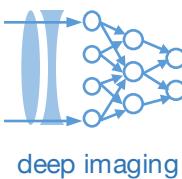


Lecture 23: Looking ahead – machine learning and imaging in 10 years

Machine Learning and Imaging

BME 590L
Roarke Horstmeyer



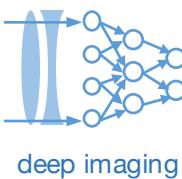
Announcements and schedule

- Last Lecture: Tuesday 11/26, course review
- TA's will hold labs on M/W next week, *and M/W the week after*
- Homework #5 Due: Tuesday December 3, 6 pm
- Friday 12/13, 7 pm – 10 pm: Final projects due and presentations
- Project help:
 - I will continue my office hours until Dec. 13th
 - Wednesday 10am – 11:30am, Thursday 11:15am – 12:45pm
 - Email me if you'd like to meet another time
 - Go to TA Labs on M/W December 2-4
 - Email TA's / reach out on Slack to meet them as well

Components of final project

40% of total grade

1. Presentation Slides – 10%
 - 7 minute presentation, 1 minute for questions
2. 4-6 page write up with at least 3 figures and 5 references – 20%
 - Introduction, related work, methods, results, discussion
3. Code used for final results in folder or .ipynb's – 5%
4. brief website template & permission to share results – 5%
5. shared annotated datasets & permissions – no grade, but would be much appreciated if using an interesting dataset



Final project webpage

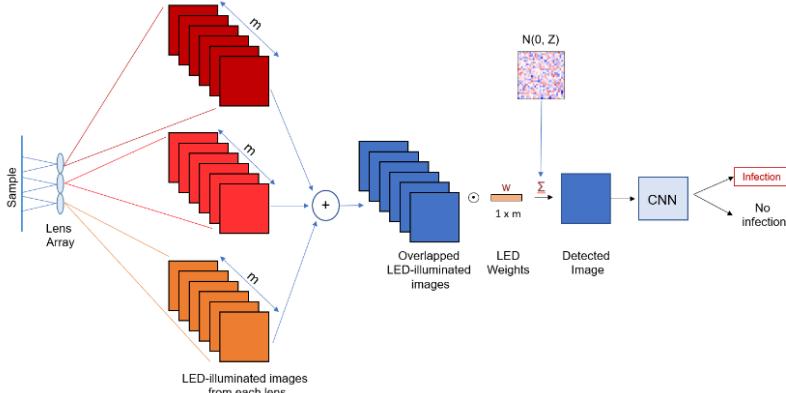
- Must be submitted
- Will share template
- Will post to deepimaging.io with permission
- Will also send permission form, which must be submitted with final project as well

Optimizing illumination for overlapped classification

Ameey A Chaware

amey.chaware@duke.edu

[Paper PDF](#)



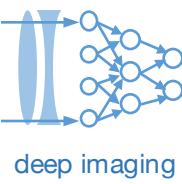
This project presents an imaging system that simultaneously captures multiple images and automatically classifies their contents to increase detection throughput. Our optical design consists of a set of multiple lenses that each image a unique field-of-view onto a single image sensor. The resulting “overlapped” image exhibits reduced contrast, but includes measurements from across a proportionally larger viewing area. We then post-process this overlapped image with a deep convolutional neural network to classify the presence or absence of certain features of interest. We examine the specific case of detecting the malaria parasite within overlapped microscope images of blood smears. We demonstrate that it is possible to overlap 7 unique images onto a common sensor while still offering accurate classification of the presence or absence of the parasite, thus offering a 7x potential speed-up for automated disease diagnosis with microscope image data. Additionally, we explore the use of supervised deep-learning network to jointly optimize the physical setup of an optical microscope to improve automatic image classification accuracy in overlapped imaging. We take advantage of the wide degree of flexibility available in choosing how a sample is illuminated in a microscope to design a specific pattern of light that leads to a better performance.

Paper:

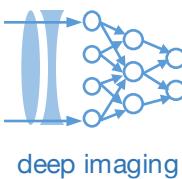
- [Paper PDF](#)

Code and Data:

- You can provide a link to your code here: [Code](#)



Where are things going with Machine Learning and Imaging in 10 years?

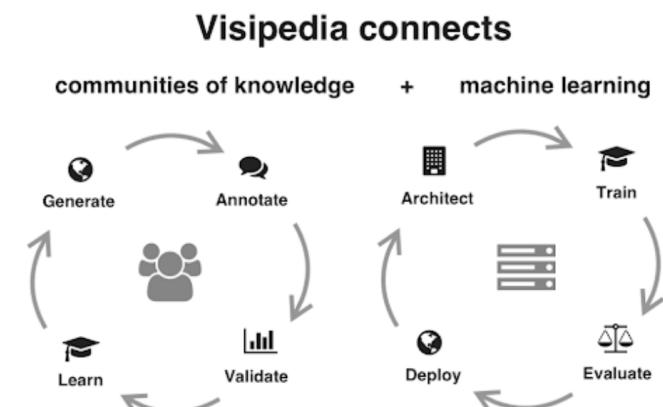


Where are things going with Machine Learning and Imaging in 10 years?

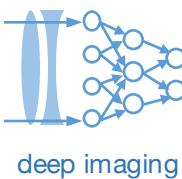
1. Proliferation of trained models, similar datasets and shared goals

The image shows two screenshots side-by-side. On the left is the OpenAI website, which has a purple header with the OpenAI logo and the URL <https://openai.com/>. Below the header are three buttons: 'Repositories 76', 'People 0', and 'Projects 0'. On the right is the Skymind AI Platform website, which has a dark blue header with the Skymind logo and the URL [https://skymind.ai](#). Below the header is a navigation bar with links: 'Ai Platform', 'Why Skymind?', 'Solutions', 'Case Studies', 'About', 'Resources', and 'CONTACT US'. The main content area features the text 'A.I. Wiki'.

Visipedia



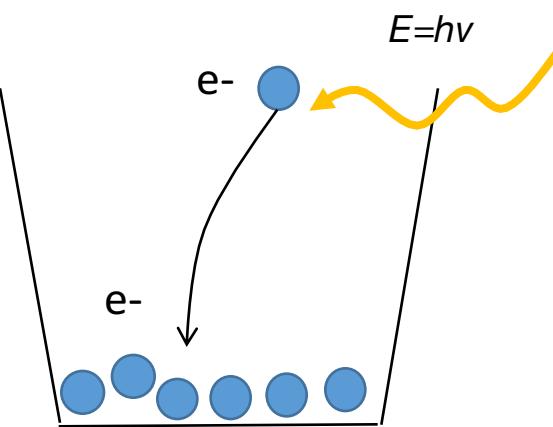
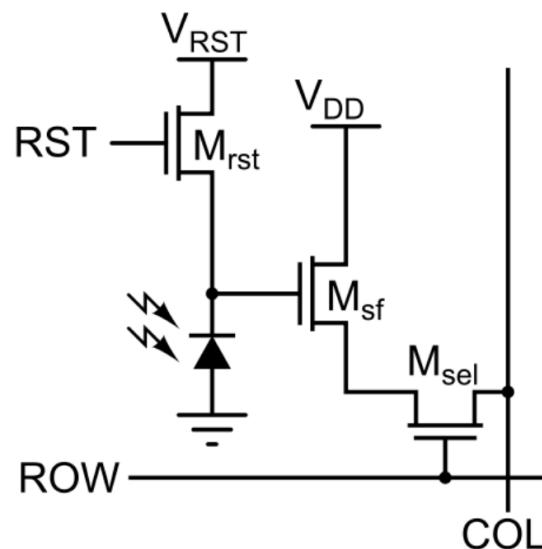
Caltech Visipedia



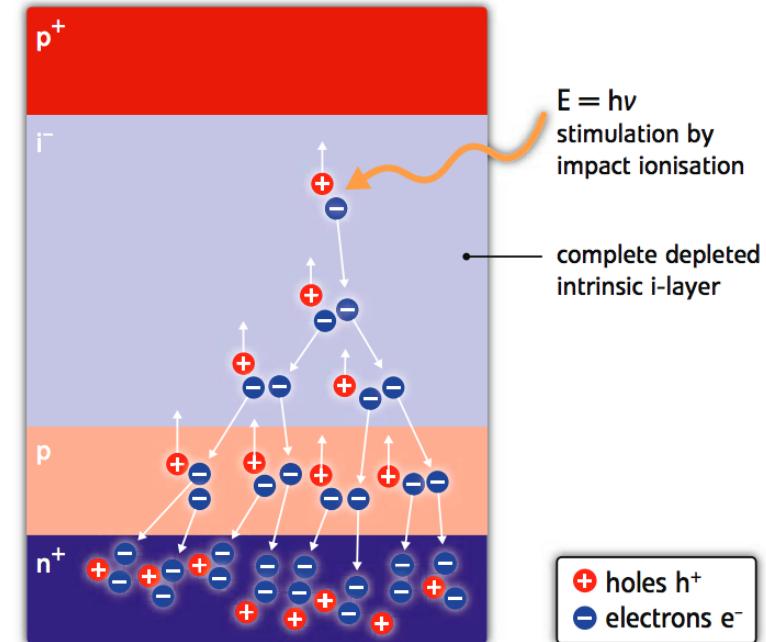
Where are things going with Machine Learning and Imaging in 10 years?

2. “Cameras” on many devices & new types of sensors

Standard CMOS pixel = bucket that collects electrons

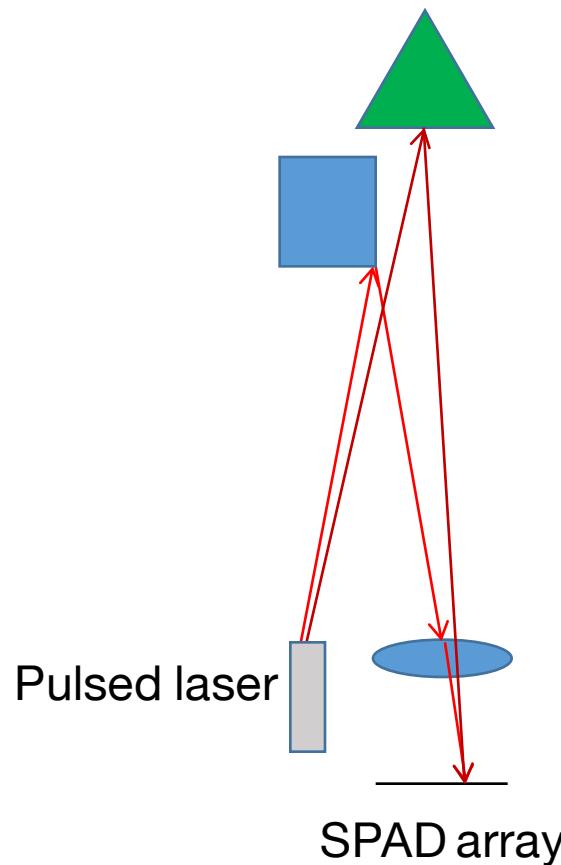


SPAD pixel: was there a photon or not?

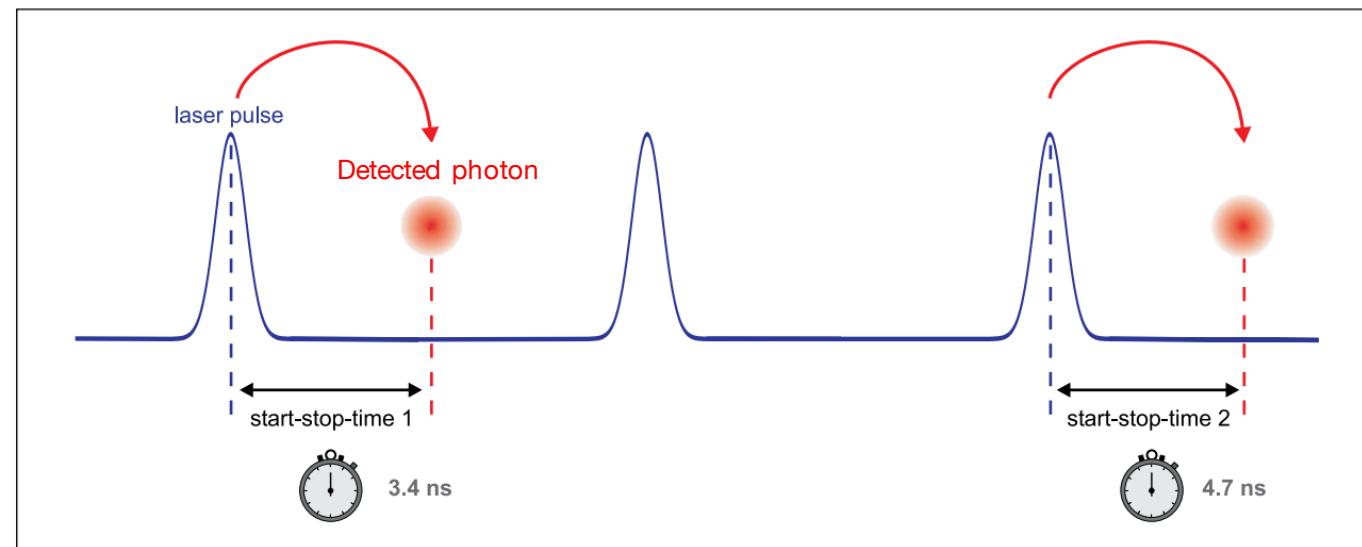


Where are things going with Machine Learning and Imaging in 10 years?

