial scales** look similar across the style-reference image and the generated image

Neural Style Transfer

◎ Summary

- Preserve content by maintaining similar high-level layer activations between the target content image and the generated image. The CNN should “see” both the target image and the generated image as containing the same things
- Preserve style by maintaining similar correlations within activations for both low-level layers and high-level layers. Feature correlations capture textures: the generated image and the style-reference image should share the same textures at different spatial scales

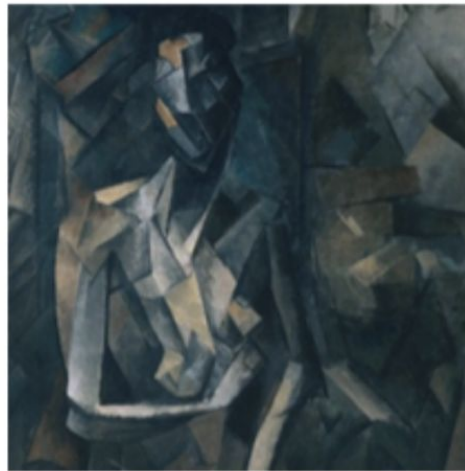
Neural Style Transfer

[Awesome blog post](#)

Content C



Style S



Generated Image G



Picasso Dancer

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$

Neural Style Transfer

Image Style Transfer Using CNNs

“ A Neural Algorithm of Artistic Style that can separate and recombine the image content and style of natural images. The algorithm allows us to produce new images of high perceptual quality that combine the content of an arbitrary photograph with the appearance of numerous well-known artworks”



The background of the slide is a light blue-grey color with a repeating pattern of a network graph. The graph consists of numerous small circular nodes, some of which are solid grey and others are hollow with a grey outline. These nodes are interconnected by a web of thin, light grey lines, creating a complex, interconnected mesh that covers the entire background.

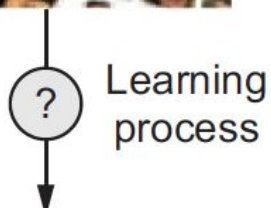
Image Generation

Generating Images

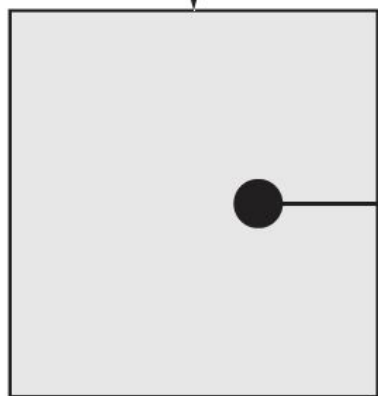
- ◎ We'll discuss two different ways of generating images
 - Variational autoencoders (VAEs)
 - Generative adversarial networks (GANs)
- ◎ These methods can also be used to generate sound, music or text, but we'll focus on images
- ◎ **Main idea:** sample from a **latent space** of images to create entirely new images
 - Latent space: a low-dimensional representation (vector space) where any point can be mapped to a realistic-looking image
 - The module that takes in a point from the latent space and generates an image is called a **generator** (in the case of GANs) or a **decoder** (in the case of VAEs)
 - Generates images never explicitly seen before



Training data



Learning
process



Latent space
of images
(a vector space)



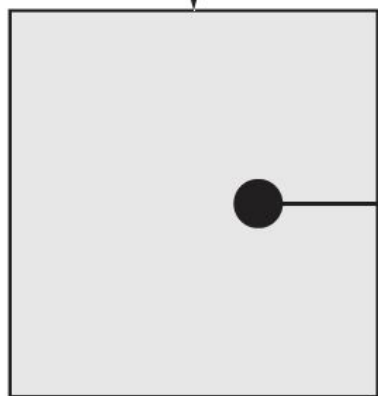
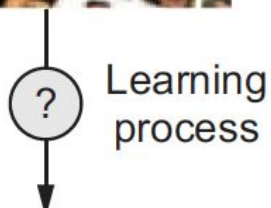
Vector from the
latent space



Artificial
image



Training data



Latent space
of images
(a vector space)



Vector from the
latent space



GANs
terminology

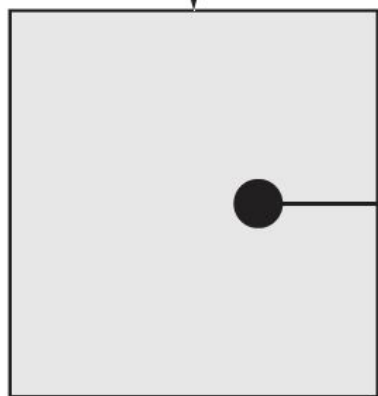
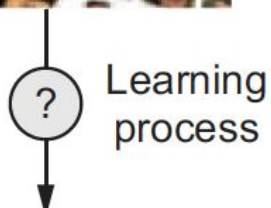
Generator / Decoder



Artificial
image



Training data

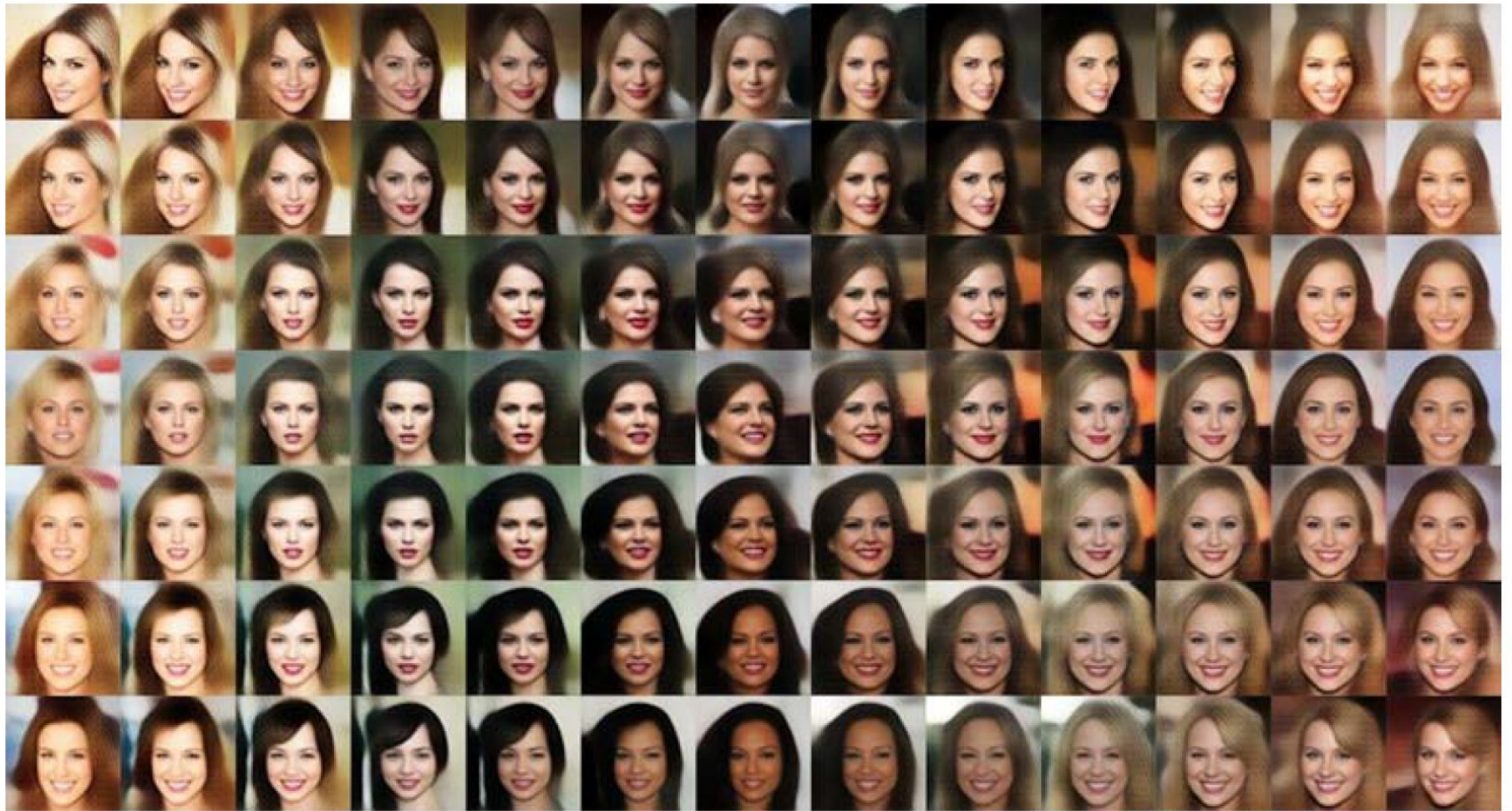


Latent space of images
(a vector space)

