Lecture 22: Looking ahead – machine learning and imaging in 10 years

deep imaging

Machine Learning and Imaging

BME 590L
Roarke Horstmeyer

Announcements and schedule

- Last Lecture: Thursday 4/16, course review
- TA's will hold labs on M/W next week
- Homework #5 Due: Tuesday April 21 (1 week)
- Then final projects will be due (Friday 24 – Wednesday 29)
- Project help:
  - I will continue my office hours
    - Wednesday and Thursday, 10am – 11am
  - Email me if you'd like to meet another time
  - Email TA's / reach out on Slack to meet them as well – I think they might setup some additional times to help out

Components of final project

40% of total grade

1. Presentation Slides – 10%
   - 8-10 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

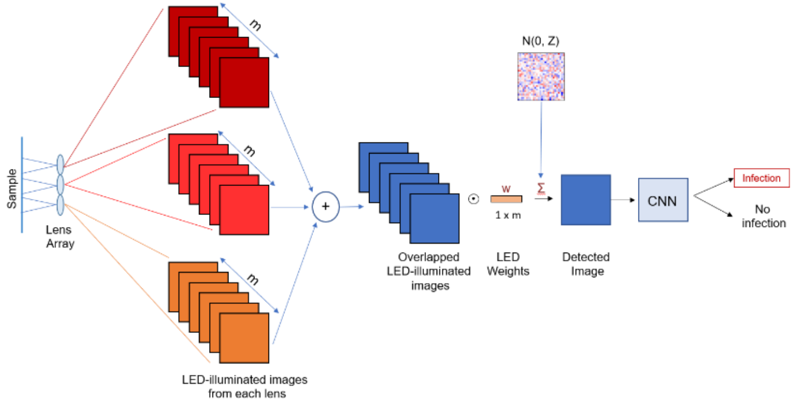
- Will also send permission form, which must be submitted with final project as well

Optimizing illumination for overlapped classification

Amey 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:

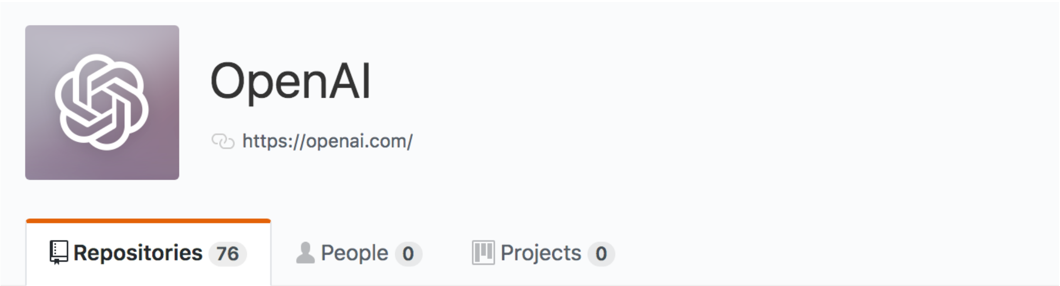
- You can provide a link to your code here: Code

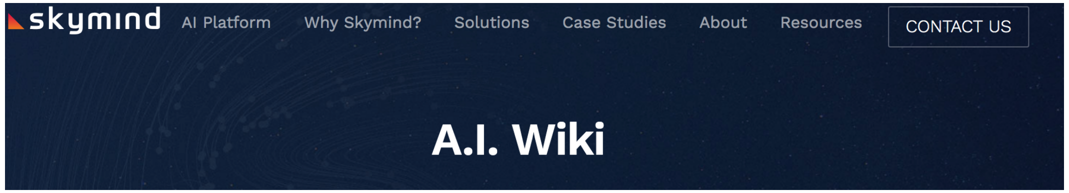
Where are things going with Machine Learning and Imaging in 10 years?