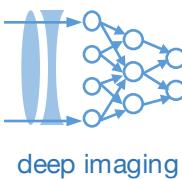
2. “Cameras” on many devices & new types of sensors



- Light travels 1 ft in 1 ns.
- SPADs can precisely photon arrival time to measure travel distance (TOF)



https://www.picoquant.com/images/uploads/page/files/7253/technote_tcspc.pdf



Where are things going with Machine Learning and Imaging in 10 years?

2. “Cameras” on many devices & new types of sensors

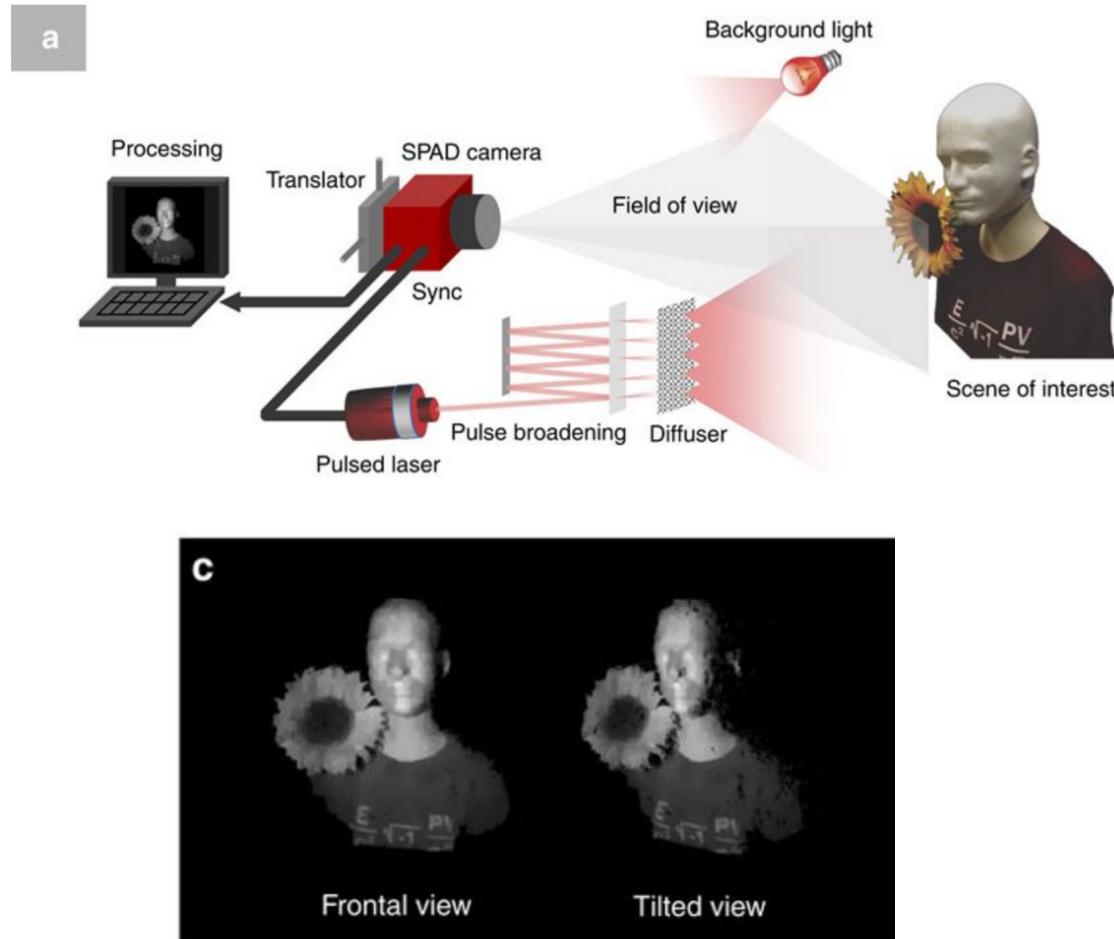
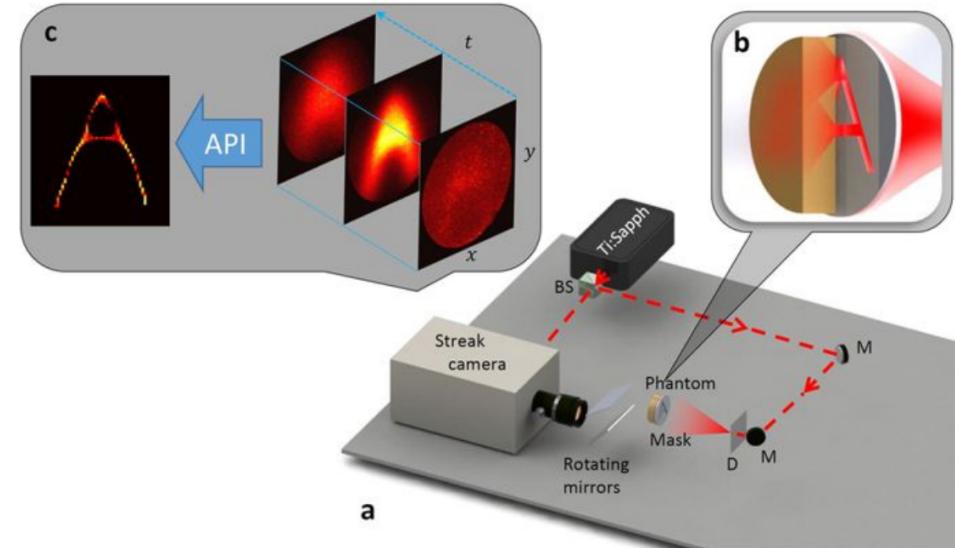


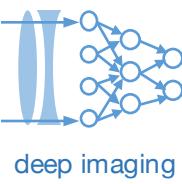
Figure 1: Imaging Through Thick Scattering.



G. Satat et al, <https://www.nature.com/articles/srep33946>

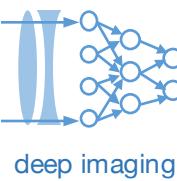
D. Shen et al, <https://www.nature.com/articles/ncomms12046>

Machine Learning and Imaging – Roarke Horstmeyer (2019)



Where are things going with Machine Learning and Imaging in 10 years?

3. Beyond convolutions - new constructs for deep networks



Where are things going with Machine Learning and Imaging in 10 years?

3. Beyond convolutions - new constructs for deep networks

Dynamic Routing Between Capsules

Sara Sabour

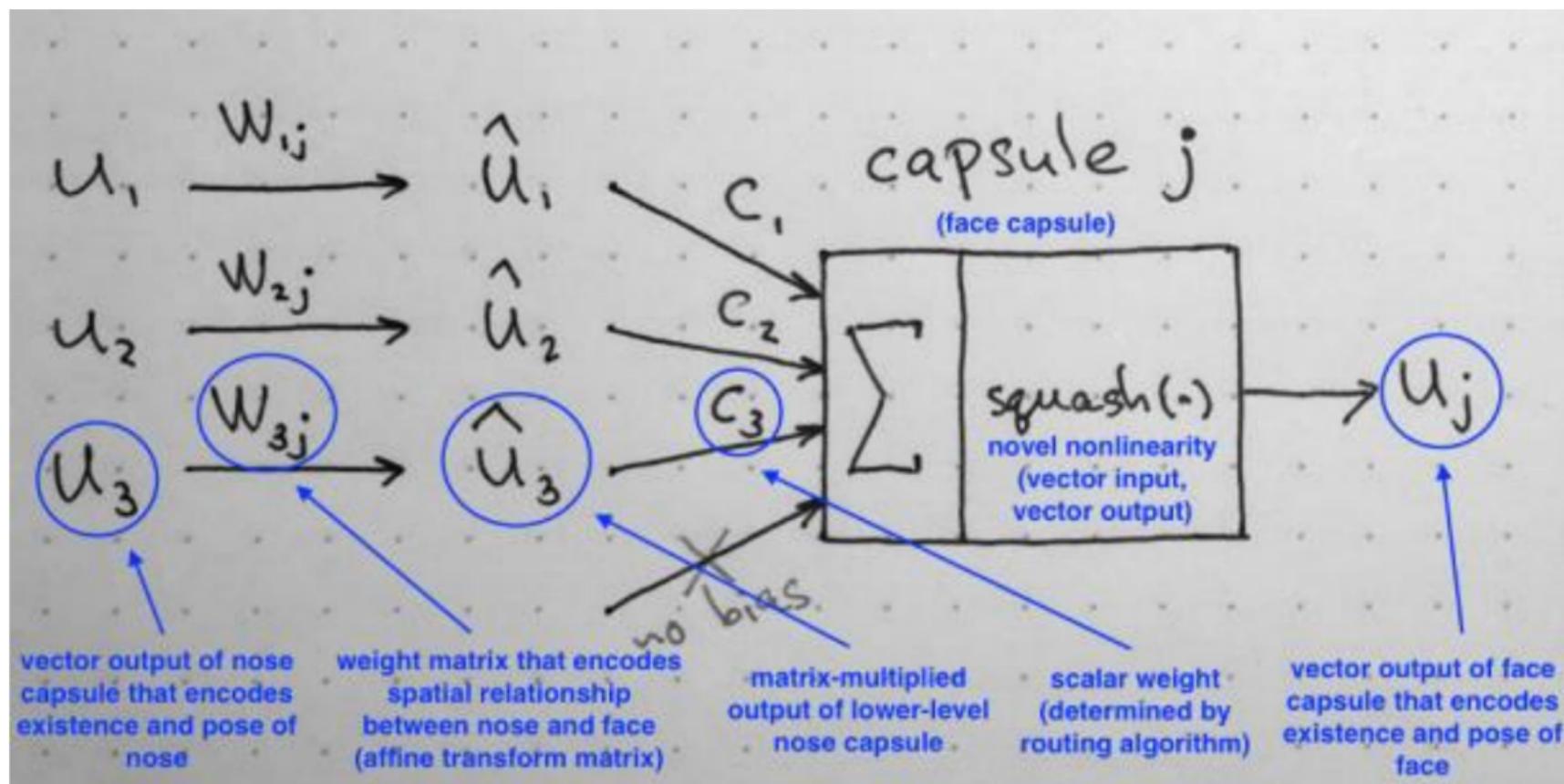
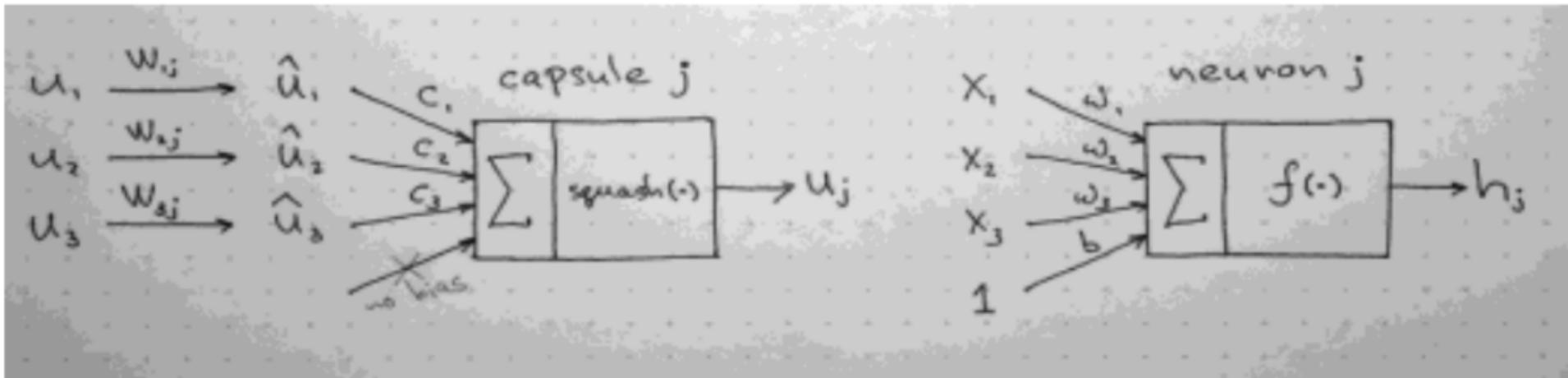
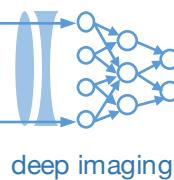
Nicholas Frosst