Vector from the
latent space



Artificial
image



Images generated from a VAE

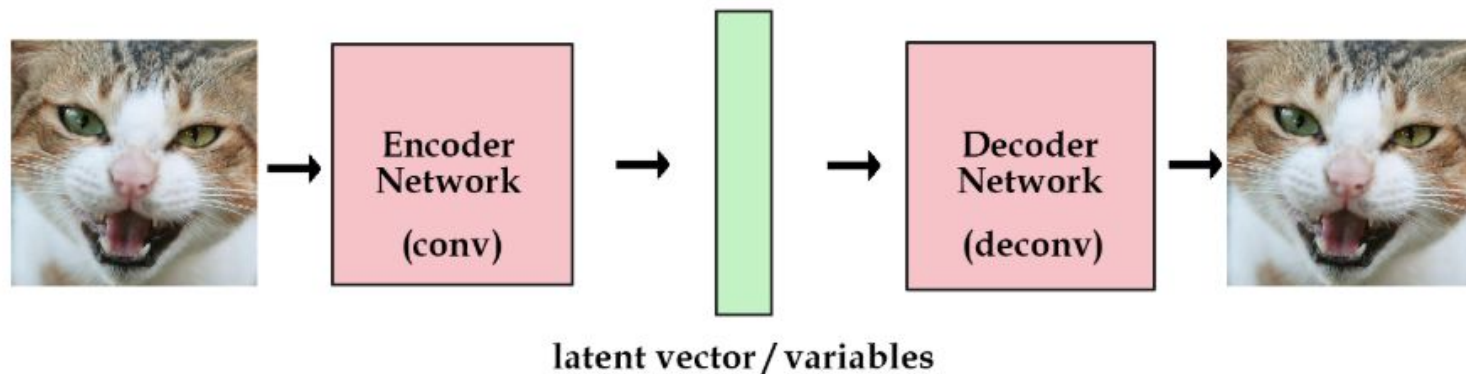
Concept vectors

- ◎ We want to create a latent vector space
 - We can think of this as embedding - just like what we did with text and RNNs
 - Certain directions in the space may encode interesting axes of variation in the original data
 - Example: vectors that represent a smile or vectors that represent sunglasses
- ◎ Once we create these vectors we can edit images by projecting them into the latent space, moving their representation in a meaningful way, and then decoding them back to image space



Variational Autoencoders (VAEs)

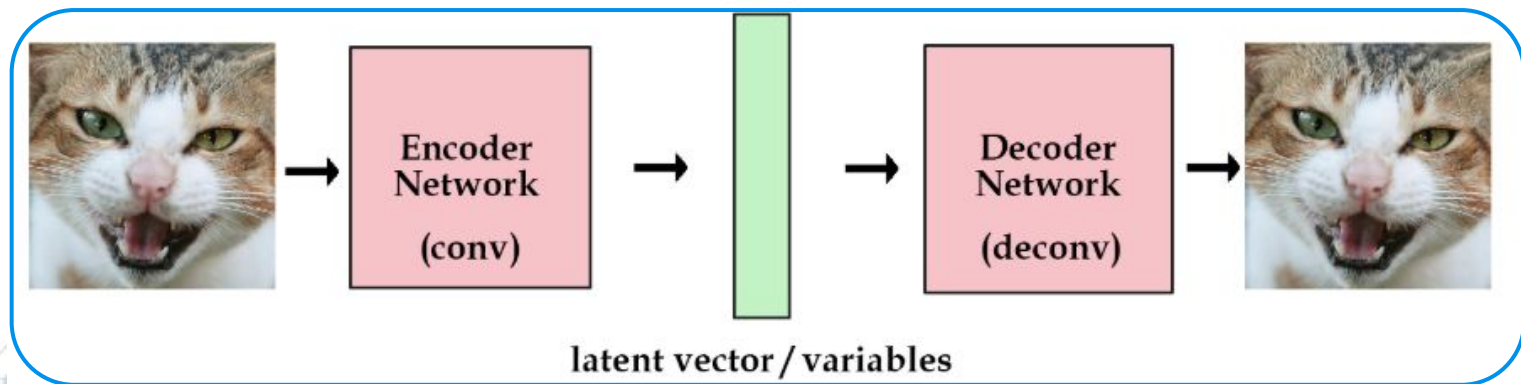
- Simultaneously discovered by [Kingma and Welling](#) in December 2013 and [Rezende, Mohamed, and Wierstra](#) in January 2014
- A generative model especially appropriate for the task of image editing using concept vectors
- Modern version of **autoencoders**
 - Autoencoders are networks that encode an input to a low-dimensional latent space and then decode it back
 - The encoder learns how to reconstruct the original inputs
 - Learn to compress the input data into fewer bits of information



Variational Autoencoders (VAEs)

- Simultaneously discovered by [Kingma and Welling](#) in December 2013 and [Rezende, Mohamed, and Wierstra](#) in January 2014
- A generative model especially appropriate for the task of image editing using concept vectors
- Modern version of **autoencoders**
 - Autoencoders are networks that encode an input to a low-dimensional latent space and then decode it back
 - The encoder learns how to reconstruct the original inputs
 - Learn to compress the input data into fewer bits of information

Autoencoder

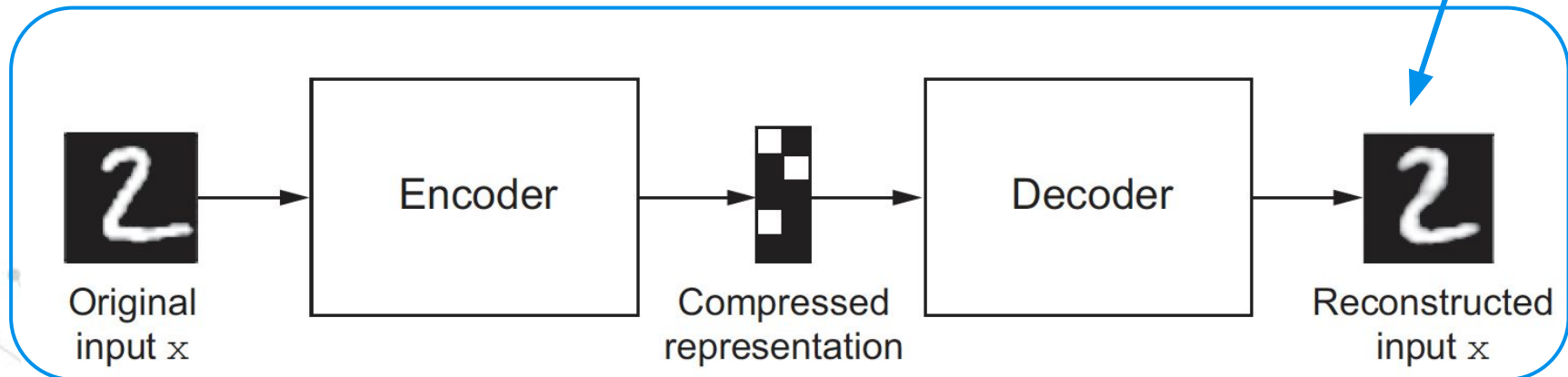


Variational Autoencoders (VAEs)

- Simultaneously discovered by [Kingma and Welling](#) in December 2013 and [Rezende, Mohamed, and Wierstra](#) in January 2014
- A generative model especially appropriate for the task of image editing using concept vectors
- Modern version of **autoencoders**
 - Autoencoders are networks that encode an input to a low-dimensional latent space and then decode it back
 - The encoder learns how to reconstruct the original inputs
 - Learn to compress the input data into fewer bits of information

VAE

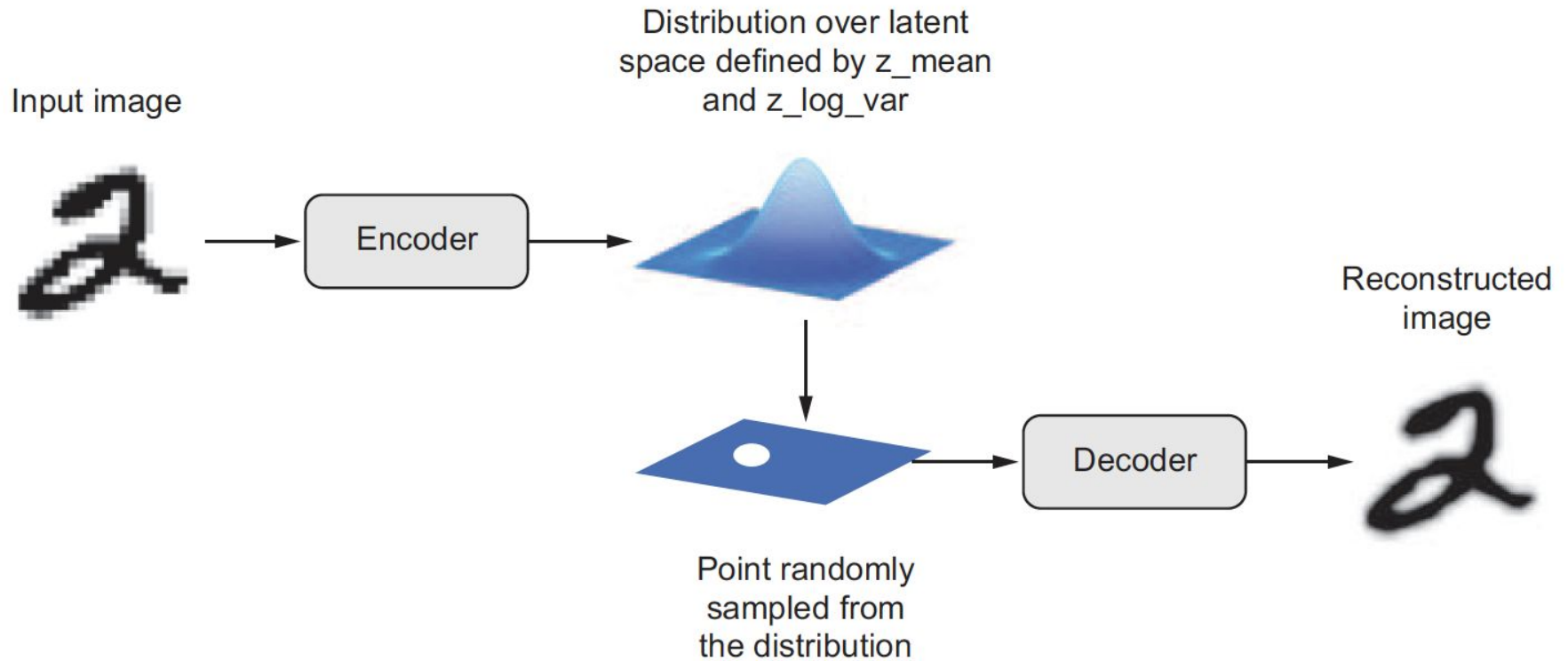
Produces output similar to the input, but not exactly the same