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

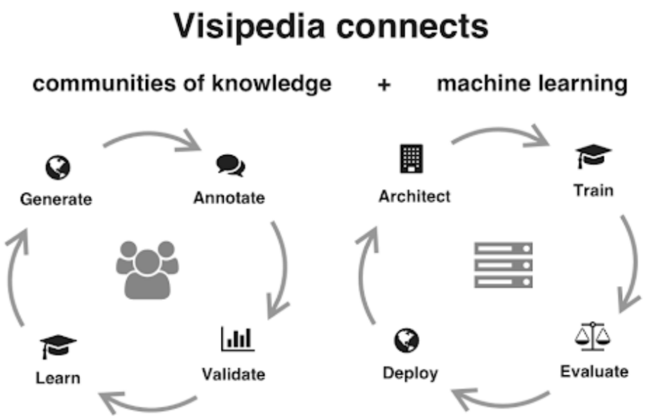
deep imaging

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





Visipedia



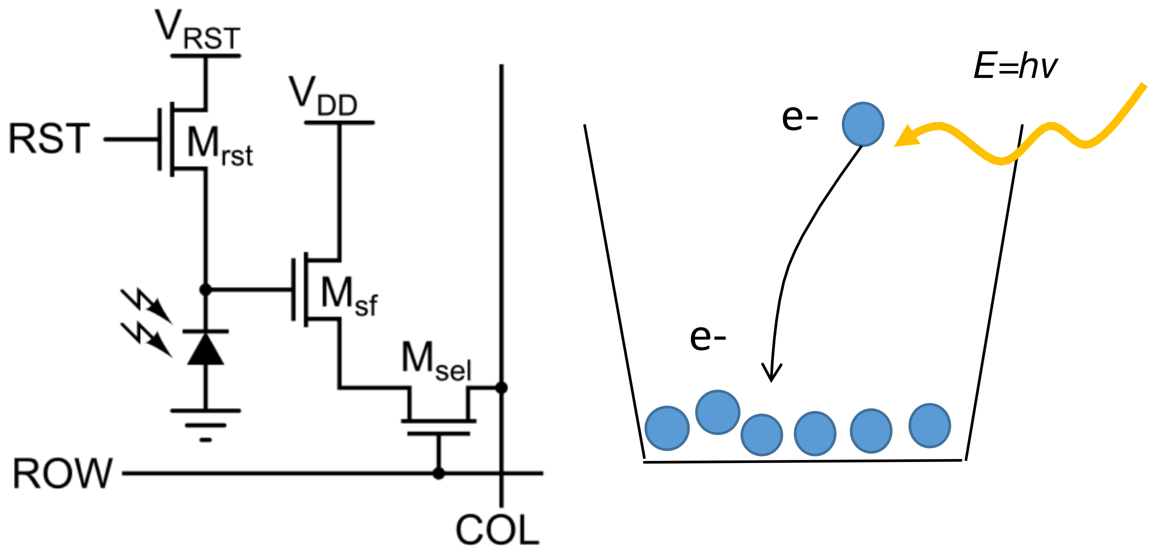
Caltech Visipedia

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

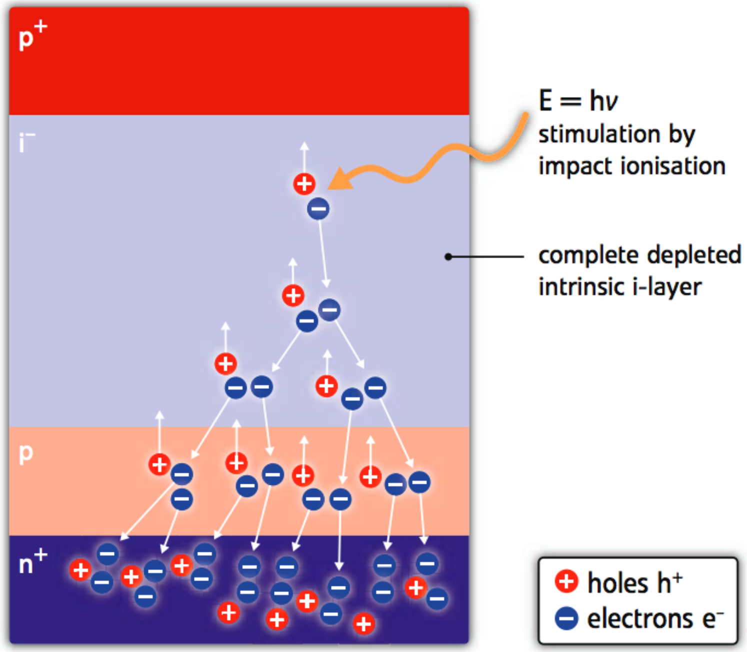
deep imaging

2. "Cameras" on many devices & new types of sensors

Standard CMOS pixel = bucket that collects electrons



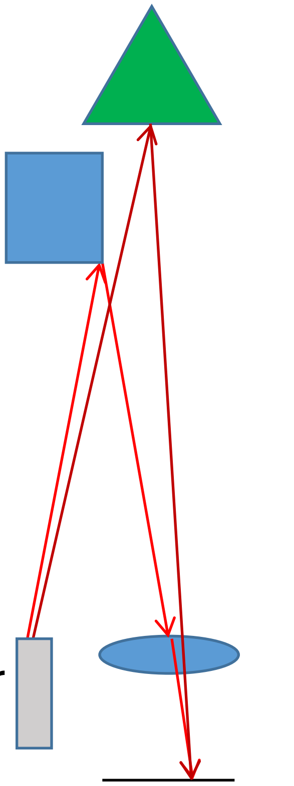
SPAD pixel: was there a photon or not?



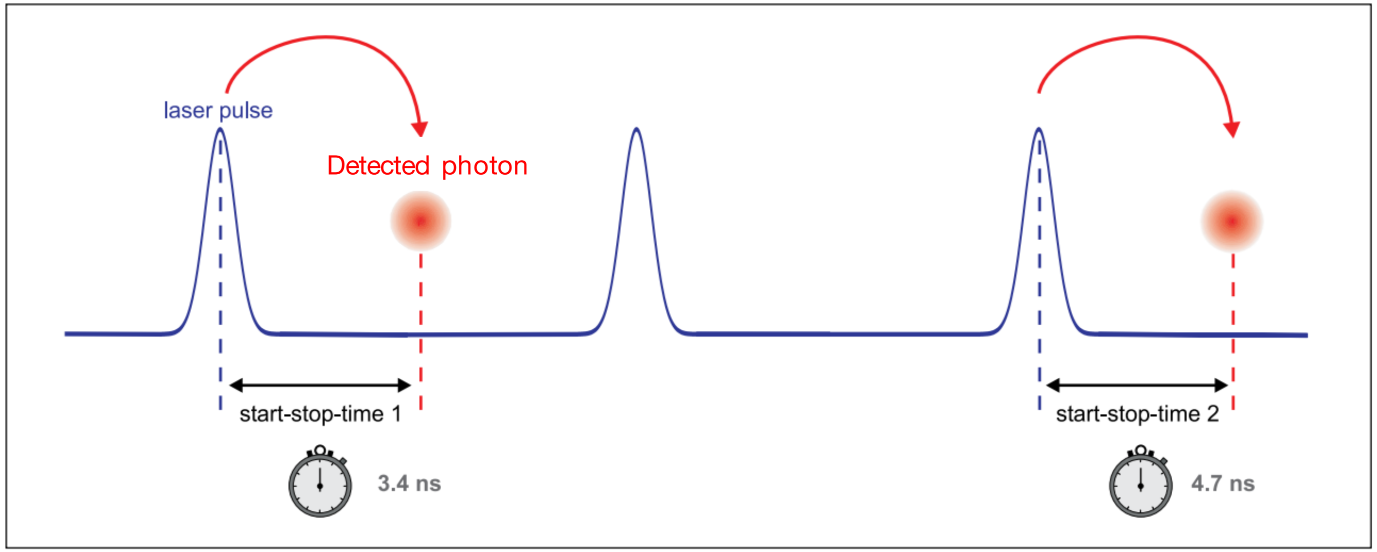
Where are things going with Machine Learning and Imaging in 10 years?

deep imaging

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)

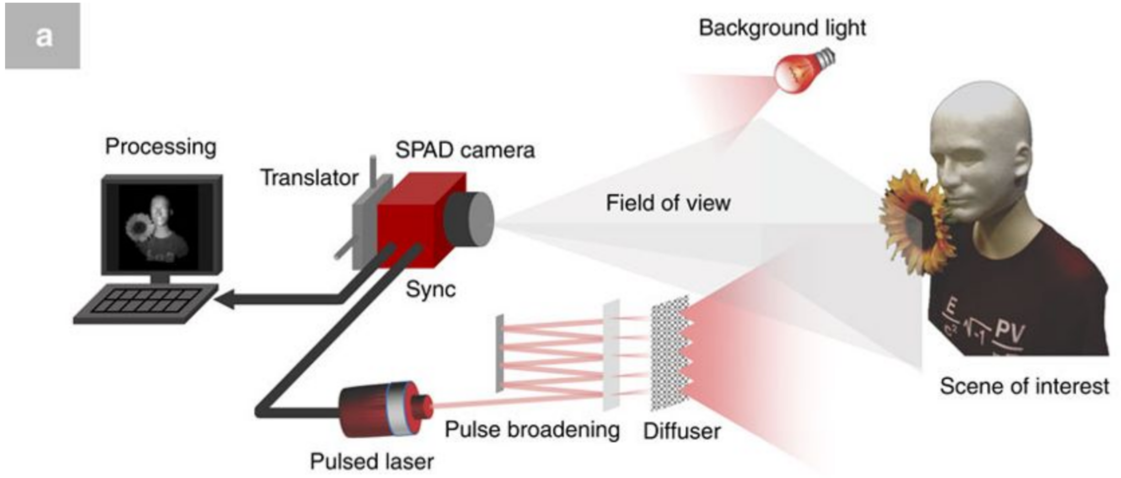


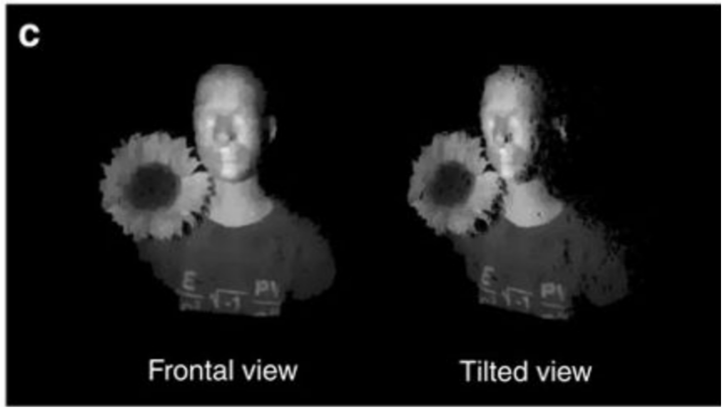
https://www.picoquant.com/images/uploads/page/files/7253/technote_tcspc.pdf