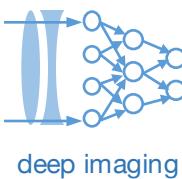
Geoffrey E. Hinton

Google Brain

Toronto

{sasabour, frosst, geoffhinton}@google.com





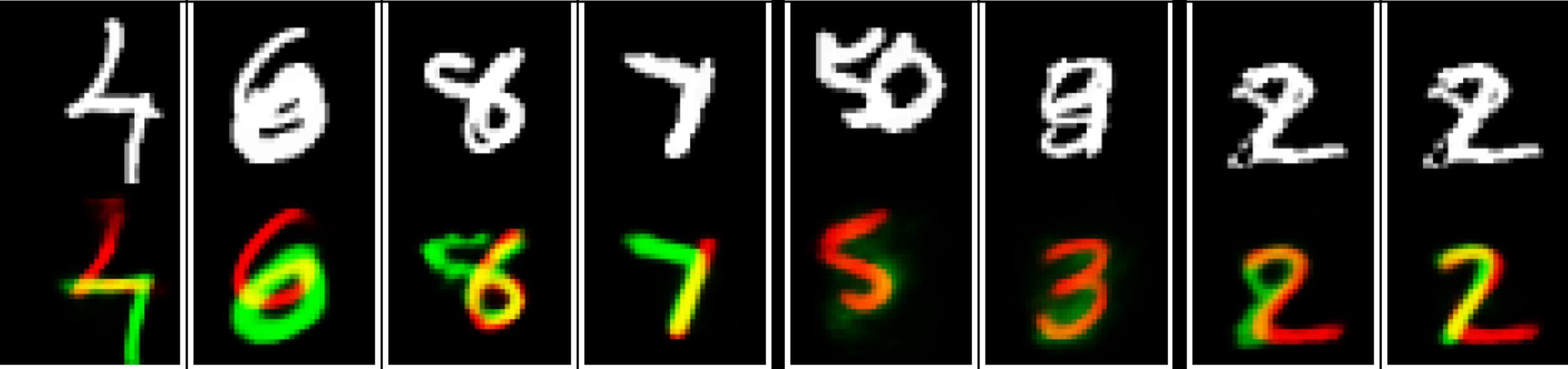
Where are things going with Machine Learning and Imaging in 10 years?

3. Beyond convolutions - new constructs for deep networks

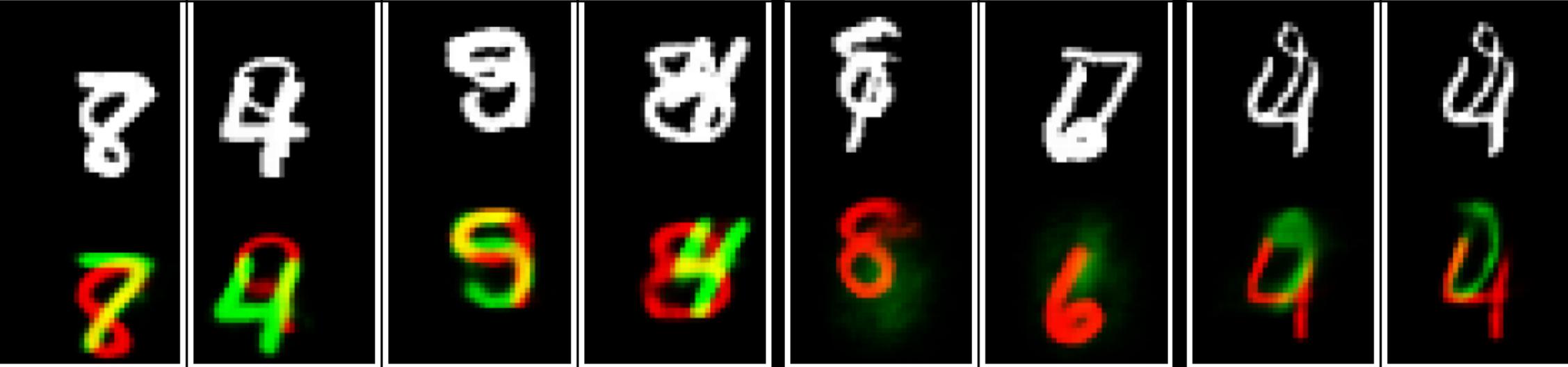
Capsule vs. Traditional Neuron		
Input from low-level capsule/neuron	vector(\mathbf{u}_i)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j i} = \mathbf{W}_{ij}\mathbf{u}_i$
	Weighting	$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j i}$
	Sum	$a_j = \sum_i w_i x_i + b$
	Nonlinear Activation	$\mathbf{v}_j = \frac{\ \mathbf{s}_j\ ^2}{1+\ \mathbf{s}_j\ ^2} \frac{\mathbf{s}_j}{\ \mathbf{s}_j\ }$
		$h_j = f(a_j)$
Output	vector(\mathbf{v}_j)	scalar(h_j)

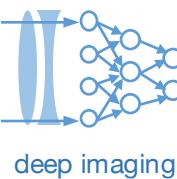
3. Therefore R will not assign one pixel to two digits if one of them does not have any other support.

R:(2, 7) L:(2, 7)	R:(6, 0) L:(6, 0)	R:(6, 8) L:(6, 8)	R:(7, 1) L:(7, 1)	*R:(5, 7) L:(5, 0)	*R:(2, 3) L:(4, 3)	R:(2, 8) L:(2, 8)	R:P:(2, 7) L:(2, 8)
----------------------	----------------------	----------------------	----------------------	-----------------------	-----------------------	----------------------	------------------------



R:(8, 7) L:(8, 7)	R:(9, 4) L:(9, 4)	R:(9, 5) L:(9, 5)	R:(8, 4) L:(8, 4)	*R:(0, 8) L:(1, 8)	*R:(1, 6) L:(7, 6)	R:(4, 9) L:(4, 9)	R:P:(4, 0) L:(4, 9)
----------------------	----------------------	----------------------	----------------------	-----------------------	-----------------------	----------------------	------------------------





Where are things going with Machine Learning and Imaging in 10 years?

4. Generative data is getting pretty realistic...



Face2Face: Real-time Face Capture and Reenactment of RGB Videos

Justus Thies¹

Michael Zollhöfer²

Marc Stamminger³

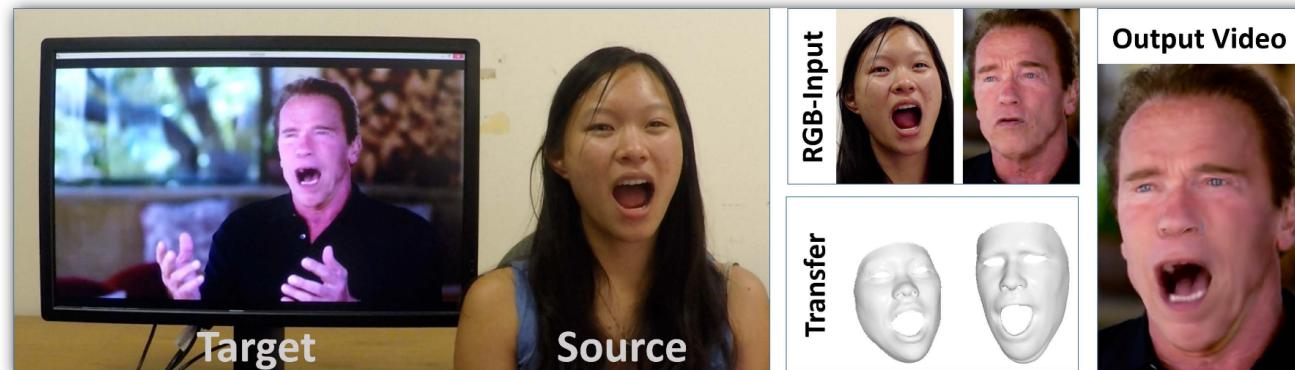
Christian Theobalt²

Matthias Nießner¹

¹ Technical University of Munich

² Max Planck Institute for Informatics

³ University of Erlangen-Nuremberg

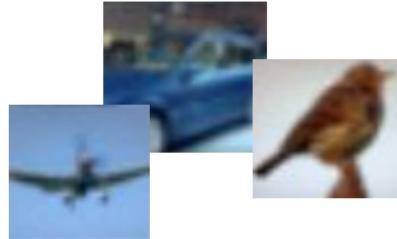


Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, June 2016

What are the implications of this for medical imaging?

Generative Models

Given training data, generate new samples from same distribution



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

Generative Models

Given training data, generate new samples from same distribution



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{\text{model}}(x)$
- Implicit density estimation: learn model that can sample from $p_{\text{model}}(x)$ w/o explicitly defining it