VAEs

- ⊙ Autoencoders aren't great at creating nicely structured latent spaces and they also don't generate anything new
- ⊙ VAEs augment autoencoders to learn highly structured latent spaces and are able to generate new images
- ⊙ Rather than compressing the image into a fixed code in the latent space, VAEs turn the image into the **parameters of a statistical distribution** (a mean and variance)
 - Assume the input image has been generated by a statistical process and that the **randomness** of this process should be taken into account when encoding and decoding
 - A VAE uses the mean and variance parameters to randomly sample one element from the distribution and decodes that element back to the original input
 - ⊙ This process improves robustness and forces the latent space to encode meaningful representations everywhere: every point in the latent space is decoded to a valid output

VAEs



VAEs

Algorithm steps:

- An encoder module turns the input samples `input_img` into two parameters in a latent space of representations, **`z_mean`** and **`z_log_variance`**
- Randomly sample a point `z` from the latent normal distribution that's assumed to generate the input image, via

$$\mathbf{z} = \mathbf{z_mean} + \exp(\mathbf{z_log_variance}) * \epsilon$$

where `epsilon` is a random tensor of small values

- A decoder module maps this point in the latent space back to the original input image
- ## Having `epsilon` be random ensures every point that's close to the latent location where you encoded the input image can be decoded to something similar to the input image - but not exactly the same
- Also forces every direction in the latent space to encode a meaningful axis of variation of the data, making the latent space very structured and highly suitable to manipulation via concept vectors

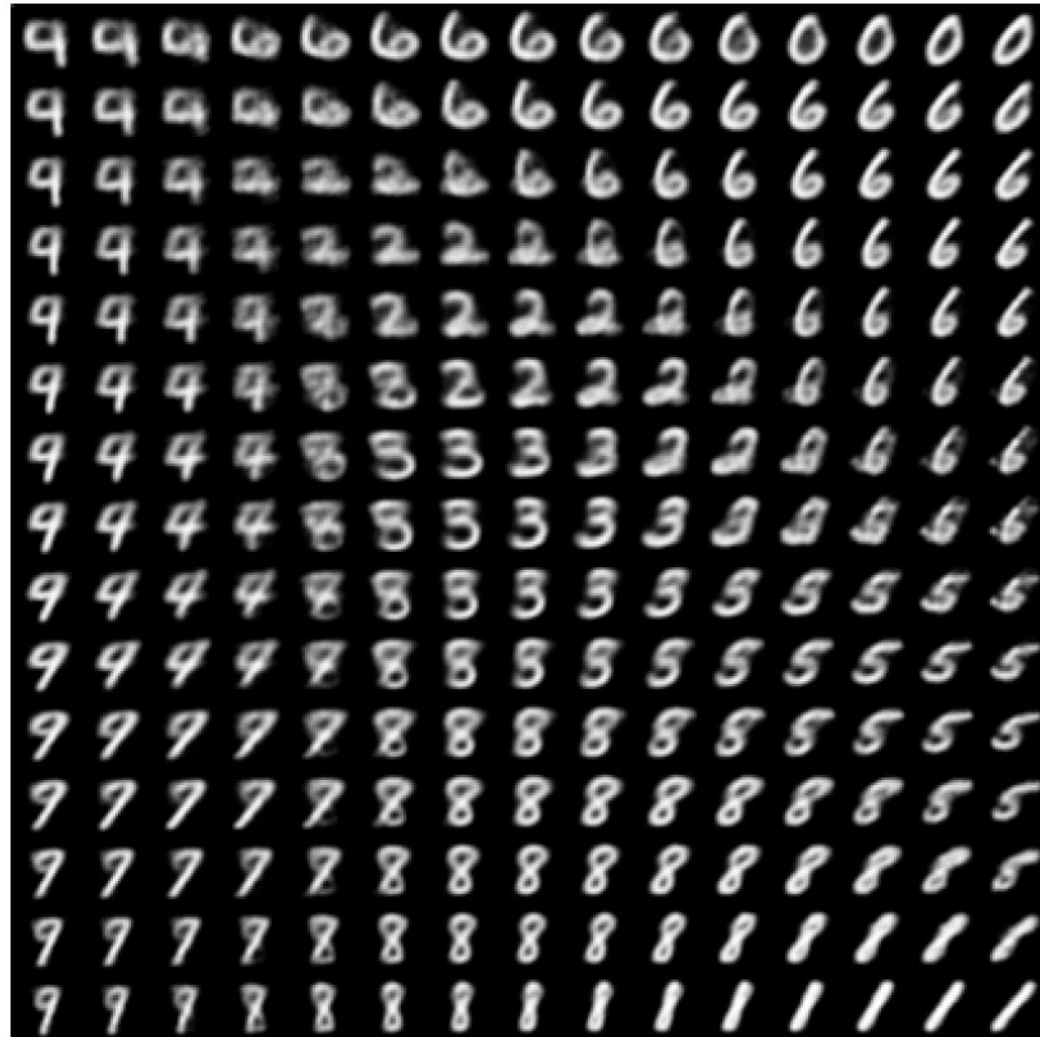
VAE Training

- Parameters are trained using two loss functions
 - Reconstruction (generative) loss:** measures how accurately the network reconstructed the images; forces the decoded samples to match the initial inputs
 - Regularization (latent) loss:** measures how closely the latent variables match a unit Gaussian distribution; helps learn well-formed latent spaces and reduce overfitting to the training data
- Because this is a different type of loss, Keras allows you to have a custom loss function and add it as a layer to your network using the **add_loss** layer

VAE on MNIST Data

Digits decoded from
the latent space

Note how one digit
morphs into another
and that directions
in this space have
specific meaning
(one direction for
“four-ness”, one
direction for
“one-ness”, etc.



VAEs for Medicine

Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders

Gregory P. Way and

Genomics and Computational Biology Graduate Program, University of Pennsylvania,
Philadelphia, PA 19104, USA

Casey S. Greene*

Department of Systems Pharmacology and Translational Therapeutics, University of
Pennsylvania, Philadelphia, PA 19104, USA

Abstract

The Cancer Genome Atlas (TCGA) has profiled over 10,000 tumors across 33 different cancer-types for many genomic features, including gene expression levels. Gene expression measurements capture substantial information about the state of each tumor. Certain classes of deep neural network models are capable of learning a meaningful latent space. Such a latent space could be used to explore and generate hypothetical gene expression profiles under various types of molecular and genetic perturbation. For example, one might wish to use such a model to predict a tumor's response to specific therapies or to characterize complex gene expression activations existing in differential proportions in different tumors. Variational autoencoders (VAEs) are a deep

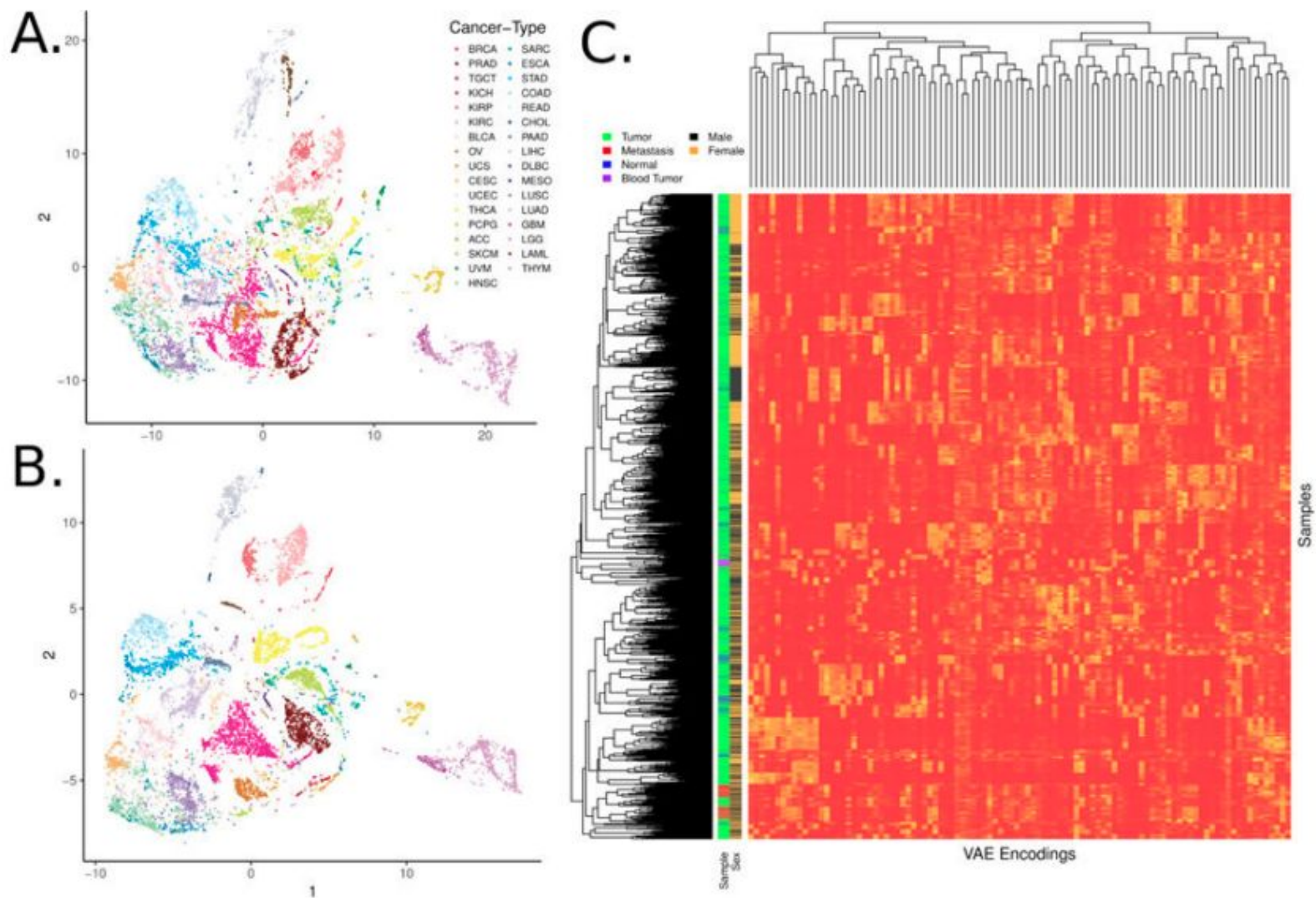




Fig. 2. Samples encoded by a variational autoencoder retain biological signals
(A) t-distributed stochastic neighbor embedding (t-SNE) of TCGA pan-cancer tumors with Tybalt encoded features. **(B)** t-SNE of 0-1 normalized gene expression features. Tybalt retains similar signals as compared to uncompressed gene expression data. **(C)** Full Tybalt encoding features by TCGA pan-cancer sample heatmap. Given on the y axis are the patients sex and type of sample.