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

deep imaging

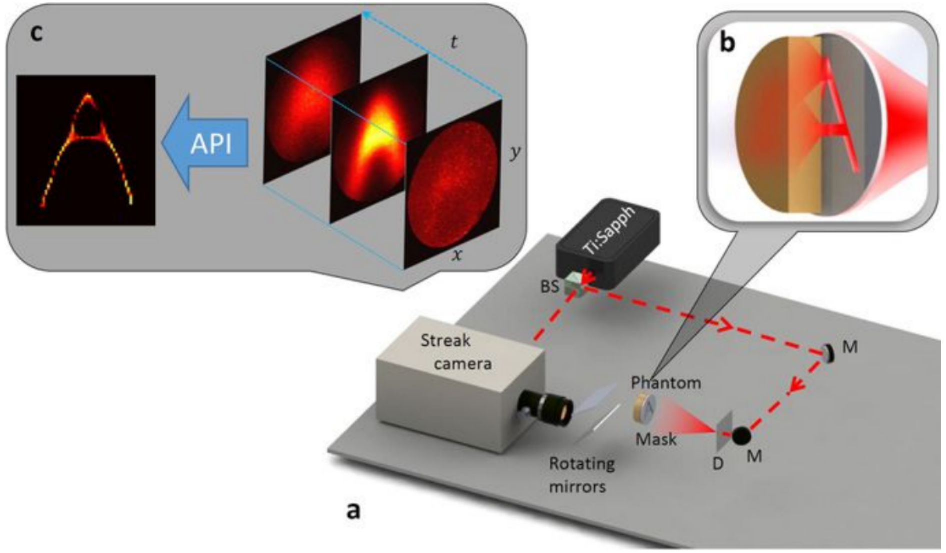
2. "Cameras" on many devices & new types of sensors





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

Figure 1: Imaging Through Thick Scattering.



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


LiDAR Scanner
LiDAR (Light Detection and Ranging) is used to determine distance by measuring how long it takes light to reach an object and reflect back. It is so advanced, it's being used by NASA for the next Mars landing mission. And it's now been engineered to fit in the thin and light iPad Pro.
The custom-designed LiDAR Scanner uses direct time of flight to measure reflected light from up to five meters away, both indoors and out. It works at the photon level, operates at nanosecond speeds, and opens up tremendous possibilities for augmented reality and beyond.

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

deep imaging

3. Beyond convolutions - new constructs for deep networks

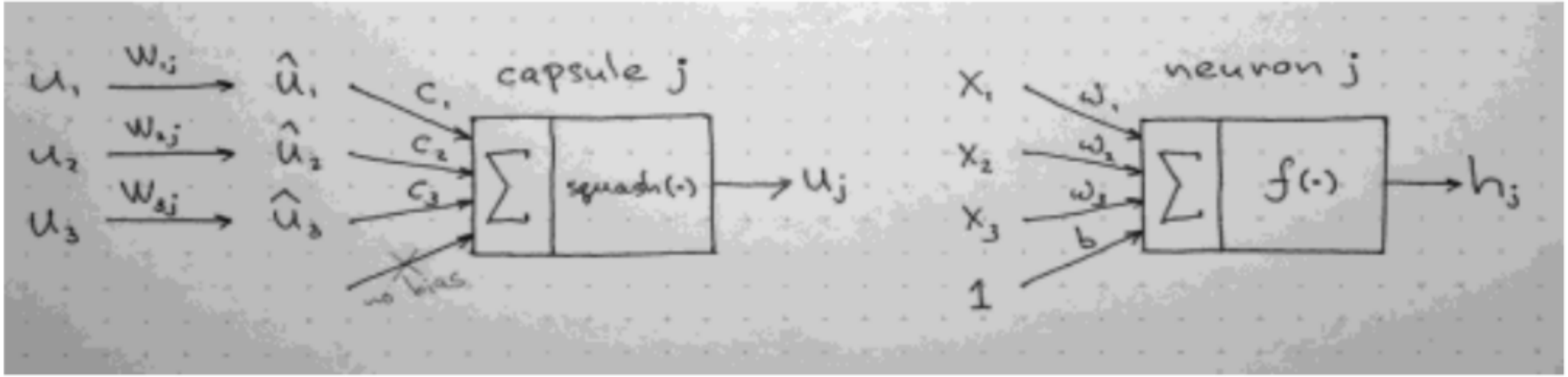
Where are things going with Machine Learning and Imaging in 10 years?

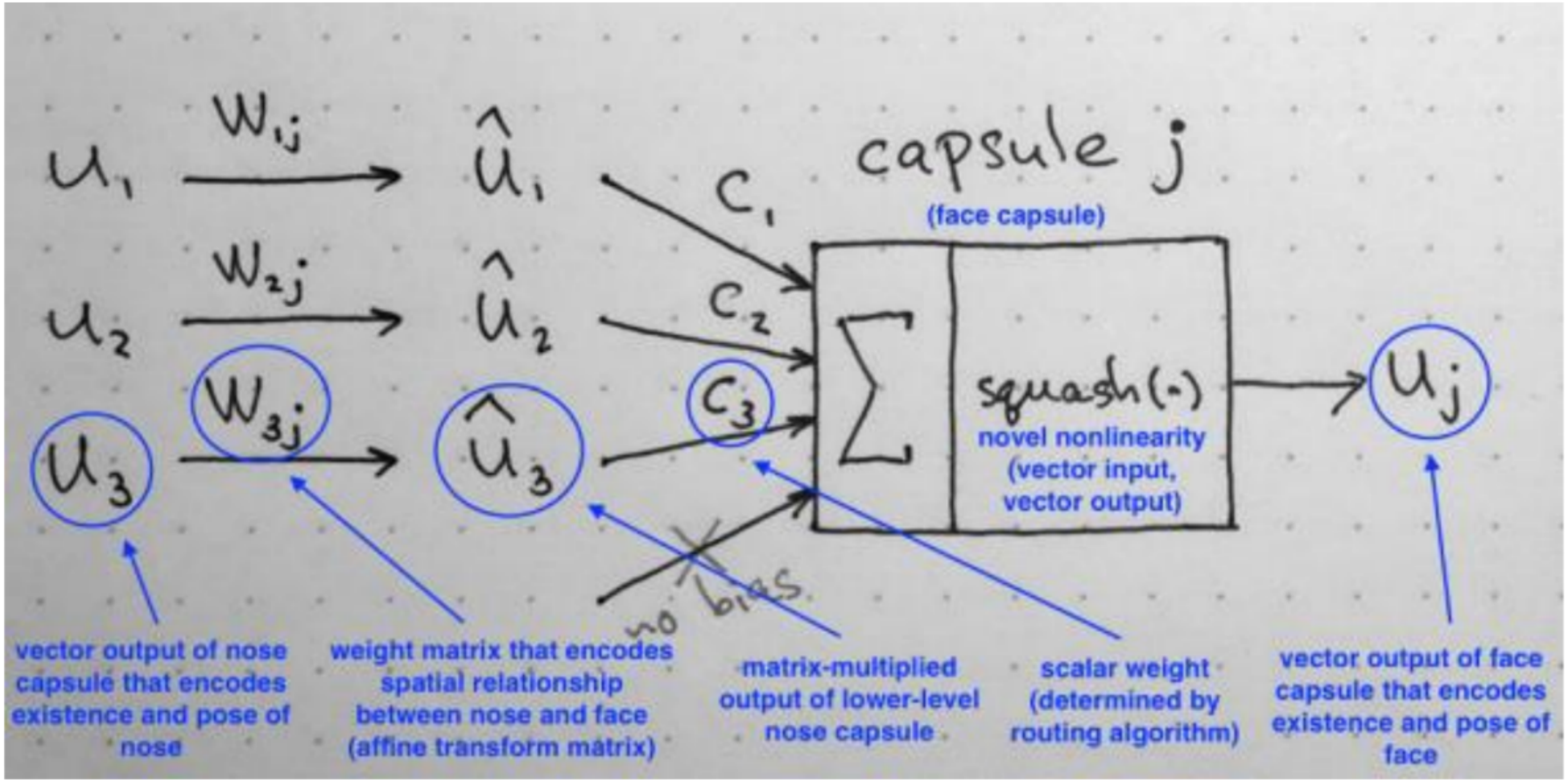
3. Beyond convolutions - new constructs for deep networks

Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com





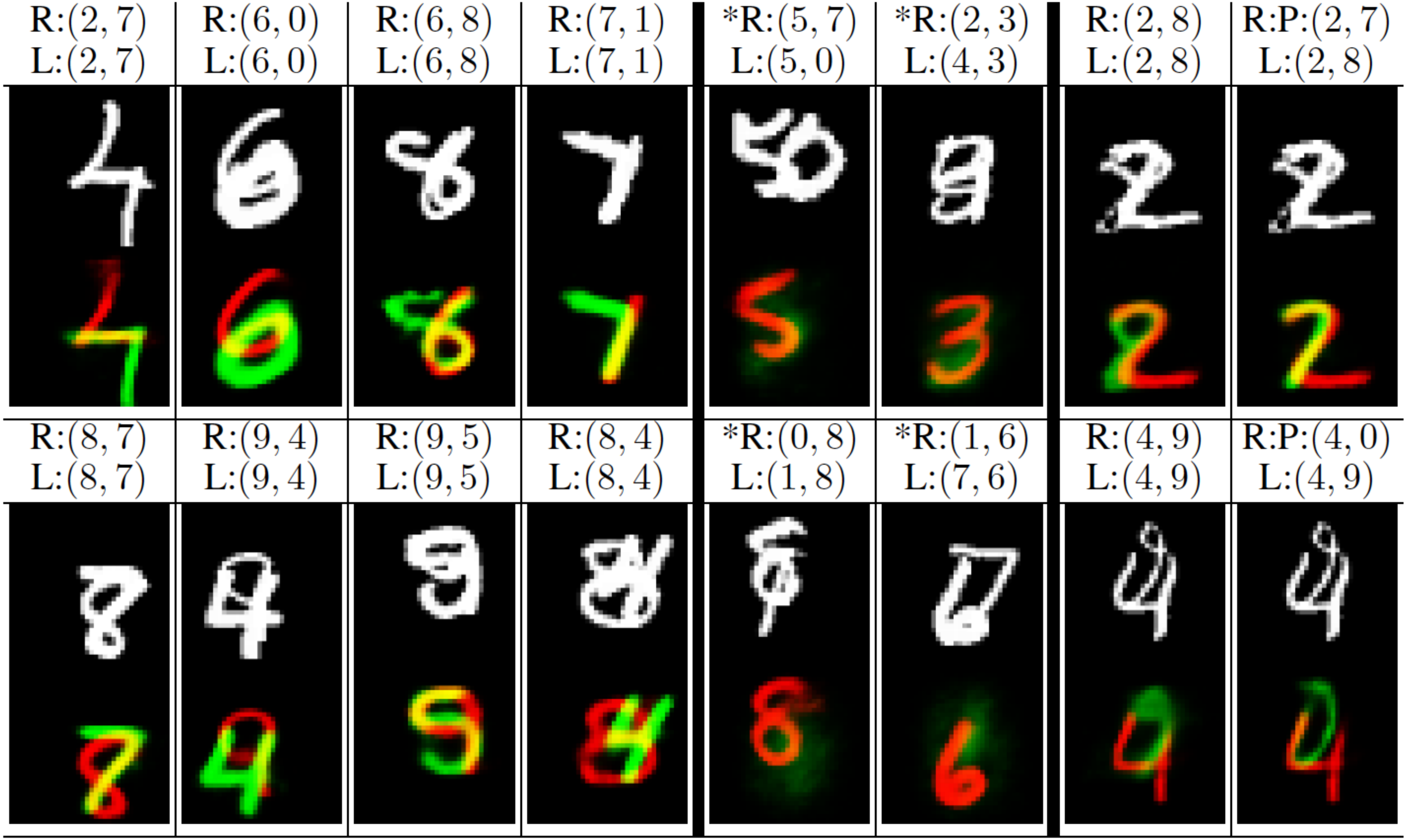
deep imaging

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	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\lVert\mathbf{s}_j\rVert^2}{1+\lVert\mathbf{s}_j\rVert^2}\frac{\mathbf{s}_j}{\lVert\mathbf{s}_j\rVert}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

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



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

deep imaging

4. Generative data is getting pretty realistic...

Example of Realistic Synthetic Photographs Generated with BigGANTaken from Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018.

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

Justus Thies [1] Michael Zollhöfer [2] Marc Stamminger [3] Christian Theobalt [2] Matthias Nießner [1]

[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 ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