Taxonomy of Generative Models

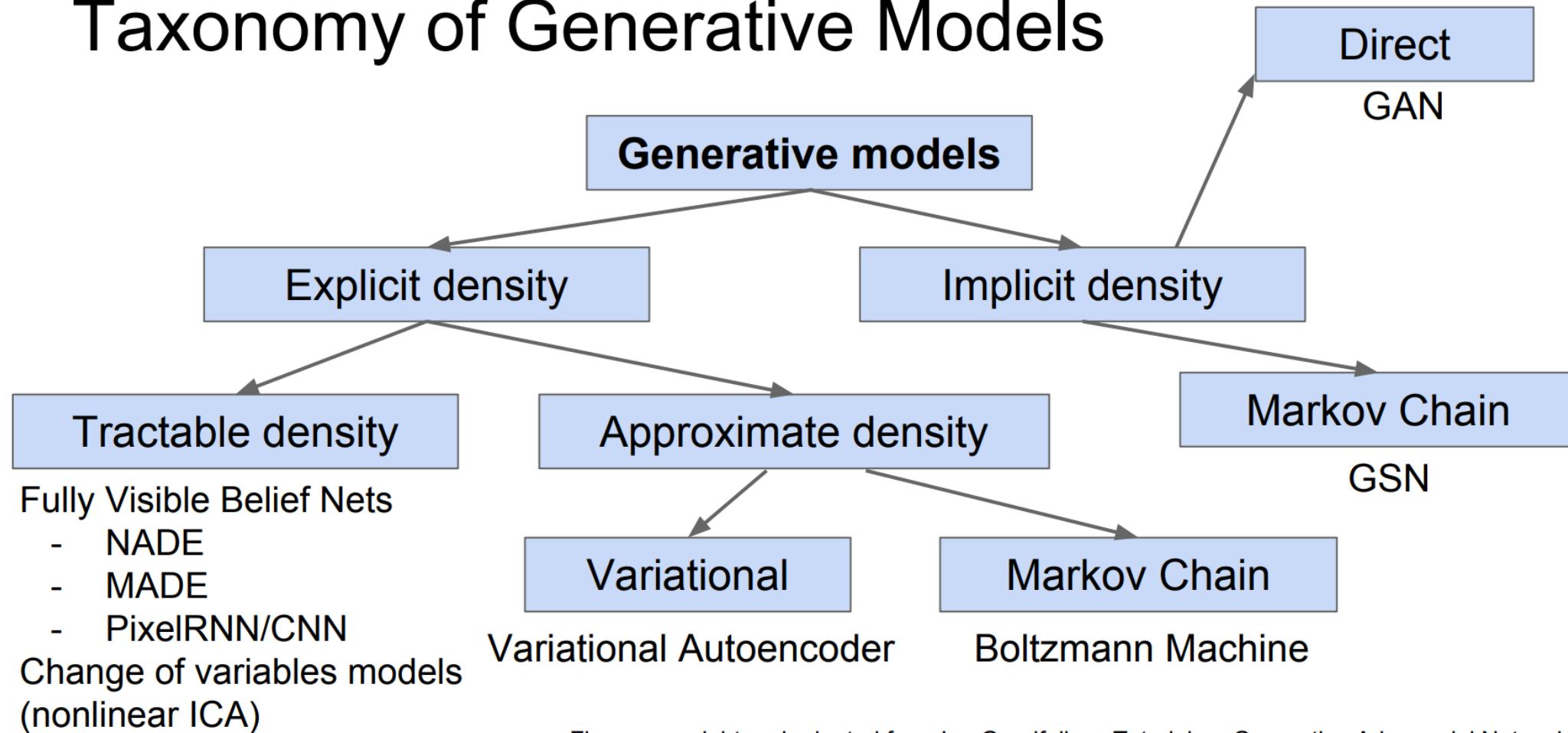


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

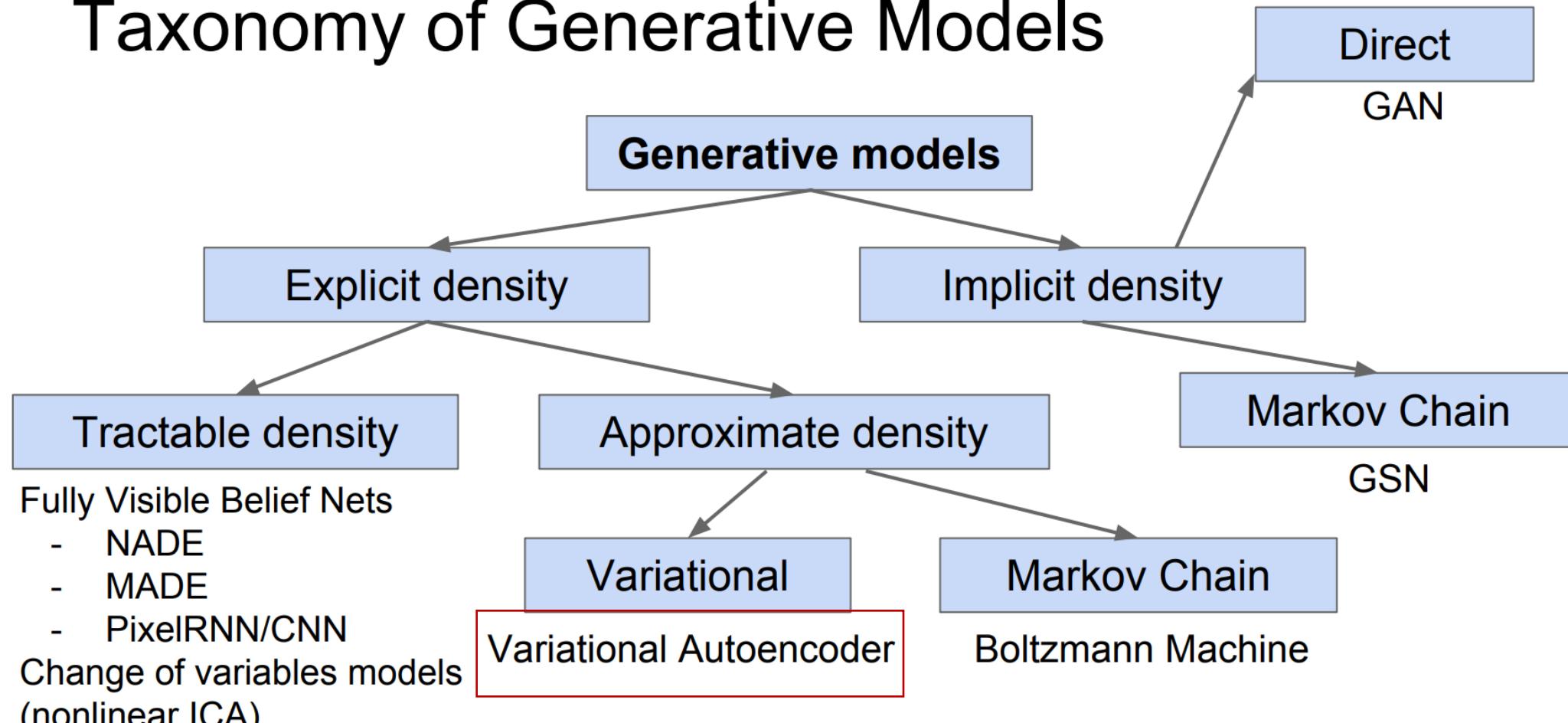
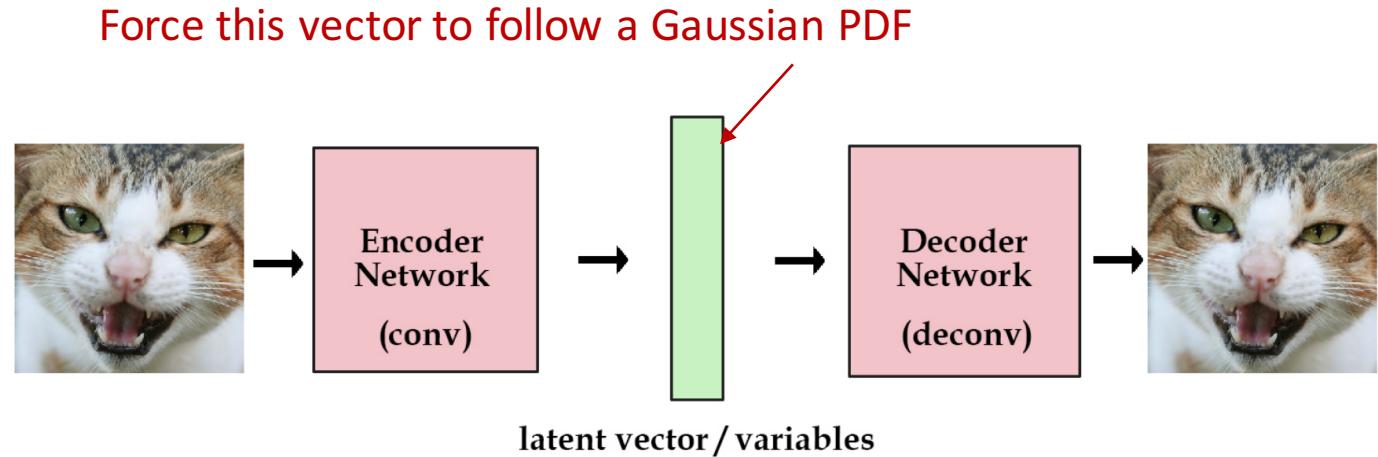


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

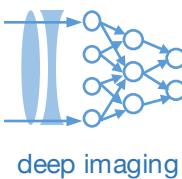
Example: Variational Autoencoder (VAE)



- Good *generative model*
- Have a clean probability distribution to select from to generate new examples

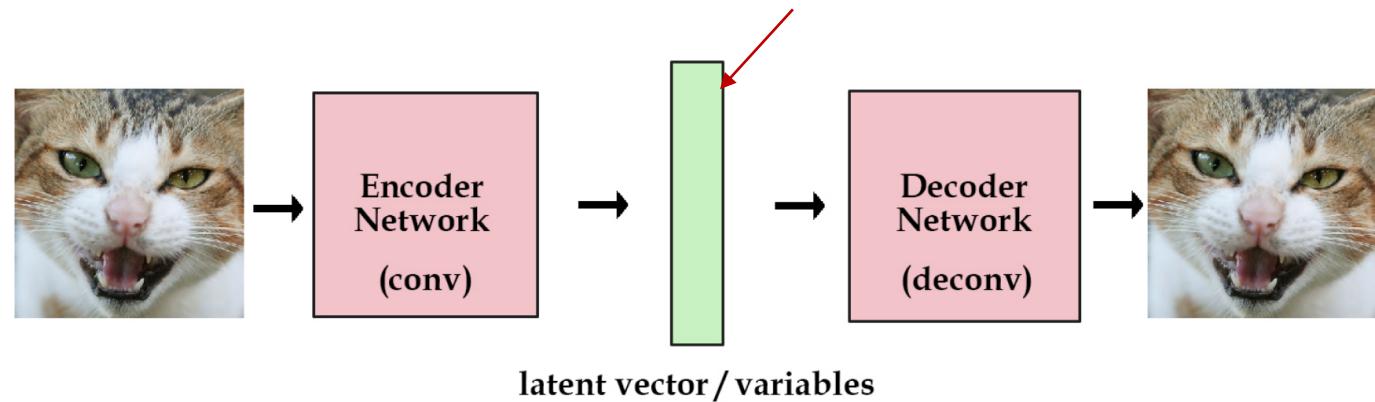
Minimize (KL) distance between latent vector and Gaussian normal





Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

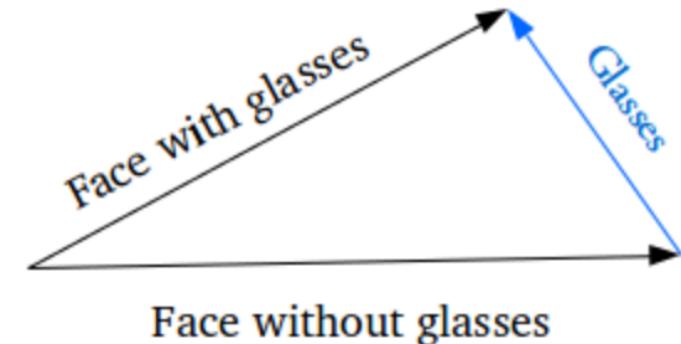


Minimize (KL) distance between latent vector and Gaussian normal

Generative Example (once trained):

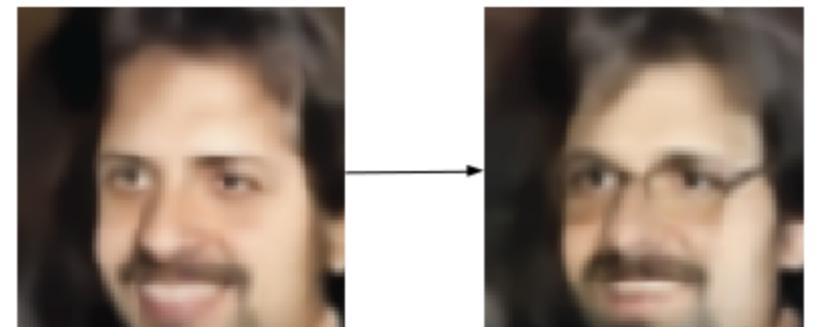
- Encode image with glasses, obtain latent vector PDF \mathbf{P}_g
- Encode image without glasses, obtain PDF \mathbf{P}_{ng}
- Compute $\mathbf{diff} = \mathbf{P}_g - \mathbf{P}_{ng}$
- Encode new image to obtain \mathbf{P}_{new} , add in \mathbf{diff}
- Decode $\mathbf{P}_{new} + \mathbf{diff}$ to get guy with glasses!

- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples

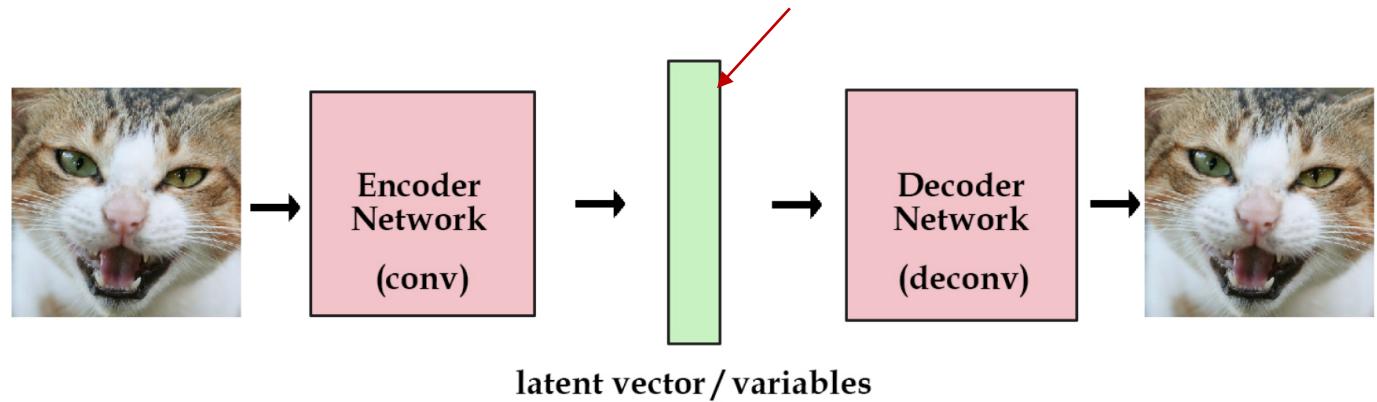
Glasses



Exploring a specific variation of input data[1]

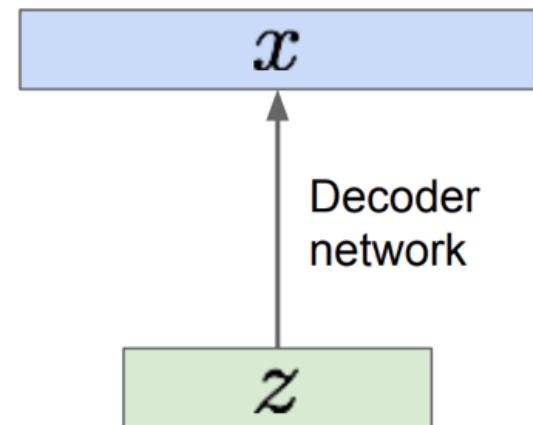
Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