 setup.py	add util import to init (#133)	a year ago
 tsne_tybalt_features.ipynb	Adding ADAGE model to tsne visualization (#114)	a year ago
 tybalt_twohidden.ipynb	Add Two Hidden Layer Model (#81)	2 years ago
 tybalt_vae.ipynb	Update conda environments (#108)	a year ago

README.md

Tybalt

A Variational Autoencoder trained on Pan-Cancer Gene Expression

Gregory Way and Casey Greene 2017

DOI [10.5281/zenodo.1047069](https://doi.org/10.5281/zenodo.1047069)

The repository stores scripts to train, evaluate, and extract knowledge from a variational autoencoder (VAE) trained on 33 different cancer-types from The Cancer Genome Atlas (TCGA).

The specific VAE model is named *Tybalt* after an instigative, cat-like character in Shakespeare's "Romeo and Juliet". Just as the character Tybalt sets off the series of events in the play, the model Tybalt begins the foray of VAE manifold learning in transcriptomics. [Also, deep unsupervised learning likes cats.](#)

We discuss the training and evaluation of Tybalt in our PSB paper:

[Extracting a Biologically Relevant Latent Space from Cancer Transcriptomes with Variational Autoencoders.](#)

<https://github.com/greenelab/tybalt>

VAEs for Medicine

Dr.VAE: Drug Response Variational Autoencoder

Ladislav Rampasek^{*†‡}

Daniel Hidru^{†‡}

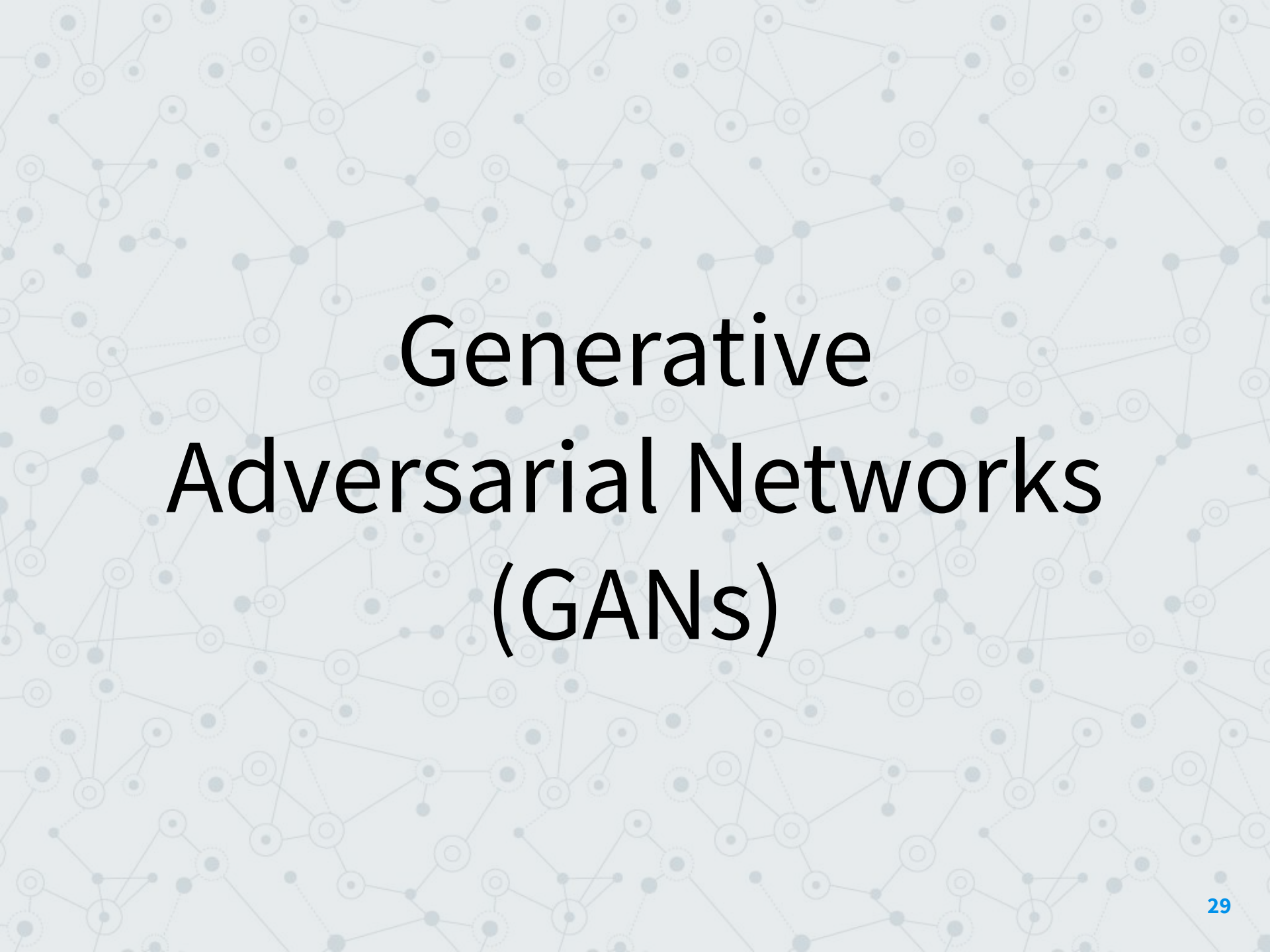
Petr Smirnov[§]

Benjamin Haibe-Kains^{§¶||}

Anna Goldenberg^{*†‡}

Abstract

We present two deep generative models based on Variational Autoencoders to improve the accuracy of drug response prediction. Our models, Perturbation Variational Autoencoder and its semi-supervised extension, Drug Response Variational Autoencoder (Dr.VAE), learn latent representation of the underlying gene states before and after drug application that depend on: (i) drug-induced biological change of each gene and (ii) overall treatment response outcome. Our VAE-based models outperform the current published benchmarks in the field by anywhere from 3 to 11% AUROC and 2 to 30% AUPR. In addition, we found that better reconstruction accuracy does not necessarily lead to improvement in classification accuracy and that jointly trained models perform better than models that minimize reconstruction error independently.

The background of the slide features a complex, light gray network pattern. It consists of numerous small circles, some of which are solid and others are hollow, connected by thin, intersecting lines that form a web-like structure across the entire page.

Generative Adversarial Networks (GANs)

GANs

- ⦿ Introduced by Goodfellow et al in 2014
- ⦿ An alternative to VAEs for learning latent spaces of images
- ⦿ Enable generation of fairly realistic synthetic images by forcing the generated images to be statistically almost indistinguishable from real ones
- ⦿ Made of 2 parts:
 - **Generator network:** takes as input a random vector (a random point in the latent space) and decodes it into a synthetic image - trained to fool the discriminator network
 - **Discriminator network (or adversary):** takes as input an image (real or synthetic) and predicts whether the image came from the training set or was created by the generator network

Intuition Examples

Think of someone trying to create counterfeit money who has a spy inside a bank. They can create the fake money and try to slip it past bank employees. The bank employees will notice the fake money easily in the beginning, but as the spy relays information to the counterfeiter, they will make better fake versions of money that will be more difficult for bank employees to distinguish as fake. Meanwhile, the bank will come up with new ways of detecting the updated counterfeit money.



Intuition Examples

Think of someone trying to create counterfeit money who has a spy inside a bank. They can create the fake money and try to slip it past bank employees. The bank employees will notice the fake money easily in the beginning, but as the spy relays information to the counterfeiter, they will make better fake versions of money that will be more difficult for bank employees to distinguish as fake. Meanwhile, the bank will come up with new ways of detecting the updated counterfeit money.