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

Stanford CS231n, Lecture 12

deep imaging

Generative Models

Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

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

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

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

Stanford CS231n, Lecture 12

Taxonomy of Generative Models

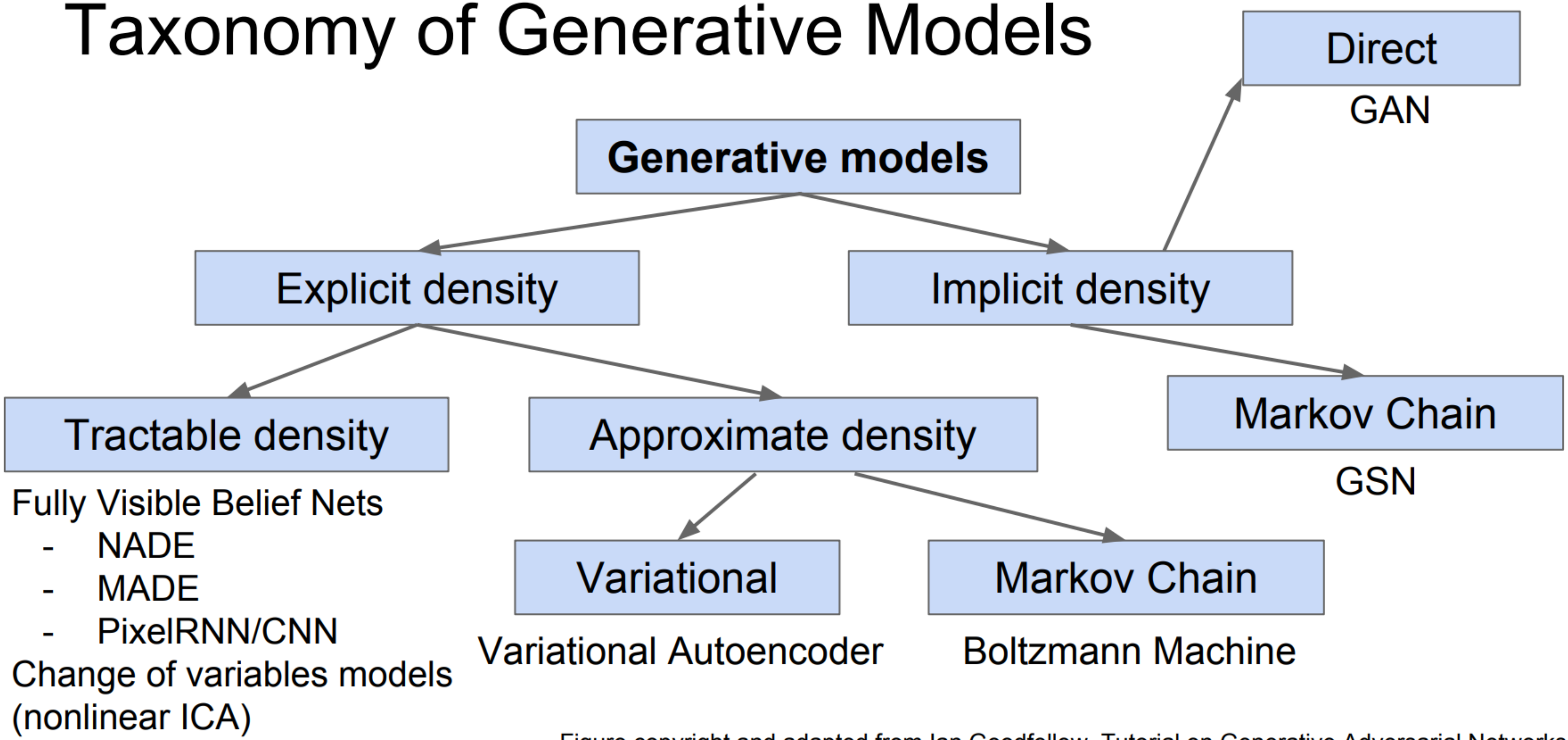


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Stanford CS231n, Lecture 12

Taxonomy of Generative Models

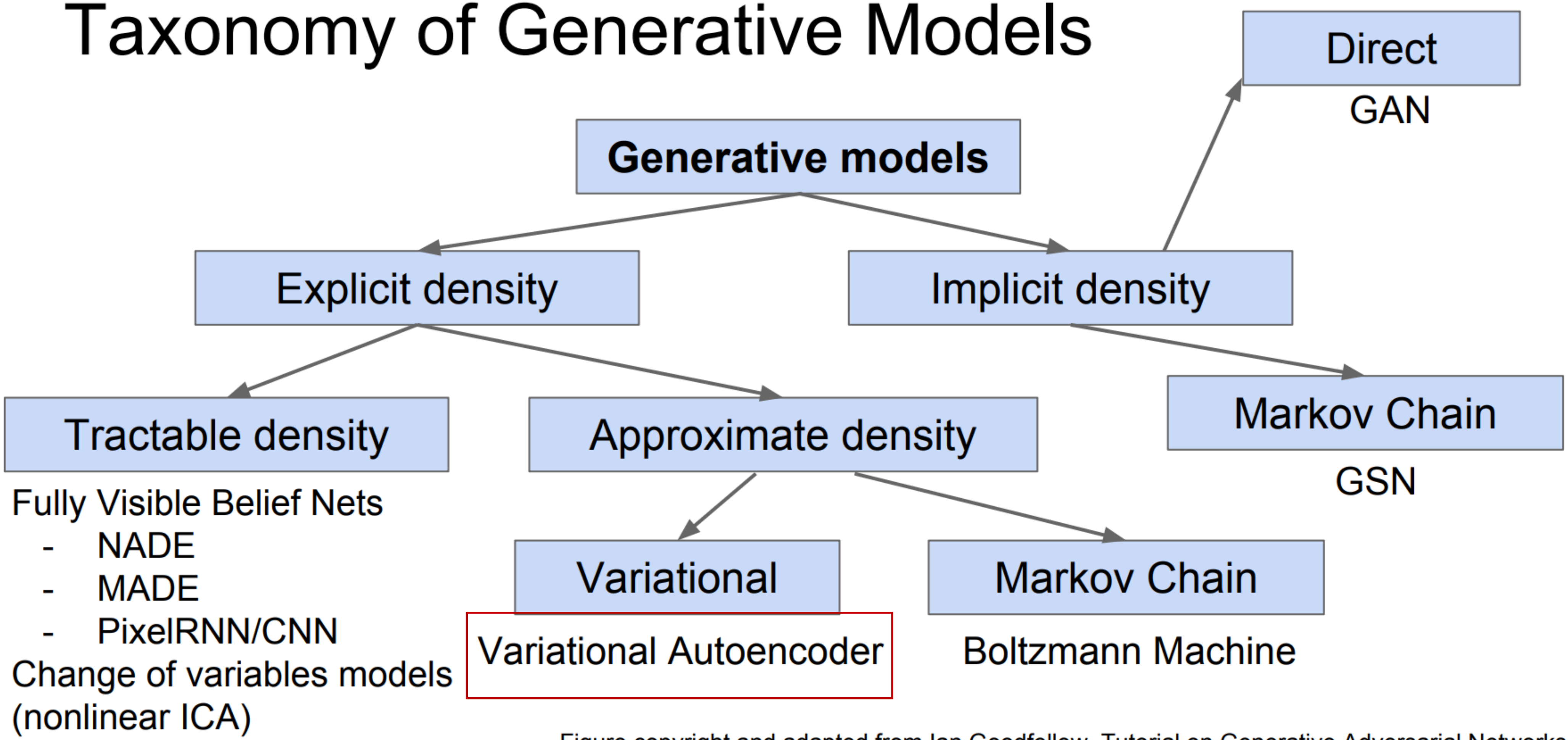
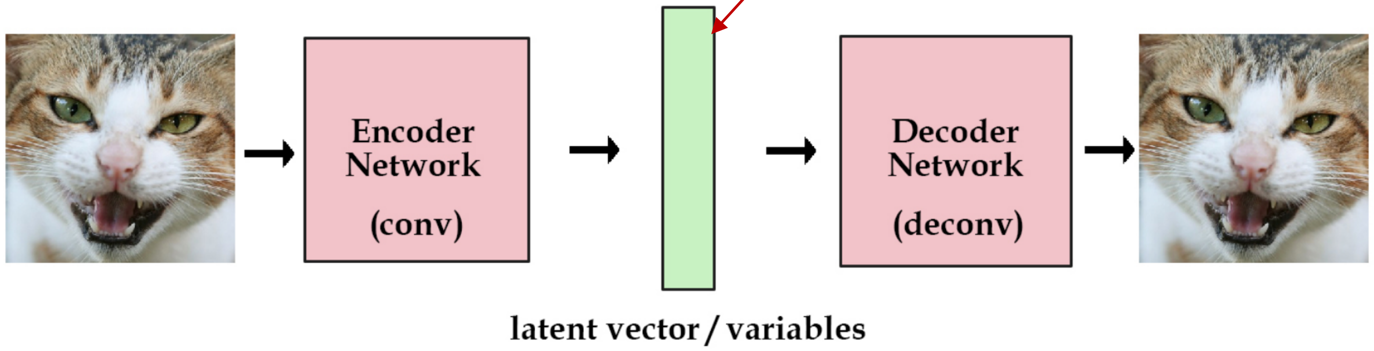