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

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



Taxonomy of Generative Models

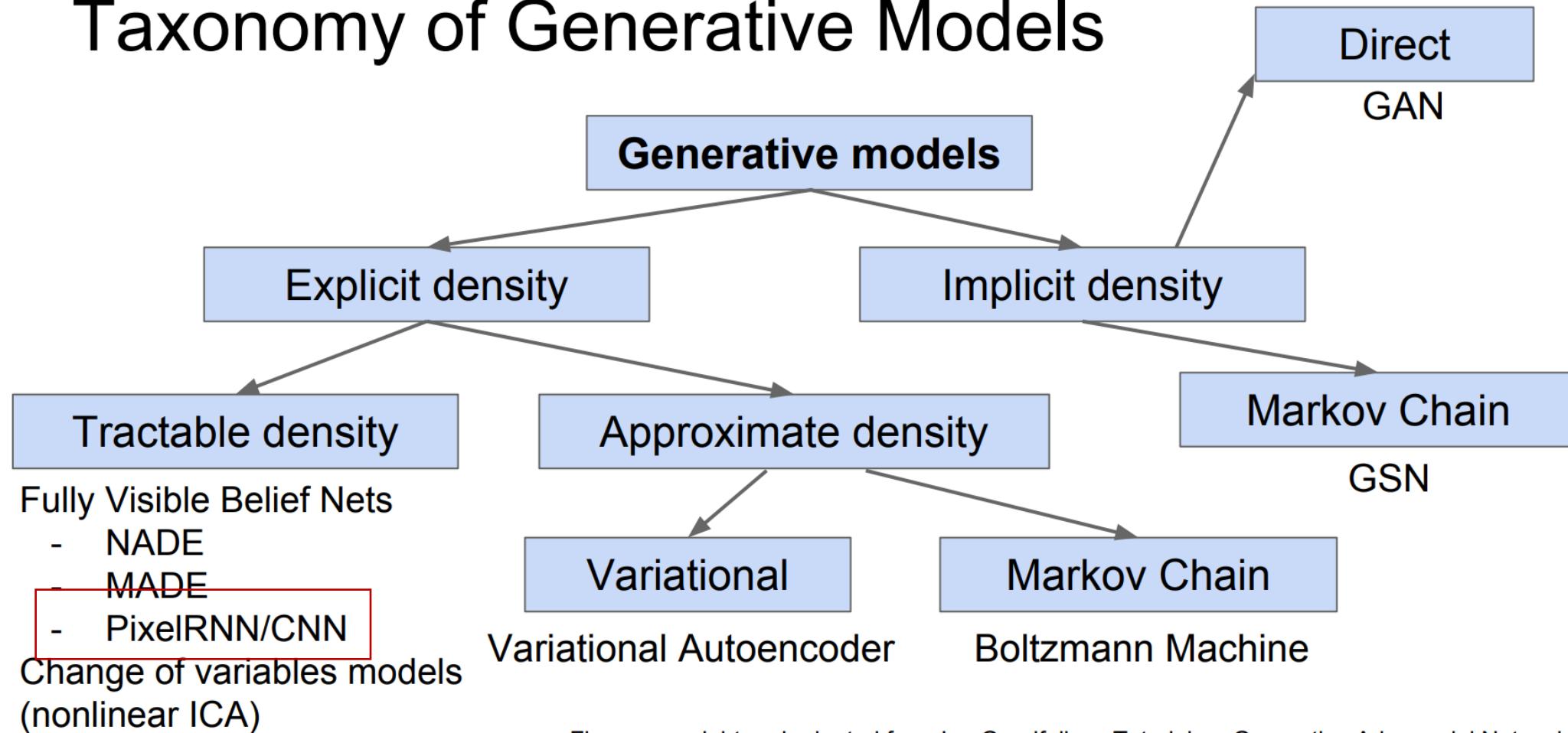
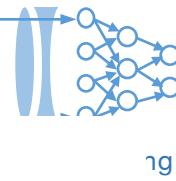


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.



Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

↑ ↑
Likelihood of Probability of i'th pixel value
image x given all previous pixels

Then maximize likelihood of training data

Stanford CS231n, Lecture 12

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

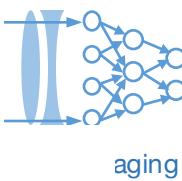
↑ ↑
Likelihood of Probability of i'th pixel value
image x given all previous pixels

This is a really complex distribution, obviously

Simplify by going through image pixel by pixel, rely on RNN

Then maximize likelihood of training data

Stanford CS231n, Lecture 12

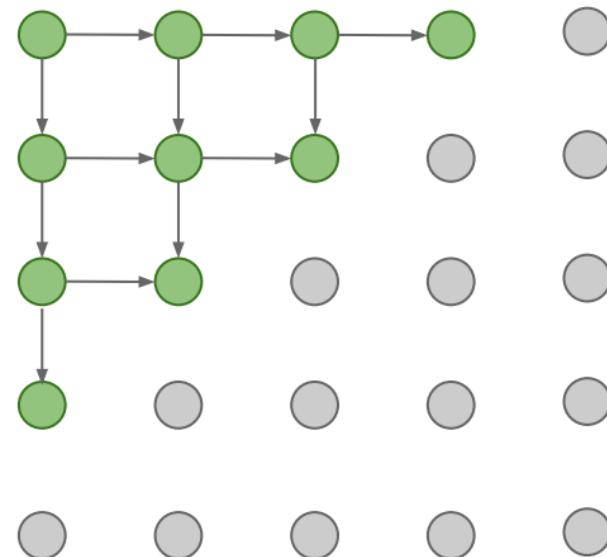


PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow!



Stanford CS231n, Lecture 12



Figure 1. Image completions sampled from a PixelRNN.

A. Van der Oord et al., <https://arxiv.org/abs/1601.06759>

PixelCNN

[van der Oord et al. 2016]

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training: maximize likelihood of training images

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

Softmax loss at each pixel

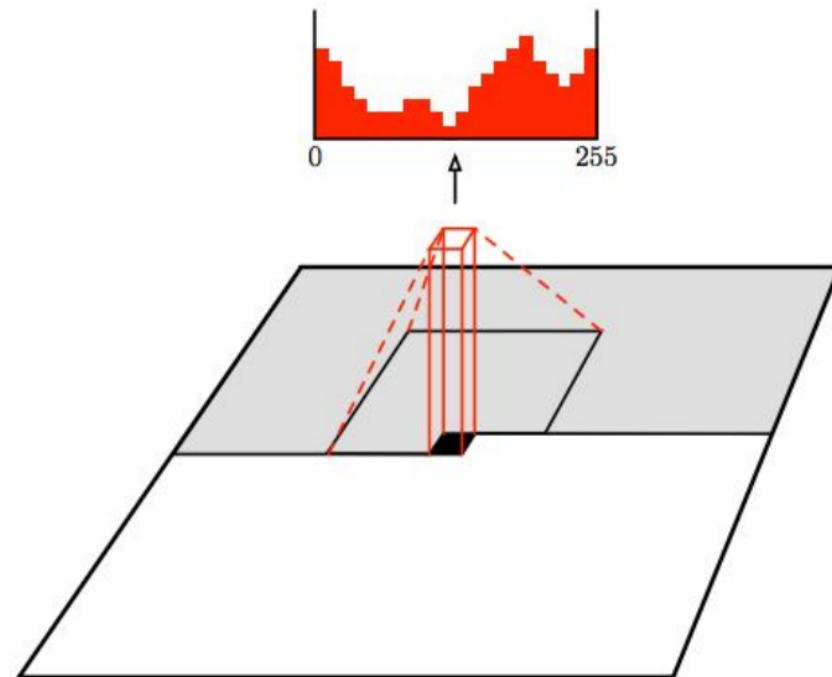
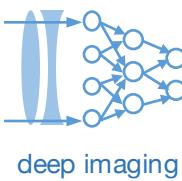


Figure copyright van der Oord et al., 2016. Reproduced with permission.

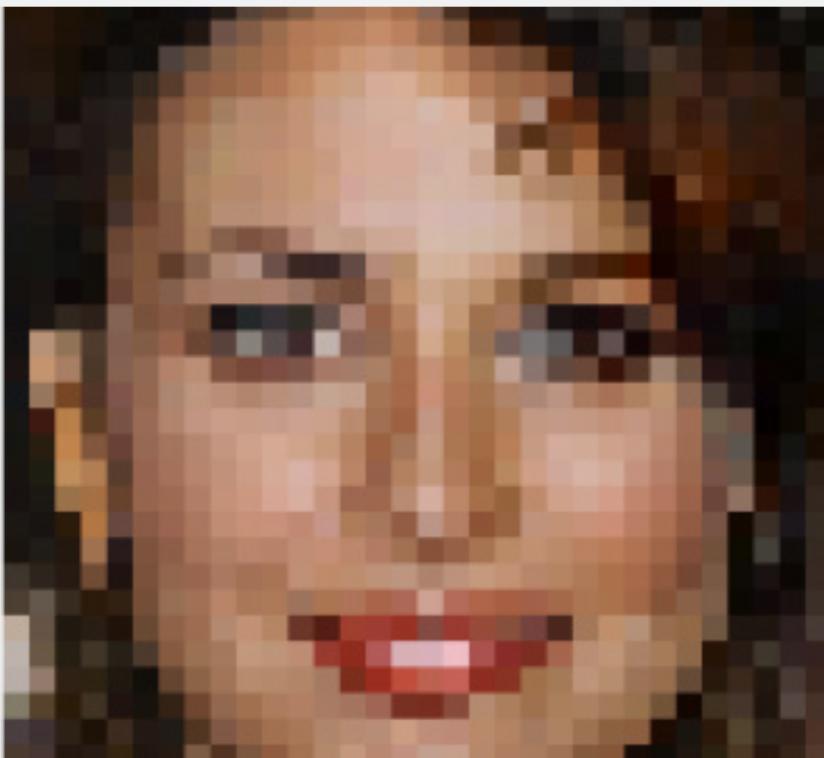
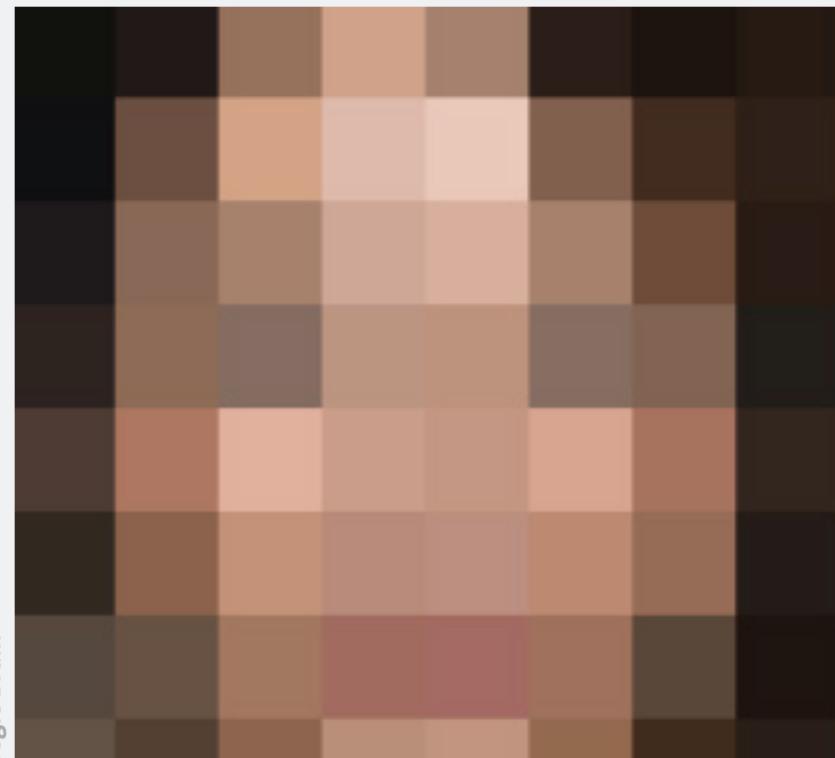


GARBAGE IN, NON-GARBAGE OUT —

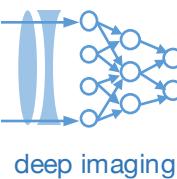
Google Brain super-resolution image tech makes “zoom, enhance!” real

Google Brain creates new image details out of thin air.

SEBASTIAN ANTHONY - 2/7/2017, 8:38 AM



vugie drulli



So far...