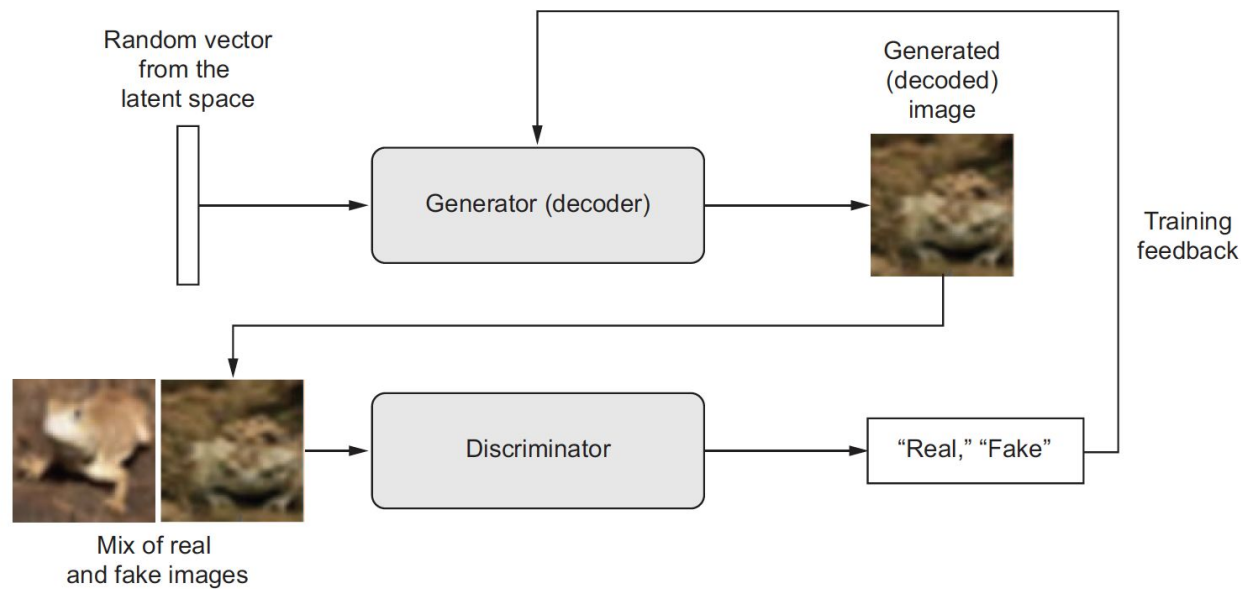
Intuition Examples

Now think of someone trying to forge Picasso paintings. They will be bad in the beginning, but will improve over time with feedback from an expert art dealer on how they detected the fake art from the real paintings. At the same time, new, more advanced techniques will be discovered to spot forged paintings.



GANs

- The generator learns to generate increasingly realistic images as it trains
- At the same time, the discriminator network is constantly adapting to the gradually improving capabilities of the generator
- After training, the generator is able to turn any point in its input space into a believable image
- Unlike VAEs, this latent space has fewer explicit guarantees of meaningful structure



Training

- ◎ The optimization minimum isn't fixed for GANs
 - Every step taken during gradient descent changes the entire landscape - the optimization process is dynamic
 - Need to find an equilibrium, not a minimum
- ◎ Very difficult to train
 - Many parameters to tune
 - Complex model architecture



Training

- ◎ Implementation of a deep convolutional GAN (DCGAN) in Keras
 - The generator and discriminator are deep CNNs
 - A **generator** network maps vectors to images
 - A **discriminator** network maps images to a binary score estimating the probability that the image is real
 - A gan network chains the generator and the discriminator together:

$$\mathbf{gan}(\mathbf{x}) = \mathbf{discriminator}(\mathbf{generator}(\mathbf{x}))$$

Thus this gan network maps latent space vectors to the discriminator's assessment of the realism of these latent vectors as decoded by the generator

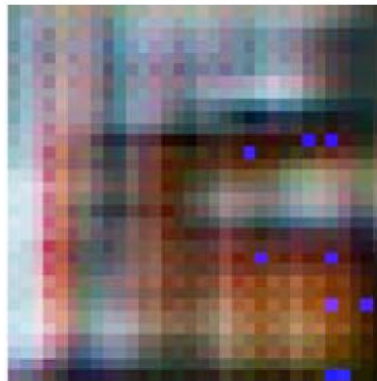
- You train the discriminator using examples of real and fake images along with “real”/“fake” labels, just as you train any regular image-classification model
- To train the generator, you use the gradients of the generator's weights with regard to the loss of the gan model. This means, at every step, you move the weights of the generator in a direction that makes the discriminator more likely to classify as “real” the images decoded by the generator. In other words, you train the generator to fool the discriminator

Tricks for Training

- ◎ There are a few useful tricks to help train a GAN
 - Use tanh as the last activation in the generator, instead of sigmoid, which is more commonly found in other types of models
 - Sample points from the latent space using a normal distribution (Gaussian distribution), not a uniform distribution
 - Stochasticity is good to induce robustness. Because GAN training results in a dynamic equilibrium, GANs are likely to get stuck in all sorts of ways. Introducing randomness during training helps prevent this. You can introduce randomness in two ways: by using dropout in the discriminator and by adding random noise to the labels for the discriminator

Tricks for Training

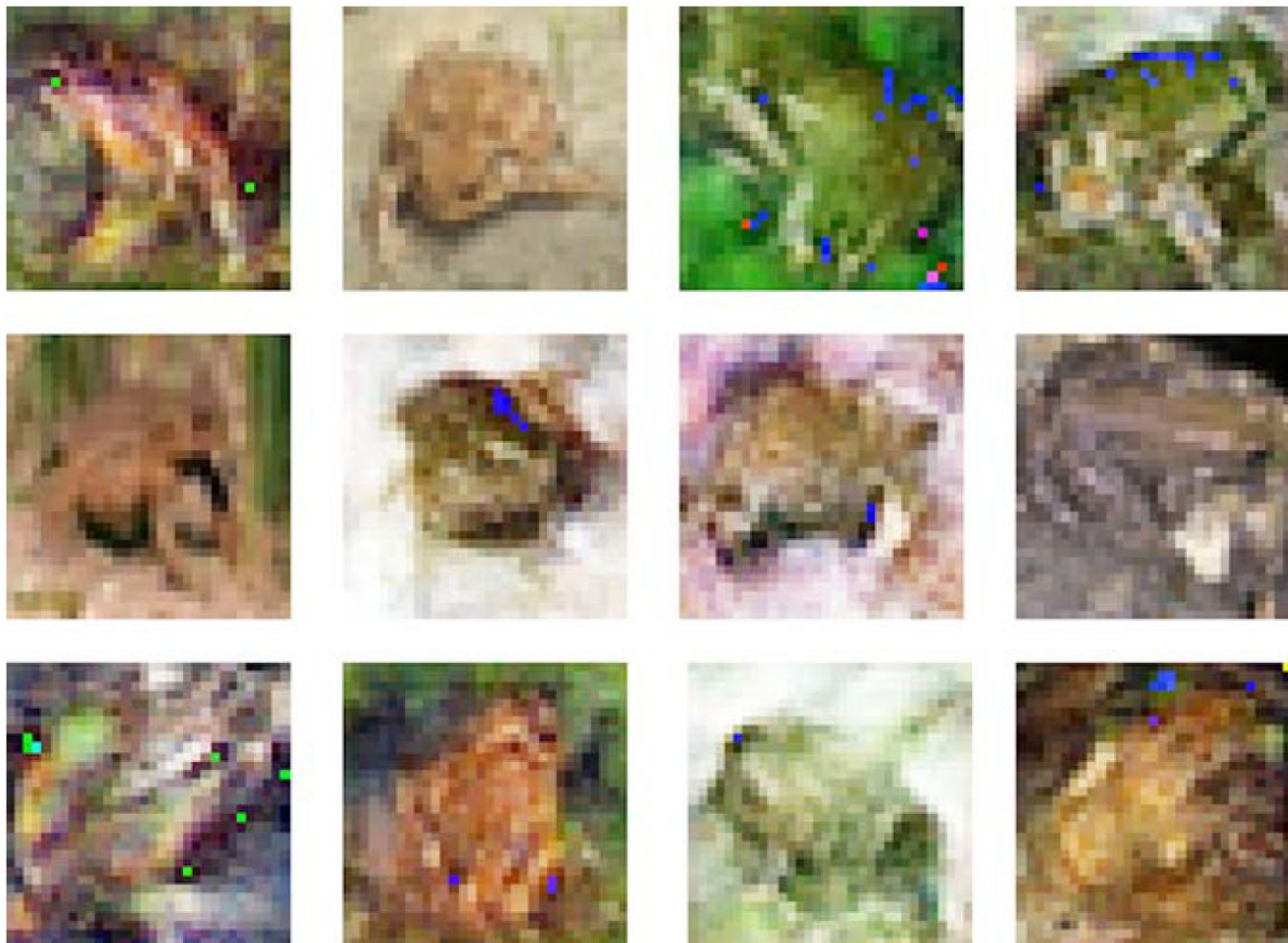
- ◎ Sparse gradients can hinder GAN training. In deep learning, sparsity is often a desirable property, but not in GANs. Two things can induce gradient sparsity: max pooling operations and ReLU activations. Instead of max pooling, it's recommended to use strided convolutions for downsampling, and using a LeakyReLU layer instead of a ReLU activation. It's similar to ReLU, but it relaxes sparsity constraints by allowing small negative activation values
- ◎ In generated images, it's common to see checkerboard artifacts caused by unequal coverage of the pixel space in the generator. To fix this, use a kernel (filter) size that's divisible by the stride size whenever you use a strided Conv2DTranpose or Conv2D in both the generator and the discriminator