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Stanford CS231n, Lecture 12

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

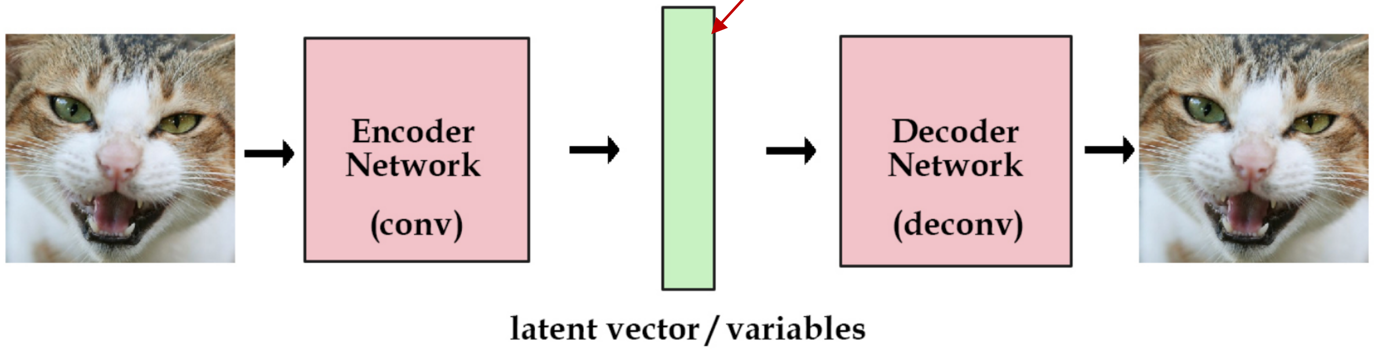


Minimize (KL) distance between latent vector and Gaussian normal

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

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

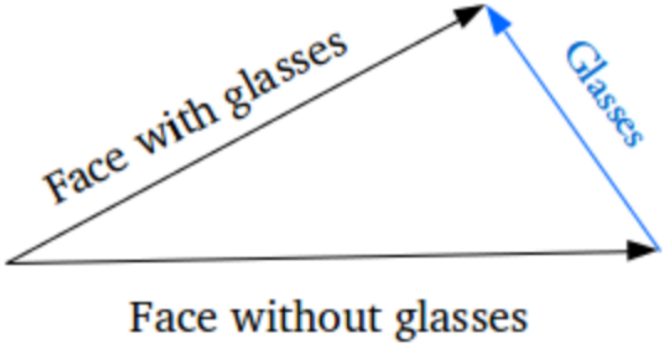


Minimize (KL) distance between latent vector and Gaussian normal

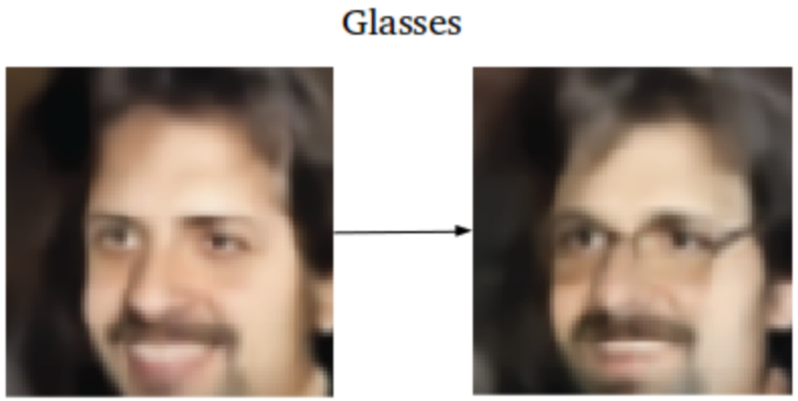
Generative Example (once trained):

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

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



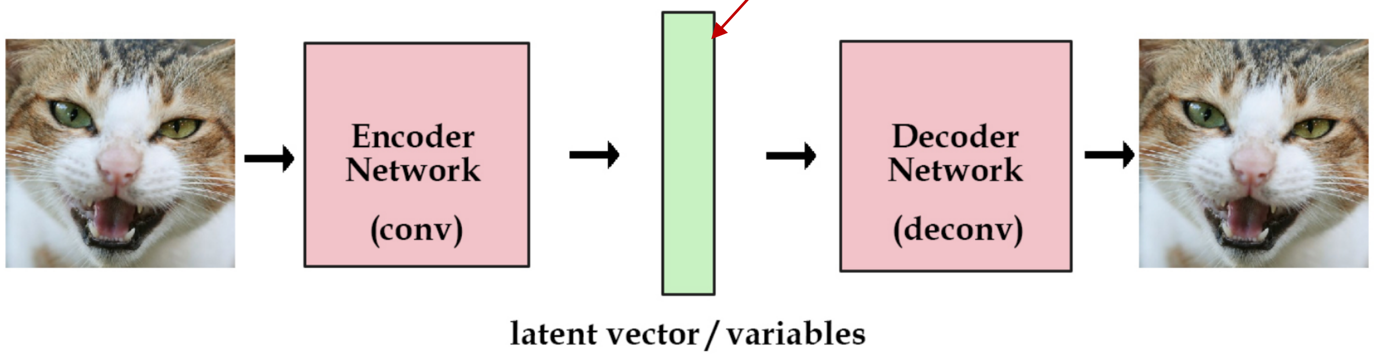
Adding new features to samples

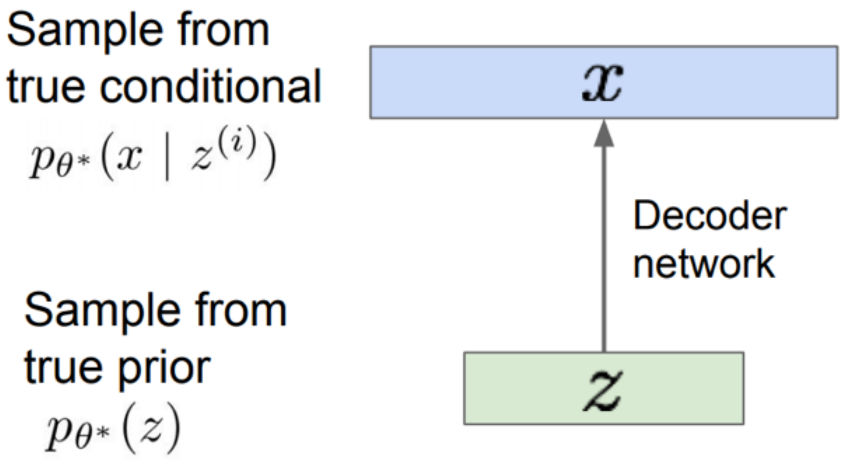


Exploring a specific variation of input data[1]