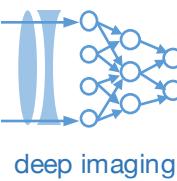
PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_{\theta}(x) = \prod_{i=1}^n p_{\theta}(x_i|x_1, \dots, x_{i-1})$$

VAEs define intractable density function with latent \mathbf{z} :

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead



So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_{\theta}(x) = \prod_{i=1}^n p_{\theta}(x_i|x_1, \dots, x_{i-1})$$

VAEs define intractable density function with latent \mathbf{z} :

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

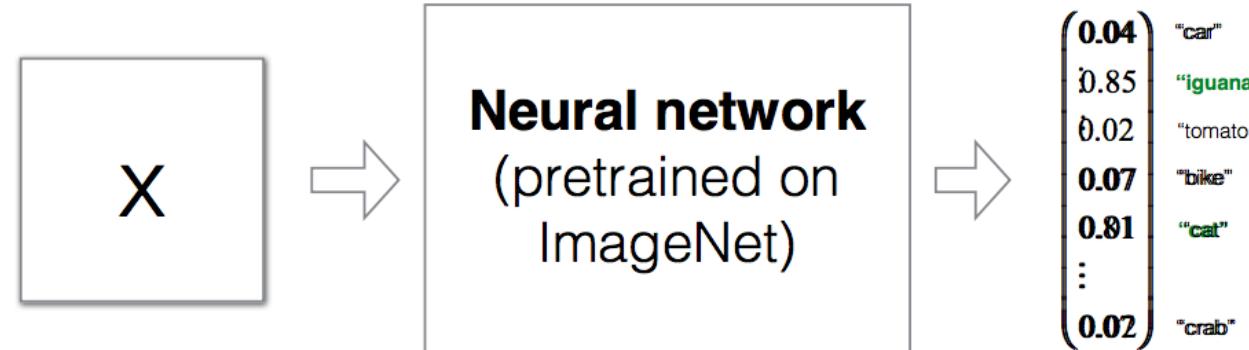
What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don't work with any explicit density function!

Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

I. A. Attacking a network with adversarial examples

Goal: Given a network pretrained on ImageNet, find an input image that is not a iguana but will be classified as an iguana.



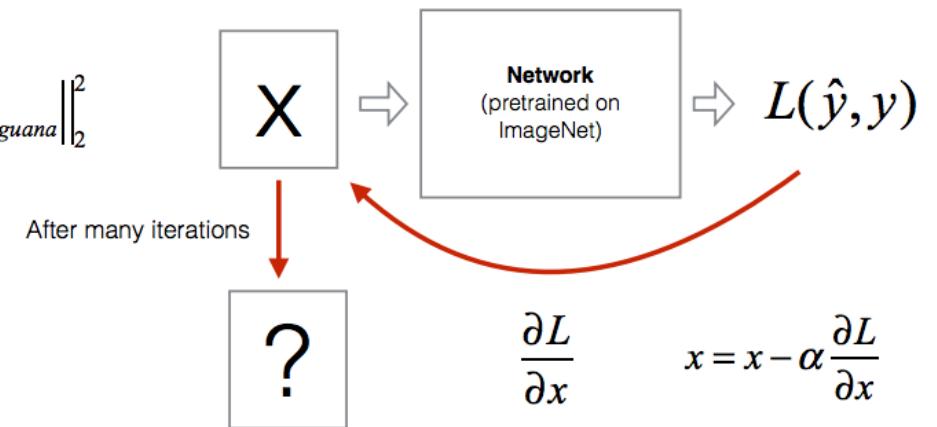
1. Rephrasing what we want:

$$\text{Find } x \text{ such that: } \hat{y}(x) = y_{\text{iguana}} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

2. Defining the loss function

$$L(\hat{y}, y) = \frac{1}{2} \|\hat{y}(W, b, x) - y_{\text{iguana}}\|_2^2$$

3. Optimize the image

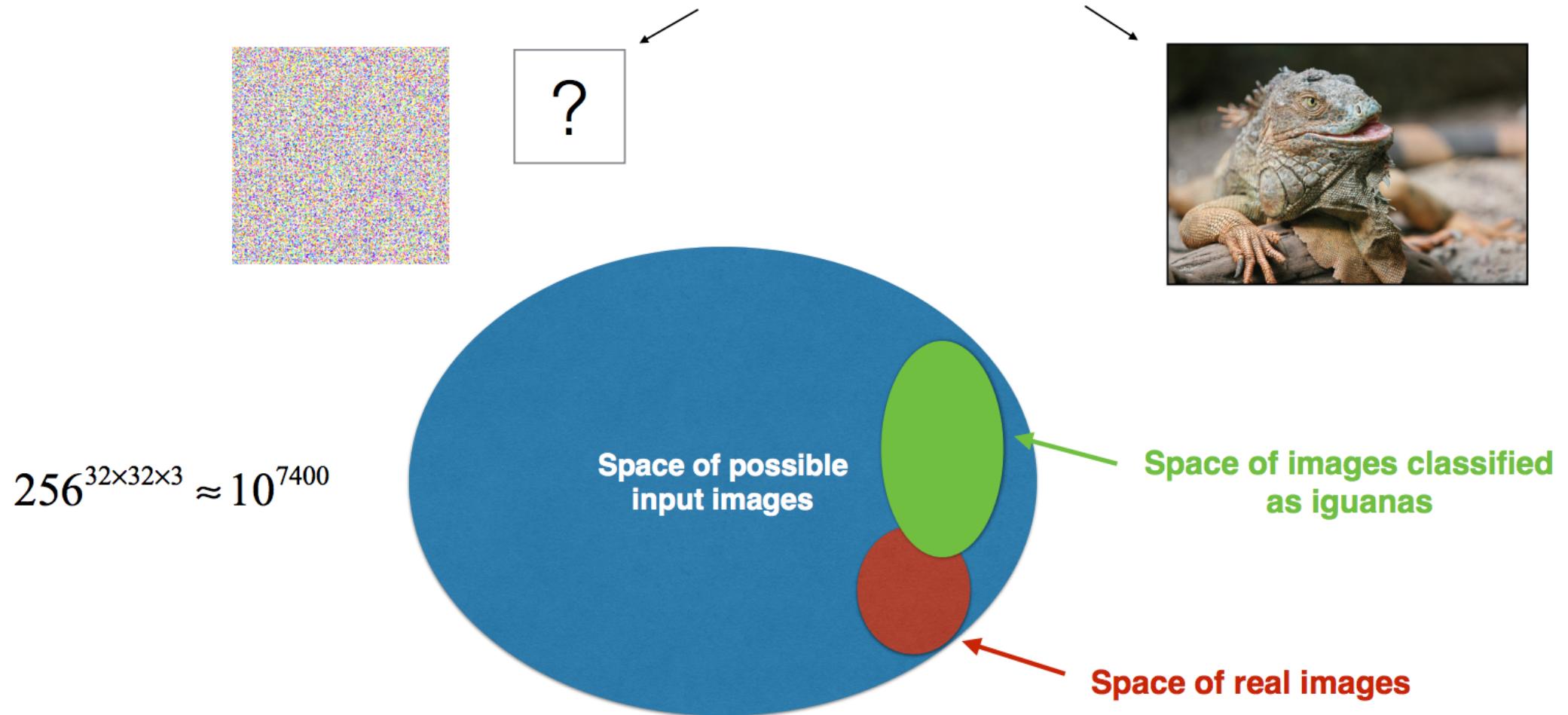


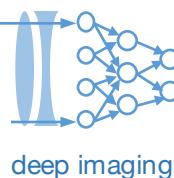
[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

I. A. Attacking a network with adversarial examples

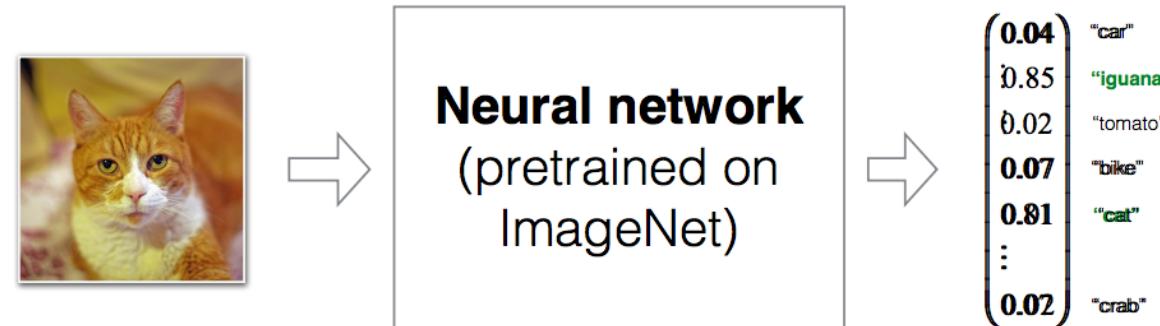
Question: Will the forged image \mathbf{x} look like an **iguana**?





I. A. Attacking a network with adversarial examples

Goal: Given a network pretrained on ImageNet, find an input image that is a cat but will be classified as an iguana.



1. Rephrasing what we want:

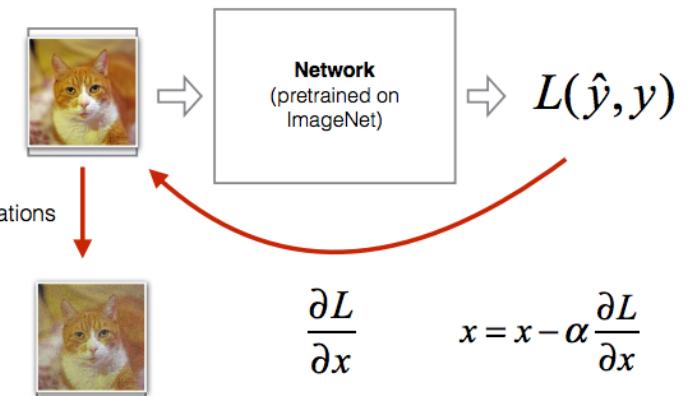
$$\text{Find } x \text{ such that: } \hat{y}(x) = y_{\text{iguana}} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\text{And: } x = x_{\text{cat}}$$

2. Defining the loss function

$$L(\hat{y}, y) = \frac{1}{2} \|\hat{y}(W, b, x) - y_{\text{iguana}}\|_2^2 + \lambda \|x - x_{\text{cat}}\|_2^2$$

3. Optimize the image

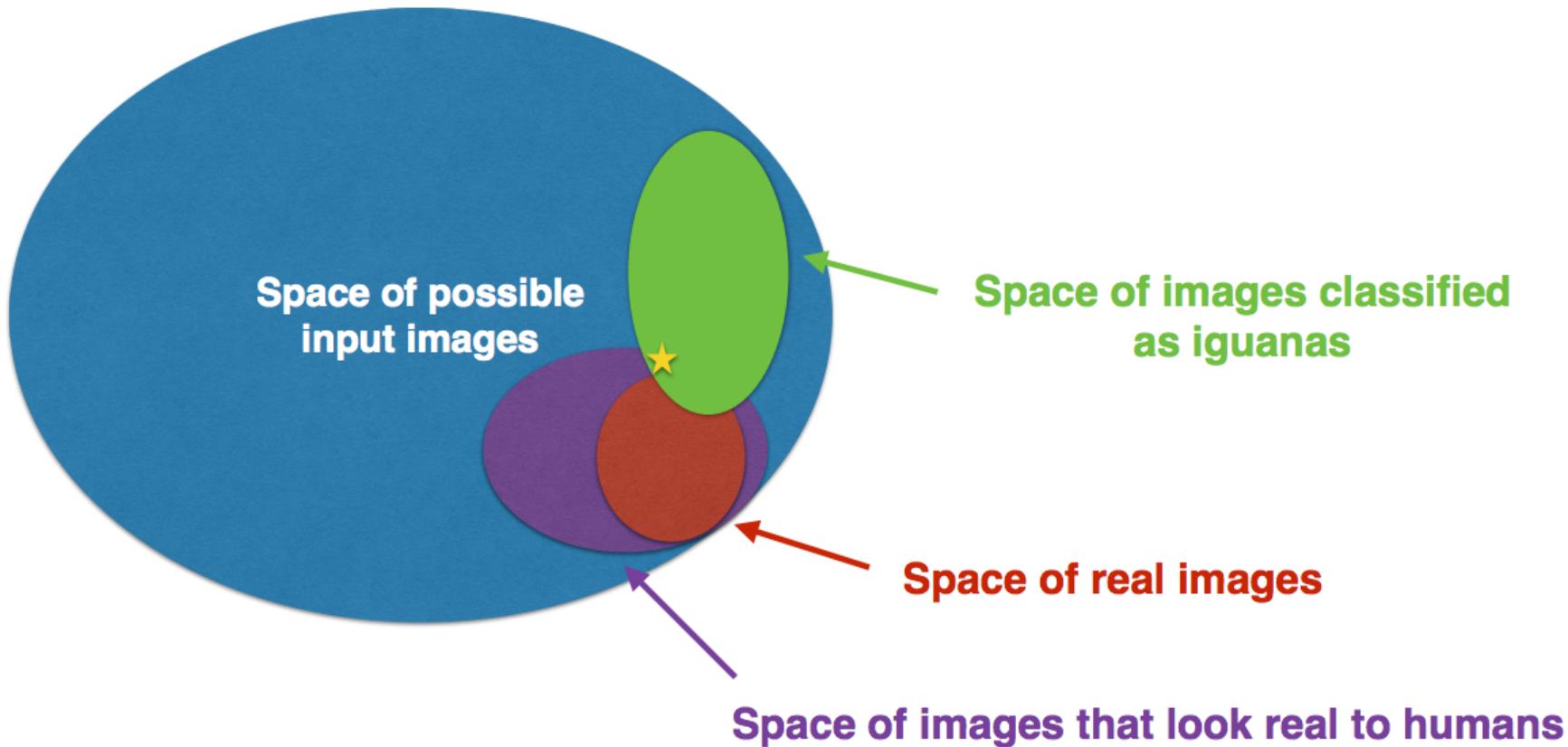


[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

I. A. Attacking a network with adversarial examples

$$256^{32 \times 32 \times 3} \approx 10^{7400}$$



I. B. Defenses against adversarial examples

Menti

Knowledge of the attacker:

- White-box
- Black-box

Solution 1

- Create a SafetyNet

Solution 2

- Train on correctly labelled adversarial examples

Solution 3

- Adversarial training $L_{new} = L(W, b, x, y) + \lambda L(W, b, x_{adv}, y)$

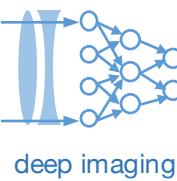
- Adversarial logit pairing $L_{new} = L(W, b, x, y) + \lambda \|f(x; W, b) - f(x_{adv}; W, b)\|_2^2$

$x =$ 
 $y = \text{cat}$

[Lu et al. (2017): SafetyNet: Detecting and Rejecting Adversarial Examples Robustly]

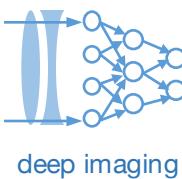
[Harini Kannan et al. (2018): Adversarial Logit Pairing]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri



NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles

Jiajun Lu*, Hussein Sibai*, Evan Fabry, David Forsyth
University of Illinois at Urbana Champaign
{jlu23, sibai2, efabry2, daf}@illinois.edu

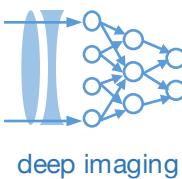


It has been shown that most machine learning algorithms are susceptible to adversarial perturbations. Slightly perturbing an image in a carefully chosen direction in the image space may cause a trained neural network model to misclassify it. Recently, it was shown that physical adversarial examples exist: printing perturbed images then taking pictures of them would still result in misclassification. This raises security and safety concerns.

Original hypothesis: "Do adversarial examples exist?"

To prove true: need just one example

To prove false: seems challenging... (do unicorns exist?)



However, these experiments ignore a crucial property of physical objects: the camera can view objects from different distances and at different angles. In this paper, we show experiments that suggest that current constructions of physical adversarial examples do not disrupt object detection from a moving platform. Instead, a trained neural network classifies most of the pictures taken from different distances and angles of a perturbed image correctly. We believe this is because the adversarial property of the perturbation is sensitive to the scale at which the perturbed picture is viewed, so (for example) an autonomous car will misclassify a stop sign only from a small range of distances.

New hypothesis: "Are adversarial examples robust?"

To prove true: need just one example implementation

To prove false: Need to show *all possible* implementations fail

4 different adversarial examples for object detector:



4 different adversarial examples for object classifier:



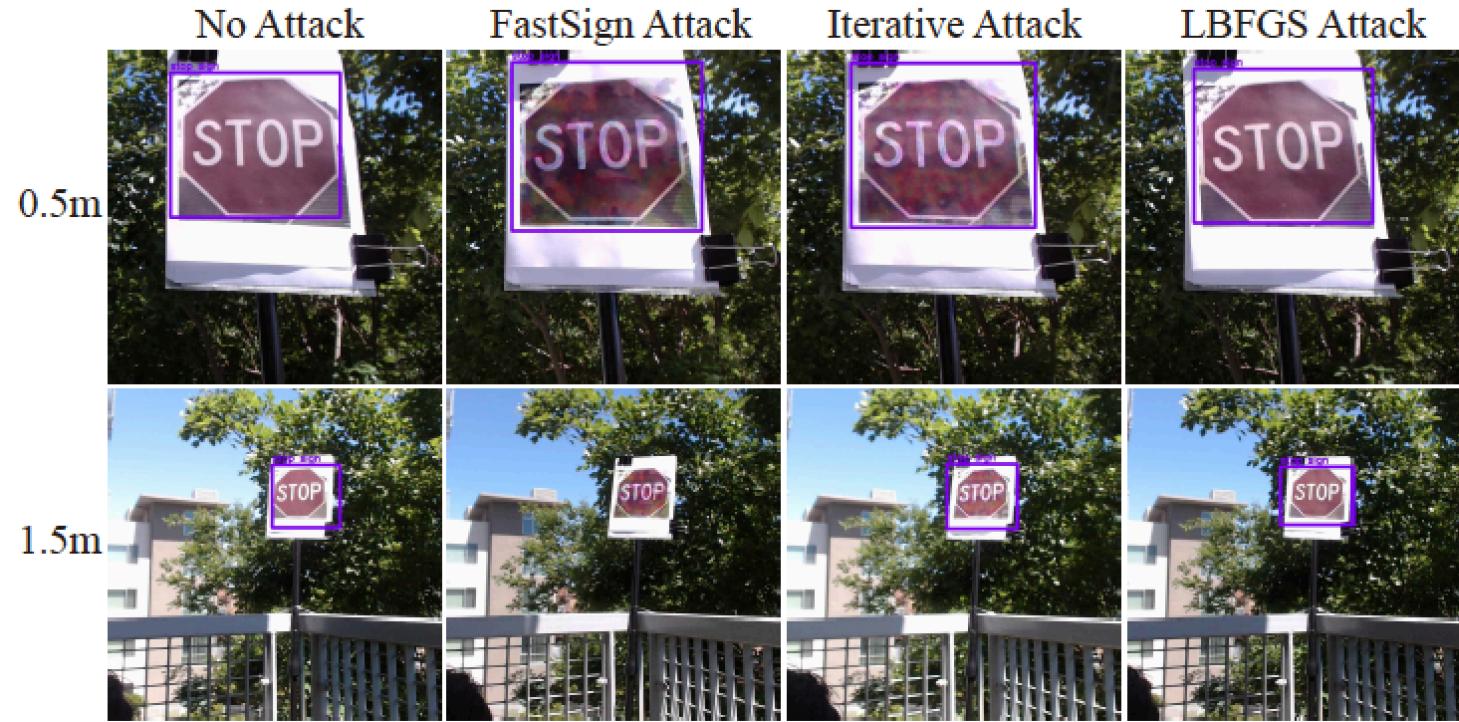
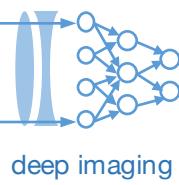
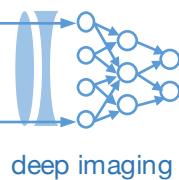


Figure 3: This figure shows experiment setup, and we use the printed stop signs to simulate real stop signs with natural background. These are examples for successful 0.5 meters and 1.5 meters detection: both original images and adversarial examples are detected in both distances. It demonstrates that adversarial examples in a physical setting do not reliably fool stop sign detectors.



Within 5 days (!), a blog post from OpenAI:

<https://blog.openai.com/robust-adversarial-inputs/>

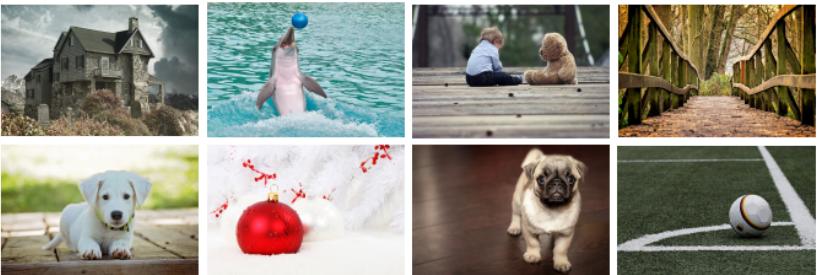


Generative adversarial networks

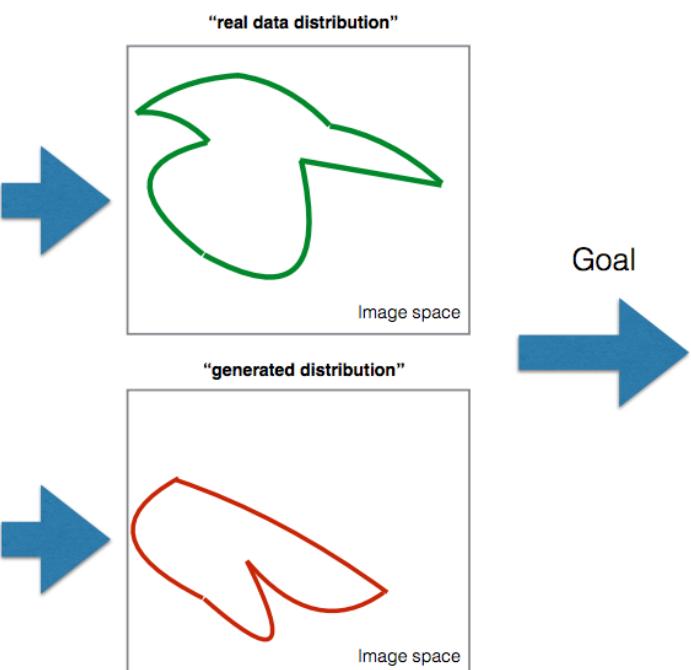
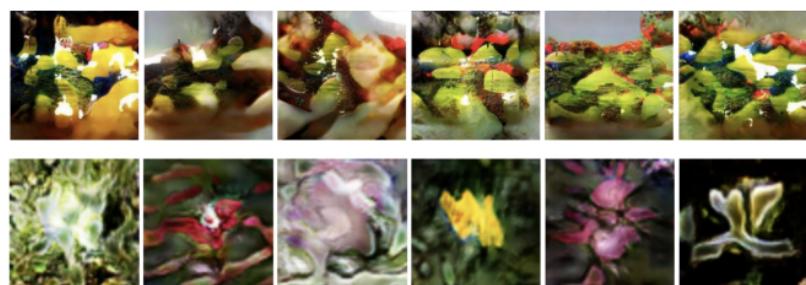
II.A - Motivation

Probability distributions:

Samples from the “real data distribution”

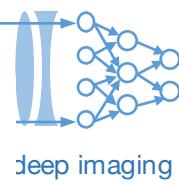


Samples from the “generated distribution”

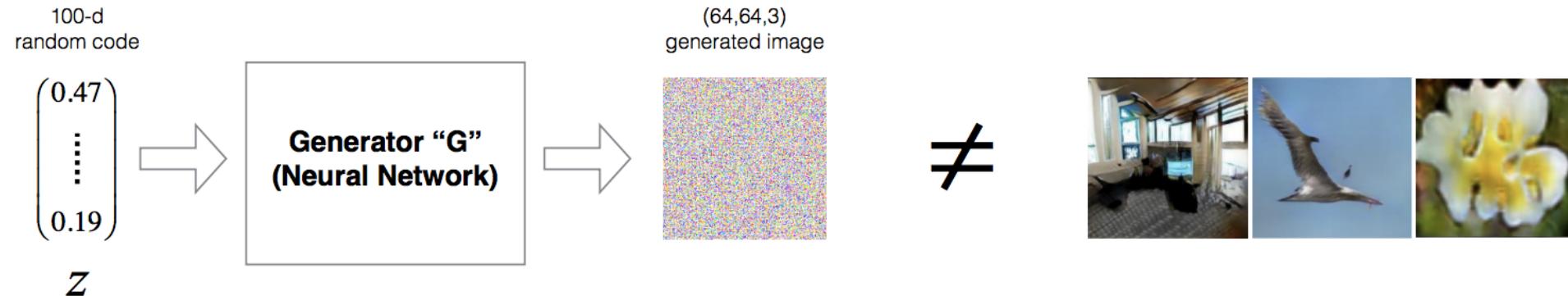


[Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas (2017): StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks]

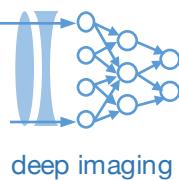
Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri