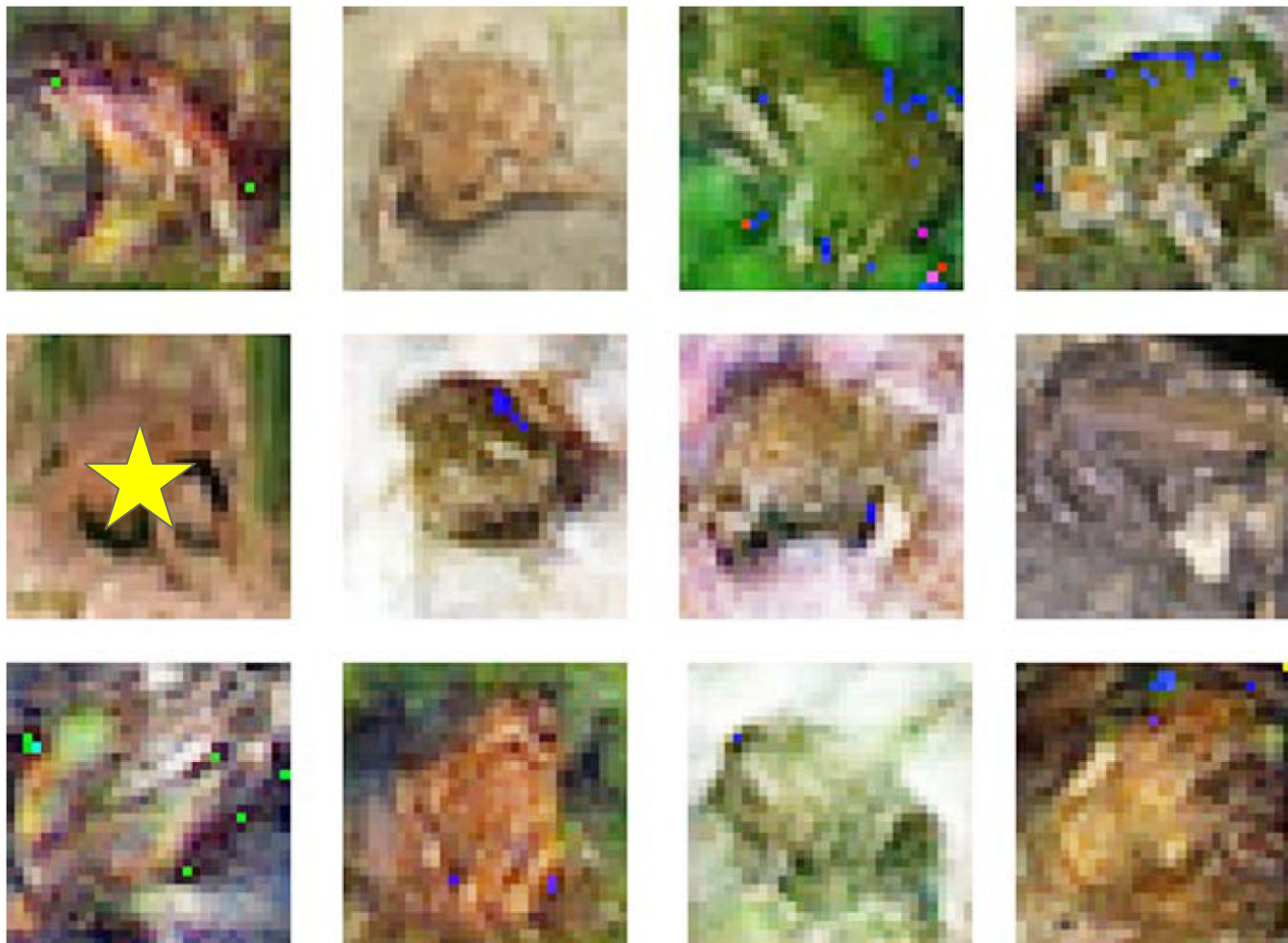
Tricks for Training

- ⊙ When training, you may see the adversarial loss begin to increase considerably, while the discriminative loss tends to zero—the discriminator may end up dominating the generator. If that's the case, try reducing the discriminator learning rate, and increase the dropout rate of the discriminator

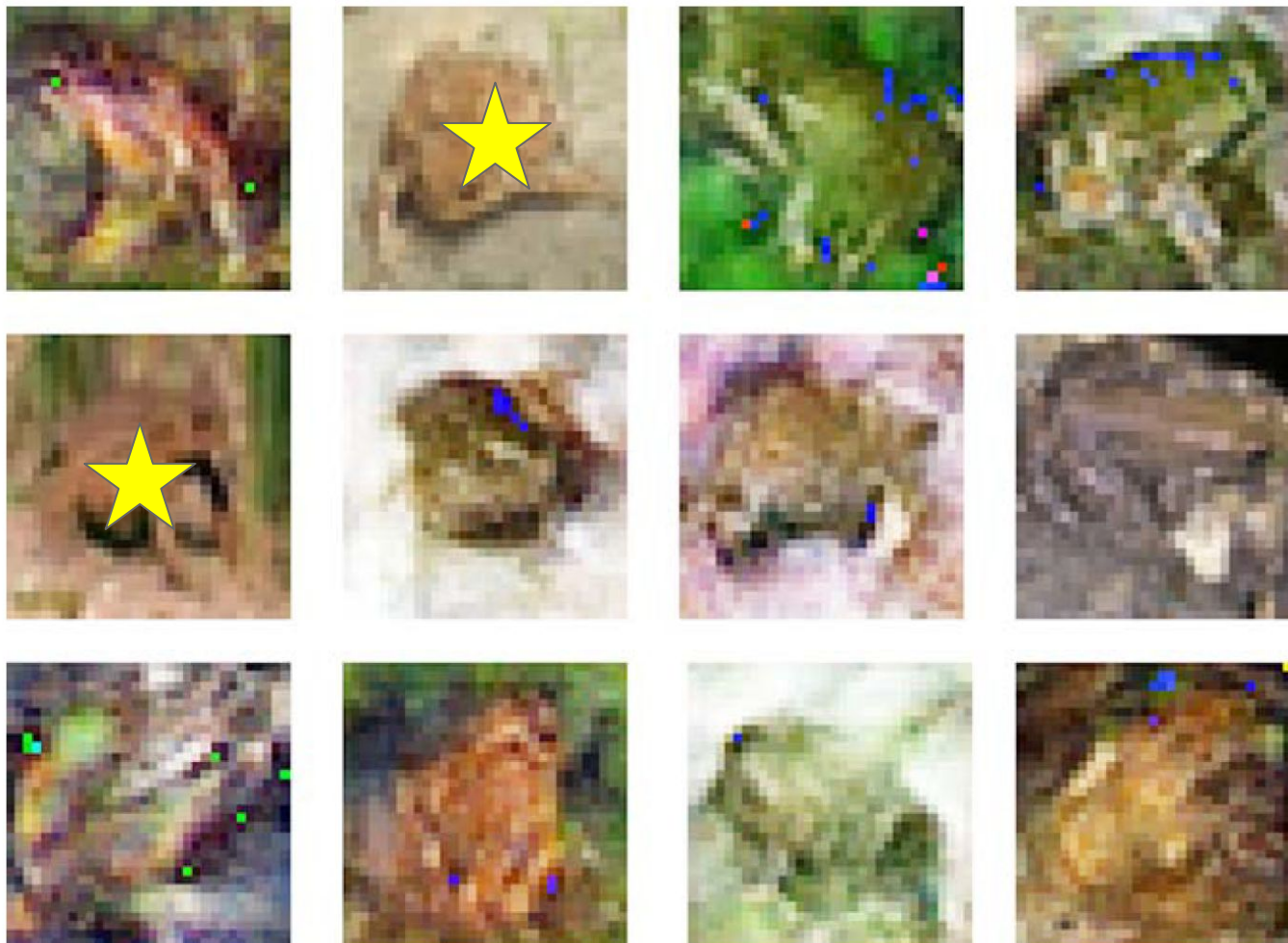
- ⦿ Can you guess which image is from the training set (not created by the generator) in each column?



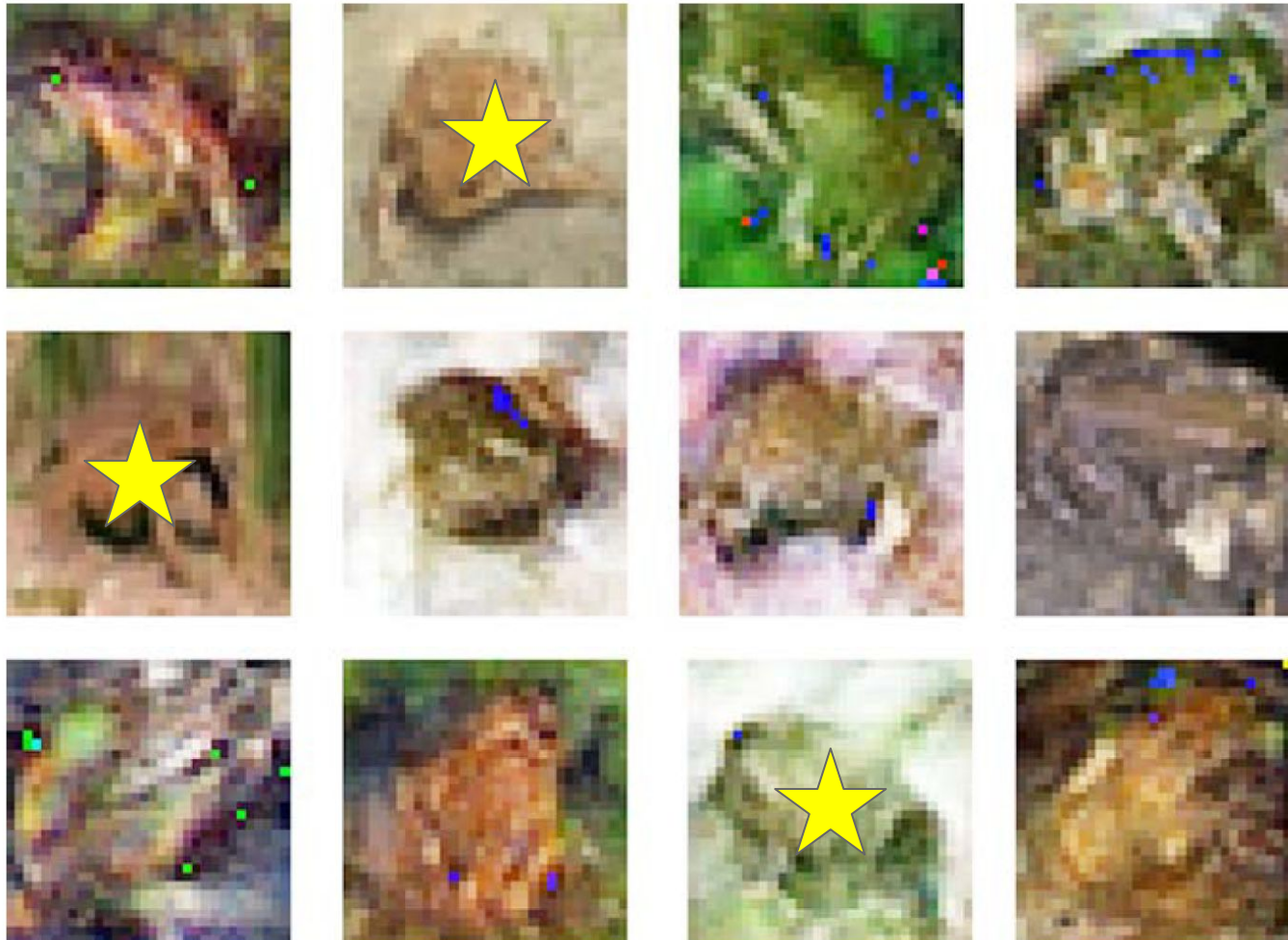
- ⦿ Can you guess which image is from the training set (not created by the generator) in each column?