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF





Taxonomy of Generative Models

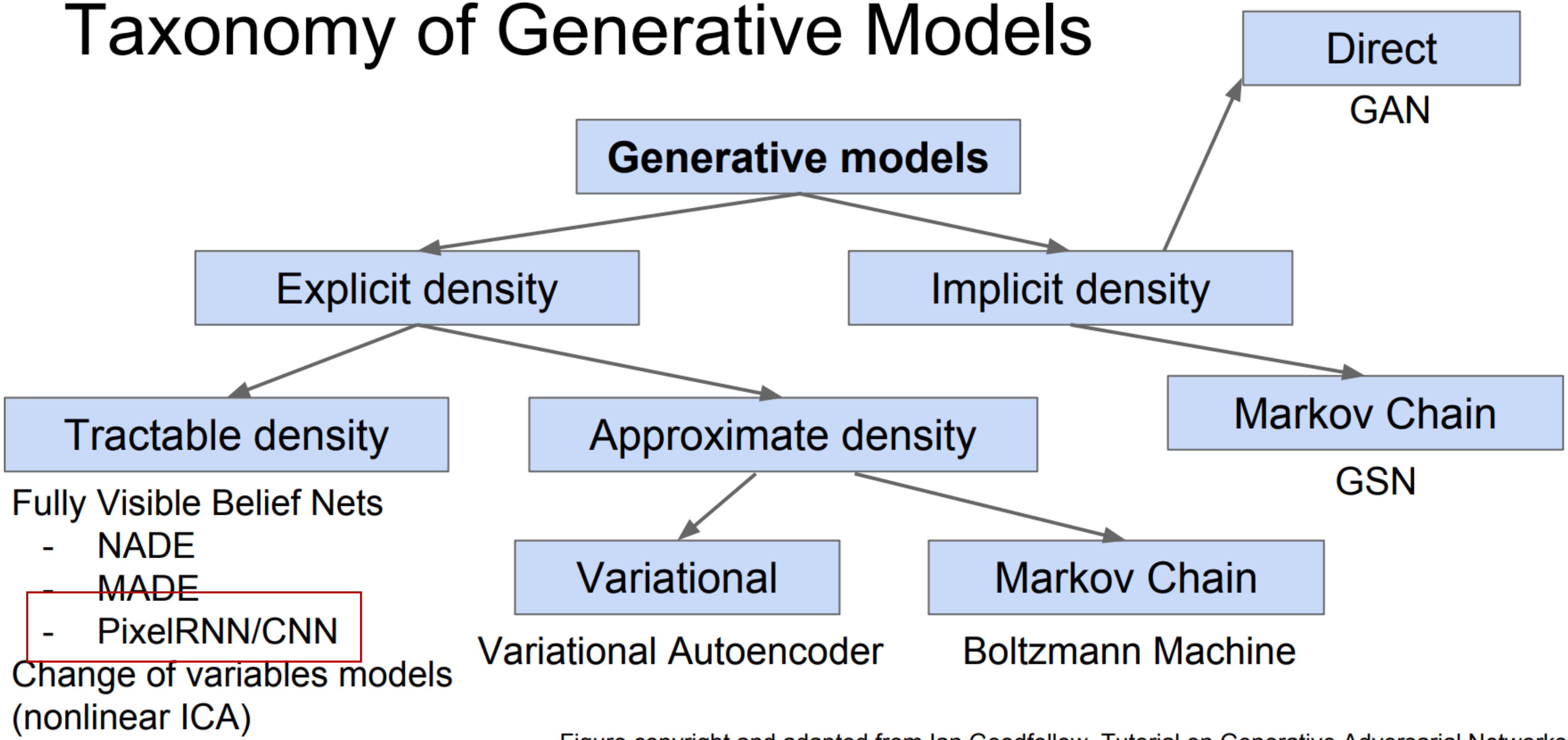


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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:

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

Likelihood of image x

Probability of i'th pixel value 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:

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

Likelihood of image x

Probability of i'th pixel value 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!

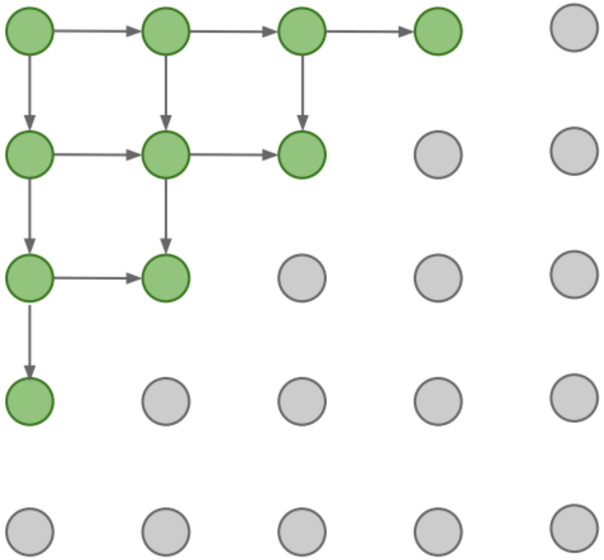


Figure 1. Image completions sampled from a PixelRNN.

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

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

Softmax loss at each pixel

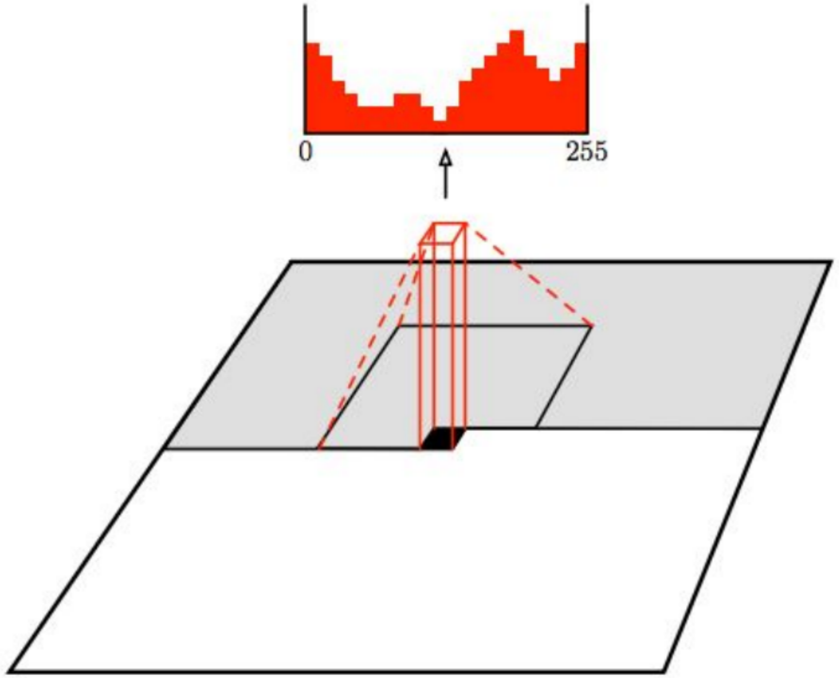


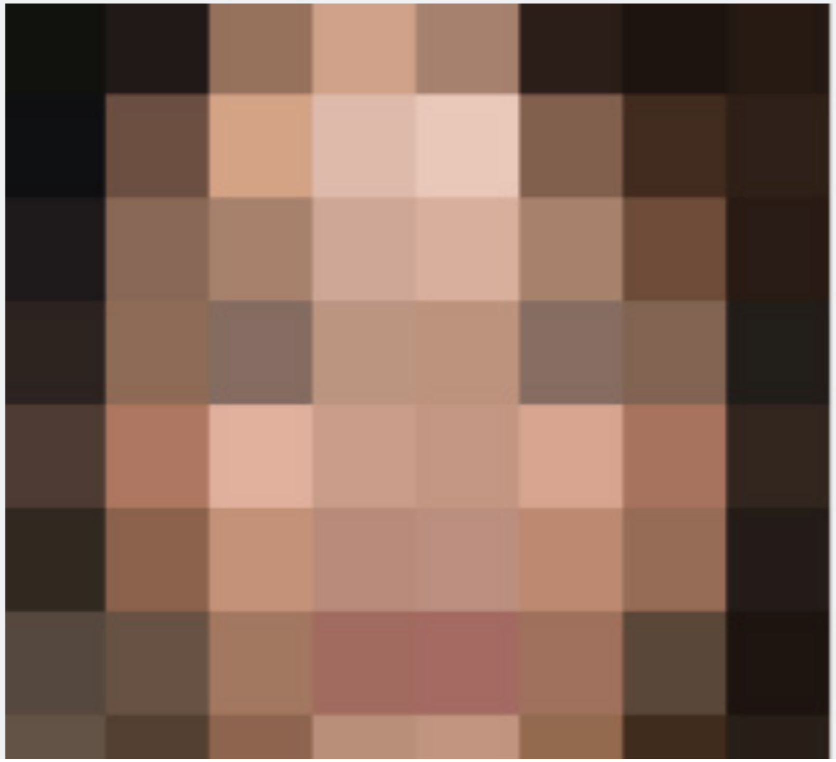
Figure copyright van der Oord et al., 2016. Reproduced with permission.

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



So far...

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

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

VAEs define intractable density function with latent **z**:

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

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

Stanford CS231n, Lecture 12

So far...

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

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

VAEs define intractable density function with latent **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

Stanford CS231n, Lecture 12

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.