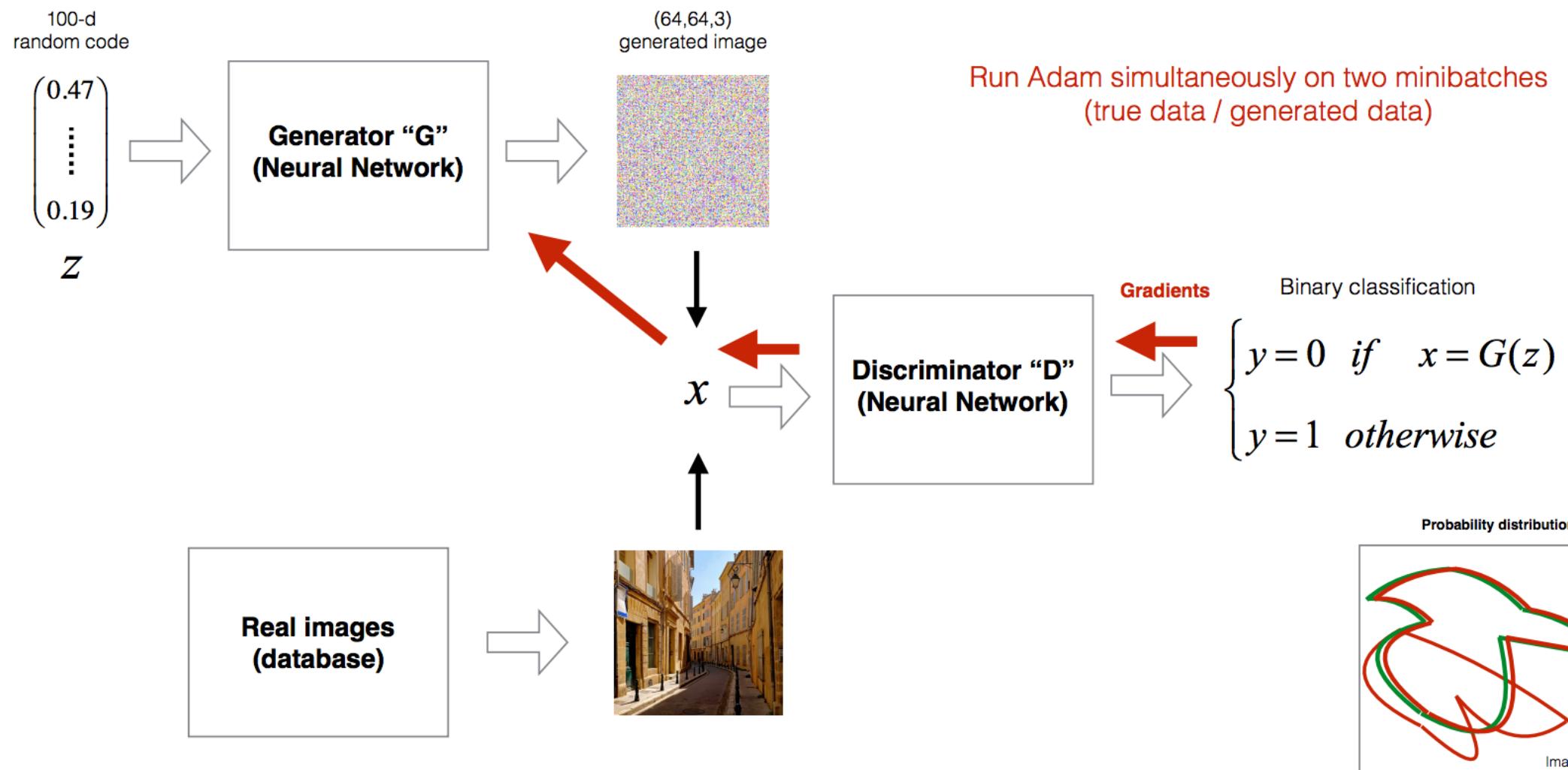
II.B - G/D Game



How can we train G to generate images from the true data distributions?

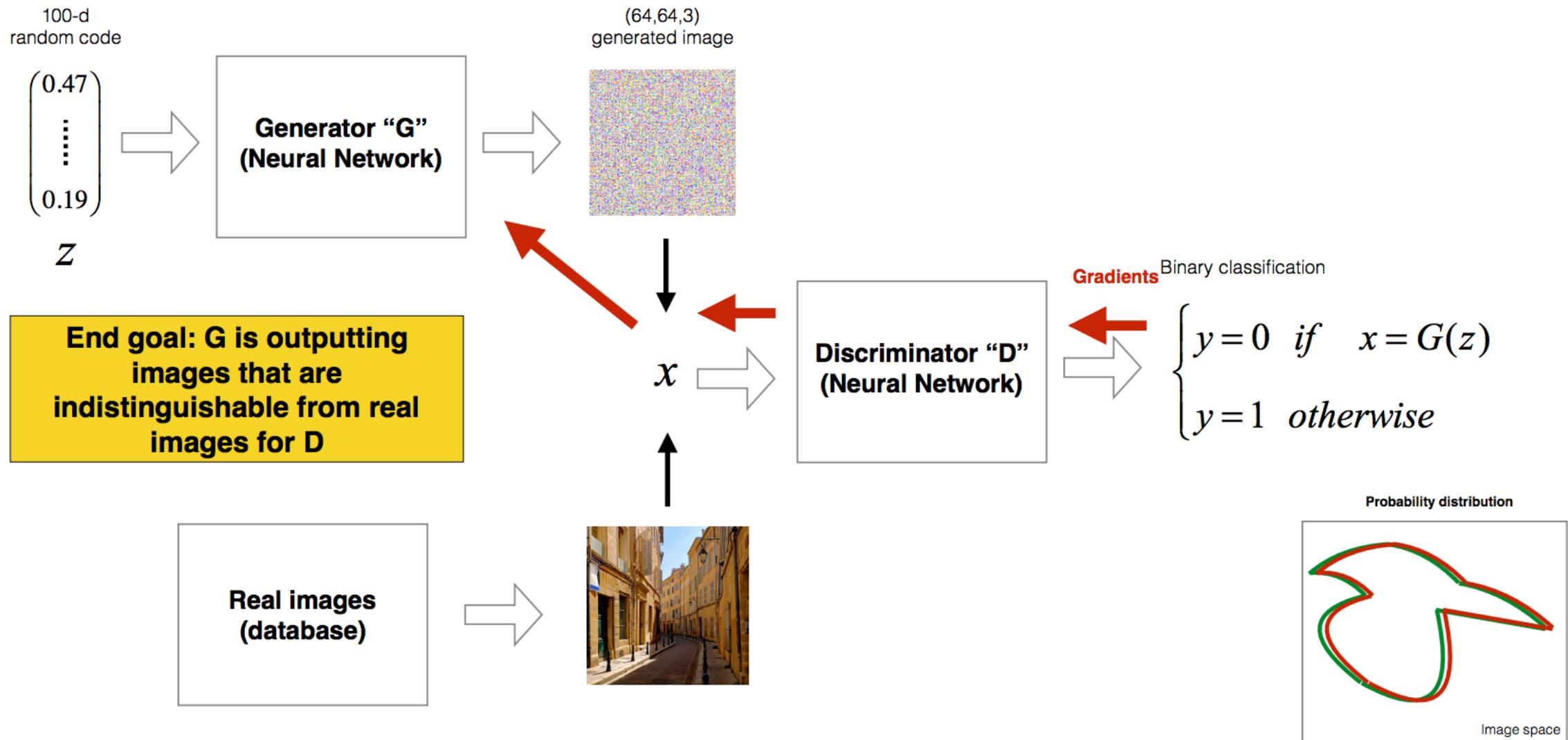


II.B - G/D Game



Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

II.B - G/D Game



Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

Training procedure, we want to minimize:

Labels: $\begin{cases} y_{real} \text{ is always 1} \\ y_{gen} \text{ is always 0} \end{cases}$

- The cost of the discriminator

$$J^{(D)} = \underbrace{-\frac{1}{m_{real}} \sum_{i=1}^{m_{real}} y_{real}^{(i)} \cdot \log(D(x^{(i)}))}_{\text{cross-entropy 1: } "D \text{ should correctly label real data as 1}"} - \underbrace{\frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} (1 - y_{gen}^{(i)}) \cdot \log(1 - D(G(z^{(i)})))}_{\text{cross-entropy 2: } "D \text{ should correctly label generated data as 0}"}$$

- The cost of the generator

$$J^{(G)} = -J^{(D)} = \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log(1 - D(G(z^{(i)})))$$

"G should try to fool D: by minimizing the opposite of what D is trying to minimize"

"Maximize probability that the discriminator is wrong and labels the fake example as a real example"

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri



II. D. In terms of code

```
# Build and compile the discriminator
self.discriminator = self.build_discriminator()
self.discriminator.compile(loss='binary_crossentropy',
    optimizer=optimizer,
    metrics=['accuracy'])

# Build the generator
self.generator = self.build_generator()

# The generator takes noise as input and generates imgs
z = Input(shape=(self.latent_dim,))
img = self.generator(z)

# For the combined model we will only train the generator
self.discriminator.trainable = False

# The discriminator takes generated images as input and determines validity
validity = self.discriminator(img)

# The combined model (stacked generator and discriminator)
# Trains the generator to fool the discriminator
self.combined = Model(z, validity)
self.combined.compile(loss='binary_crossentropy', optimizer=optimizer)
```

```
def build_discriminator():

    model = Sequential()

    model.add(Flatten(input_shape=self.img_shape))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation='sigmoid'))

    model.summary()

    img = Input(shape=self.img_shape)
    validity = model(img)

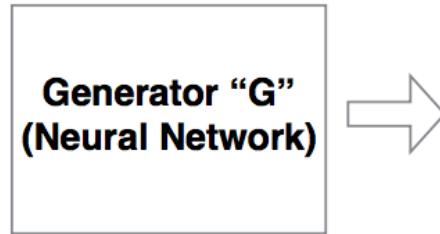
    return Model(img, validity)
```

II.E - Nice results

Operation on codes

Code 1

$$\begin{pmatrix} 0.12 \\ \vdots \\ 0.92 \end{pmatrix}$$

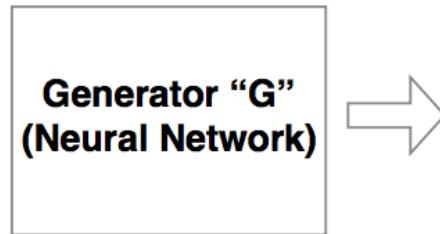


(64,64,3)
generated image



Code 2

$$\begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix}$$

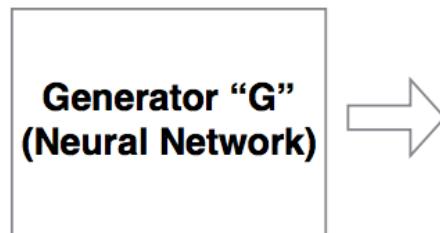


(64,64,3)
generated image



Code 3

$$\begin{pmatrix} 0.42 \\ \vdots \\ 0.07 \end{pmatrix}$$

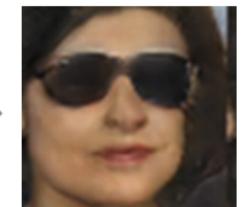
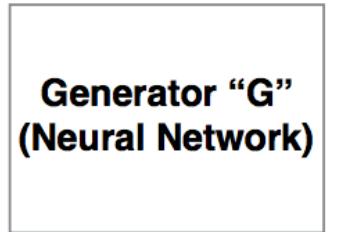


(64,64,3)
generated image

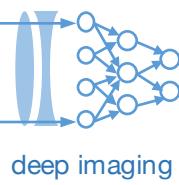


Code 1 Code 2 Code 3

$$\begin{pmatrix} 0.12 \\ \vdots \\ 0.92 \end{pmatrix} - \begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix} + \begin{pmatrix} 0.42 \\ \vdots \\ 0.07 \end{pmatrix}$$

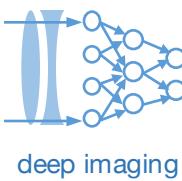


Man with glasses - man + woman = woman with glasses



deep imaging

[https://colab.research.google.com/github/tensorflow/gan/blob
/master/tensorflow_gan/examples/colab_notebooks/tfgan_tut
orial.ipynb?utm_source=ss-gan&utm_campaign=colab-
external&utm_medium=referral&utm_content=tfgan-intro](https://colab.research.google.com/github/tensorflow/gan/blob/master/tensorflow_gan/examples/colab_notebooks/tfgan_tutorial.ipynb?utm_source=ss-gan&utm_campaign=colab-external&utm_medium=referral&utm_content=tfgan-intro)



Take-aways for the future of machine learning and imaging

1. It's not going away....it works, there's a big community, and lots of \$
2. Hardware *and* software are rapidly evolving
3. CNN's work very well, but they are not the final solution...
4. There is currently a lack of safeguards and not enough consideration for how to ensure processed results are accurate, secure and trustworthy
5. Things are likely going to get quite complicated...