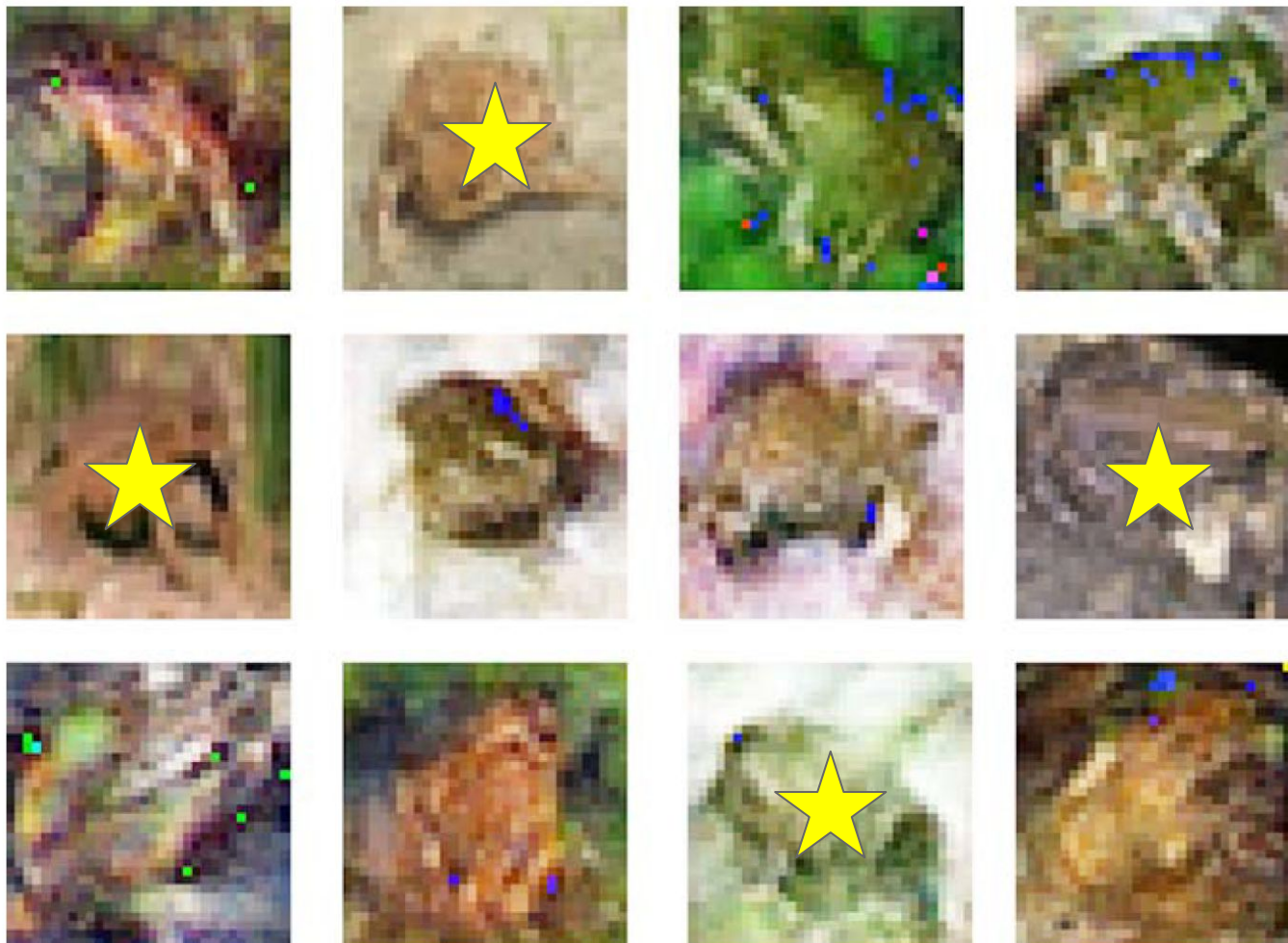
- ⦿ Can you guess which image is from the training set (not created by the generator) in each column?



- ⦿ Can you guess which image is from the training set (not created by the generator) in each column?



- ⦿ Can you guess which image is from the training set (not created by the generator) in each column?



GANs for Medicine

GANs for Medical Image Analysis

Salome Kazeminia^{a,1}, Christoph Baur^{b,1}, Arjan Kuijper^c, Bram van Ginneken^d, Nassir Navab^b, Shadi Albarqouni^b, Anirban Mukhopadhyay^a

^a*Department of Computer Science, TU Darmstadt, Germany*

^b*Computer Aided Medical Procedures (CAMP), TU Munich, Germany*

^c*Fraunhofer IGD, Darmstadt, Germany*

^d*Radboud University Medical Center, Nijmegen, The Netherlands*

Abstract

Generative Adversarial Networks (GANs) and their extensions have carved open many exciting ways to tackle well known and challenging medical image analysis problems such as medical image de-noising, reconstruction, segmentation, data simulation, detection or classification. Furthermore, their ability to synthesize images at unprecedented levels of realism also gives hope that the chronic scarcity of labeled data in the medical field can be resolved with the help of these generative models. In this review paper, a broad overview of recent literature on GANs for medical applications is given, the shortcomings and opportunities of the proposed methods are thoroughly discussed and potential future work is elaborated. We review the most relevant papers published until the submission date. For quick access, important details such as the underlying method, datasets and performance are tabulated. An interactive visualization categorizes all papers to keep the review alive².

GANs for Medicine

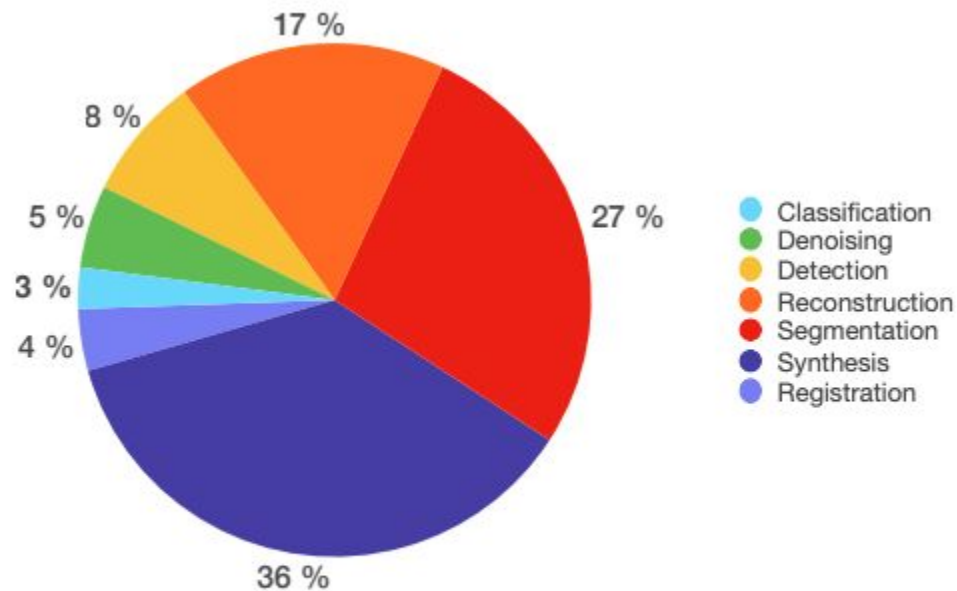


Figure 1: The distribution of papers among the different categories.

*GAN

◎ A ton of different GAN architectures have been created

- cGAN
- SeGAN
- DCGAN
- ACGAN
- CycleGAN
- MGAN
- LSGAN
- VAE-GAN
- WGAN
- ...

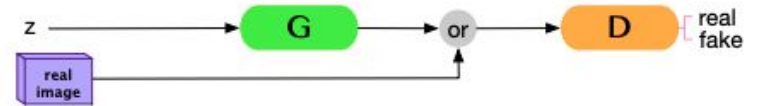


Figure 2: GAN

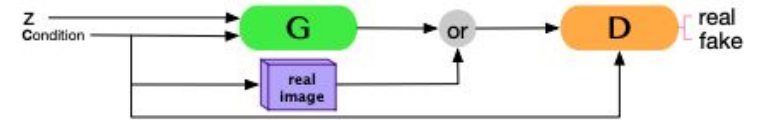


Figure 3: cGAN

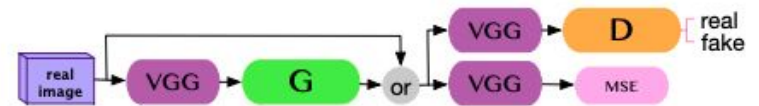


Figure 4: MGAN

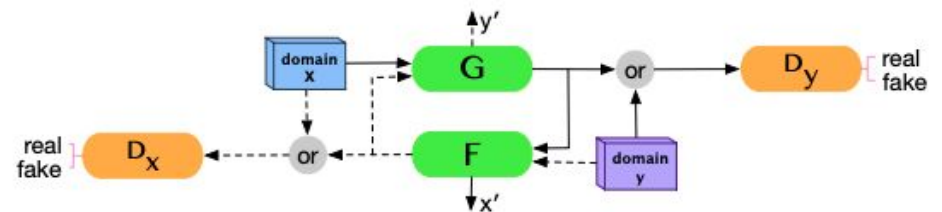


Figure 5: cycleGAN

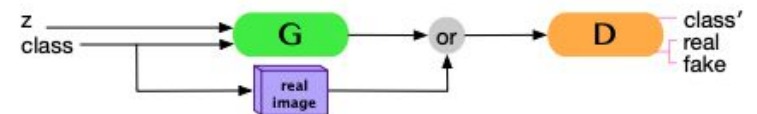


Figure 6: AC-GAN

Configuration of tree
Choose categories

Sequence of tree layers will be identical

GAN type

Filter tree
GAN typ * show

Filter database
GAN typ * or

Filter publication properties
publication tj
publication d

Advanced Options

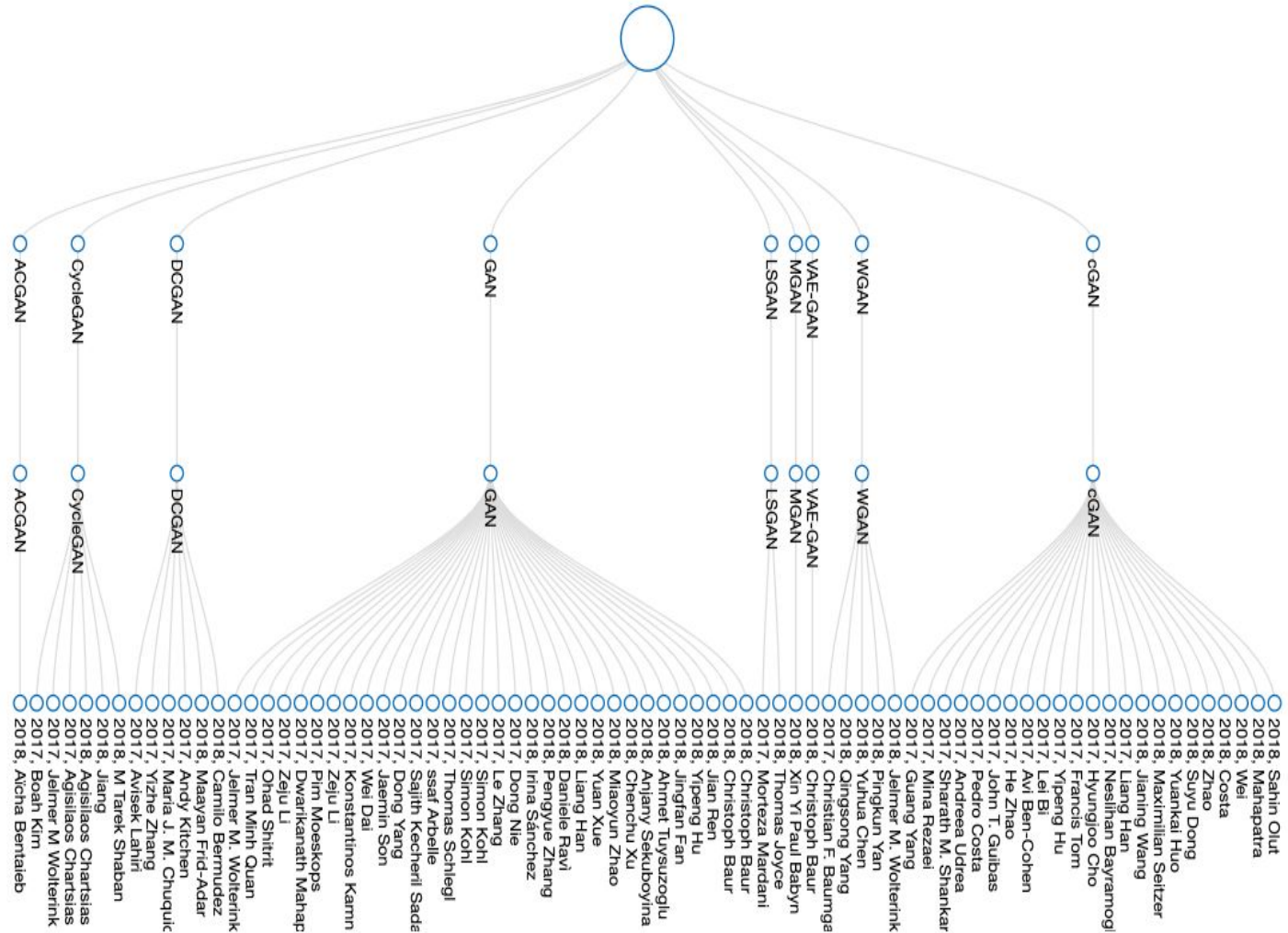
- ☒ Bézier curves
- ☒ Circles
- ☒ Mark visited links
- ☒ Enable tooltips
- ☒ Inkscape support
- ☐ Color nodes
- ☒ Draw root
- ☐ Compact
- ☐ Text shadow
- ☒ Export tooltips

Change level height

10 % 100 % 200 %

10 30 50 70 90 110 130 150 170 190 200

SVG EPS PDF



GANs for Medicine

Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery

Thomas Schlegl^{1,2} *, Philipp Seeböck^{1,2}, Sebastian M. Waldstein²,
Ursula Schmidt-Erfurth², and Georg Langs¹

¹Computational Imaging Research Lab, Department of Biomedical Imaging and
Image-guided Therapy, Medical University Vienna, Austria

thomas.schlegl@meduniwien.ac.at

²Christian Doppler Laboratory for Ophthalmic Image Analysis, Department of
Ophthalmology and Optometry, Medical University Vienna, Austria

Abstract. Obtaining models that capture imaging markers relevant for disease progression and treatment monitoring is challenging. Models are typically based on large amounts of data with annotated examples of known markers aiming at automating detection. High annotation effort and the limitation to a vocabulary of known markers limit the power of such approaches. Here, we perform unsupervised learning to identify anomalies in imaging data as candidates for markers. We propose *AnoGAN*, a deep convolutional generative adversarial network to learn a manifold of normal anatomical variability, accompanying a novel anomaly scoring scheme based on the mapping from image space to a latent space. Applied to new data, the model labels anomalies, and scores image patches indicating their fit into the learned distribution. Results on optical coherence tomography images of the retina demonstrate that the approach correctly identifies anomalous images, such as images containing retinal fluid or hyperreflective foci.

GANs for Medicine

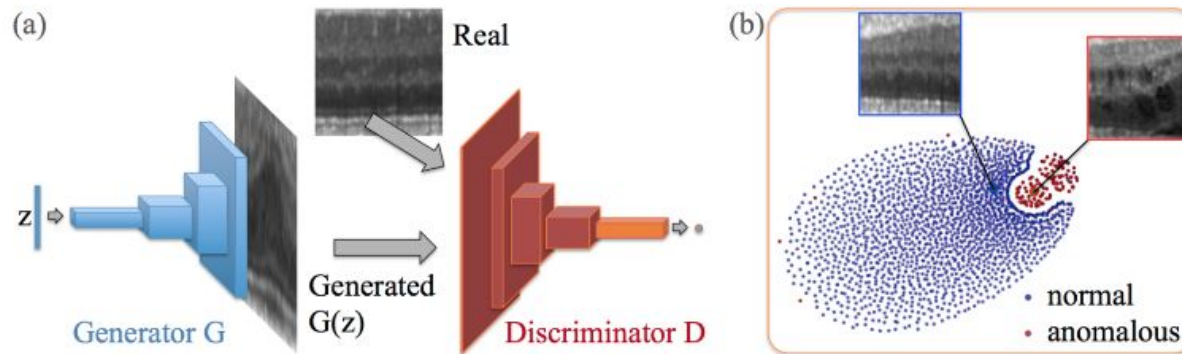
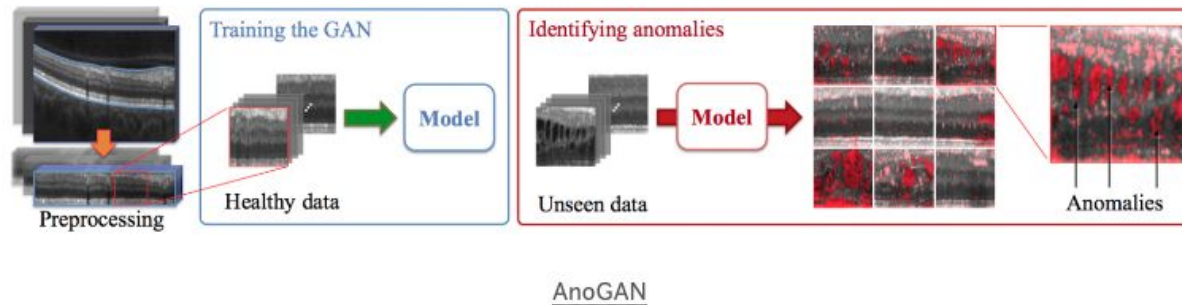


Fig. 2. (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator.

AnoGAN