X ⇨ **Neural network** (pretrained on ImageNet) ⇨

$$\begin{pmatrix} 0.04 \\ 0.85 \\ 0.02 \\ 0.07 \\ 0.81 \\ \vdots \\ 0.02 \end{pmatrix} \begin{array}{l} \text{"car"} \\ \text{"iguana} \\ \text{"tomato"} \\ \text{"bike"} \\ \text{"cat"} \\ \\ \text{"crab"} \end{array}$$

1. Rephrasing what we want:

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

2. Defining the loss function

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

3. Optimize the image

X ⇨ **Network** (pretrained on ImageNet) ⇨ $L(\hat{y}, y)$

After many iterations

?

$$\frac{\partial L}{\partial x} \qquad x = x - \alpha \frac{\partial L}{\partial x}$$

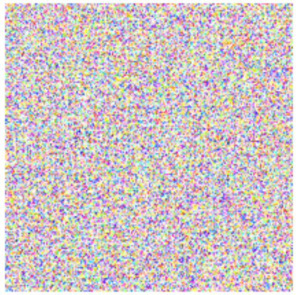
[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

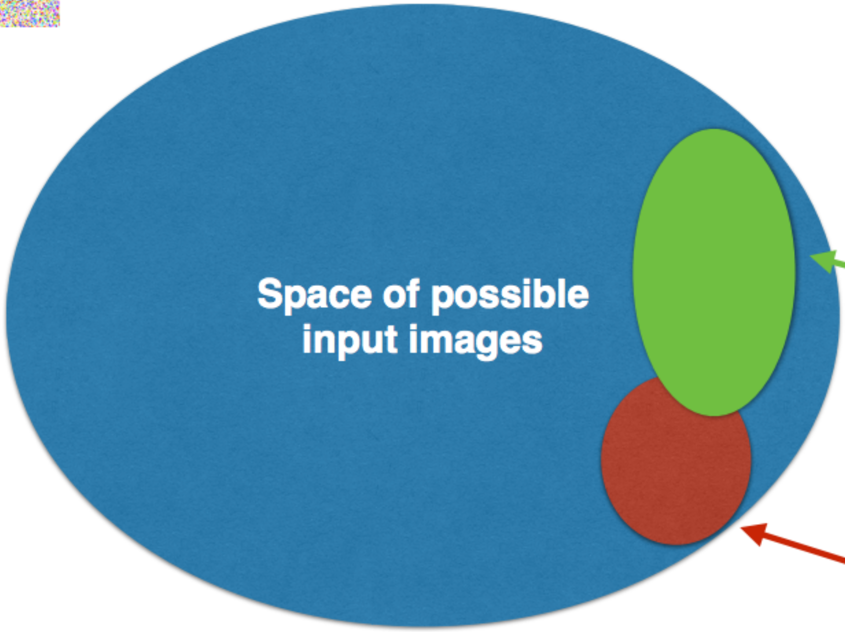
I. A. Attacking a network with adversarial examples

Question: Will the forged image **x** look like an **iguana**?



?

$$256^{32\times32\times3} \approx 10^{7400}$$



Space of images classified as iguanas

Space of real images

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

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 classify as an iguana.

⇨ **Neural network** (pretrained on ImageNet) ⇨

$$\begin{pmatrix} 0.04 \\ 0.85 \\ 0.02 \\ 0.07 \\ 0.81 \\ \vdots \\ 0.02 \end{pmatrix} \begin{matrix} \text{"car"} \\ \text{"iguana} \\ \text{"tomato"} \\ \text{"bike"} \\ \text{"cat"} \\ \\ \text{"crab"} \end{matrix}$$

1. Rephrasing what we want:

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

And: $x = x_{cat}$

2. Defining the loss function

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

3. Optimize the image

⇨ **Network** (pretrained on ImageNet) ⇨ $L(\hat{y}, y)$

After many iterations

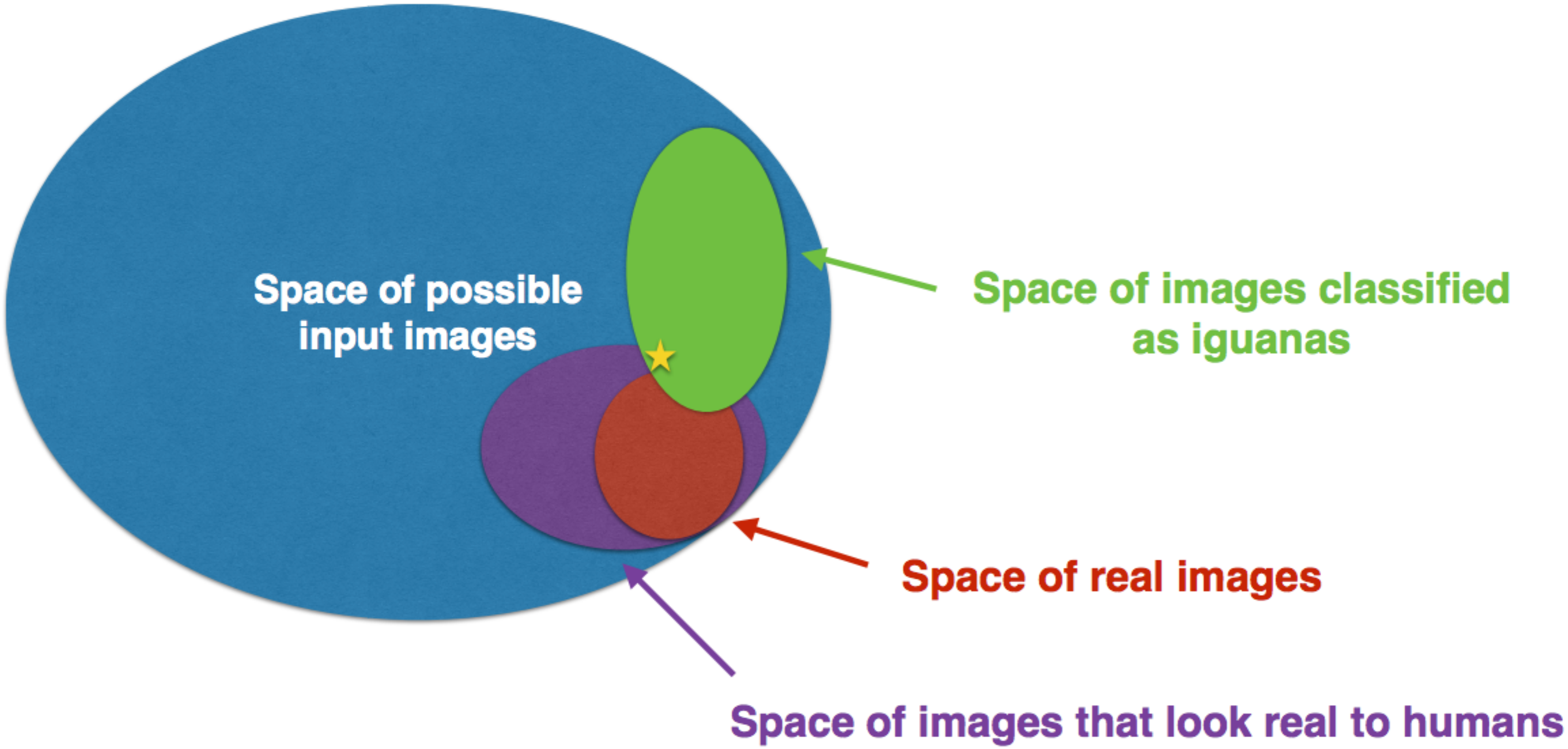
$\frac{\partial L}{\partial x}$ $\quad x = x - \alpha \frac{\partial L}{\partial x}$

[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

$$256^{32\times32\times3} \approx 10^{7400}$$



I. B. Defenses against adversarial examples

Menti

Knowledge of the attacker:

- White-box
- Black-box

Solution 1

- Create a SafetyNet

$x =$ 

$y =$ cat

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

[Lu et al. (2017): SafetyNet: Detecting and Rejecting Adversarial Examples Robustly]
[Harini Kannan et al. (2018): Adversarial Logit Pairing]

deep imaging

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?"

<u>To prove true</u>: need just one example

<u>To prove false</u>: 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?"

<u>To prove true</u>: need just one example implementation

<u>To prove false</u>: Need to show *all possible* implementations fail

4 different adversarial examples for object detector:

4 different adversarial examples for object classifier:

deep imaging

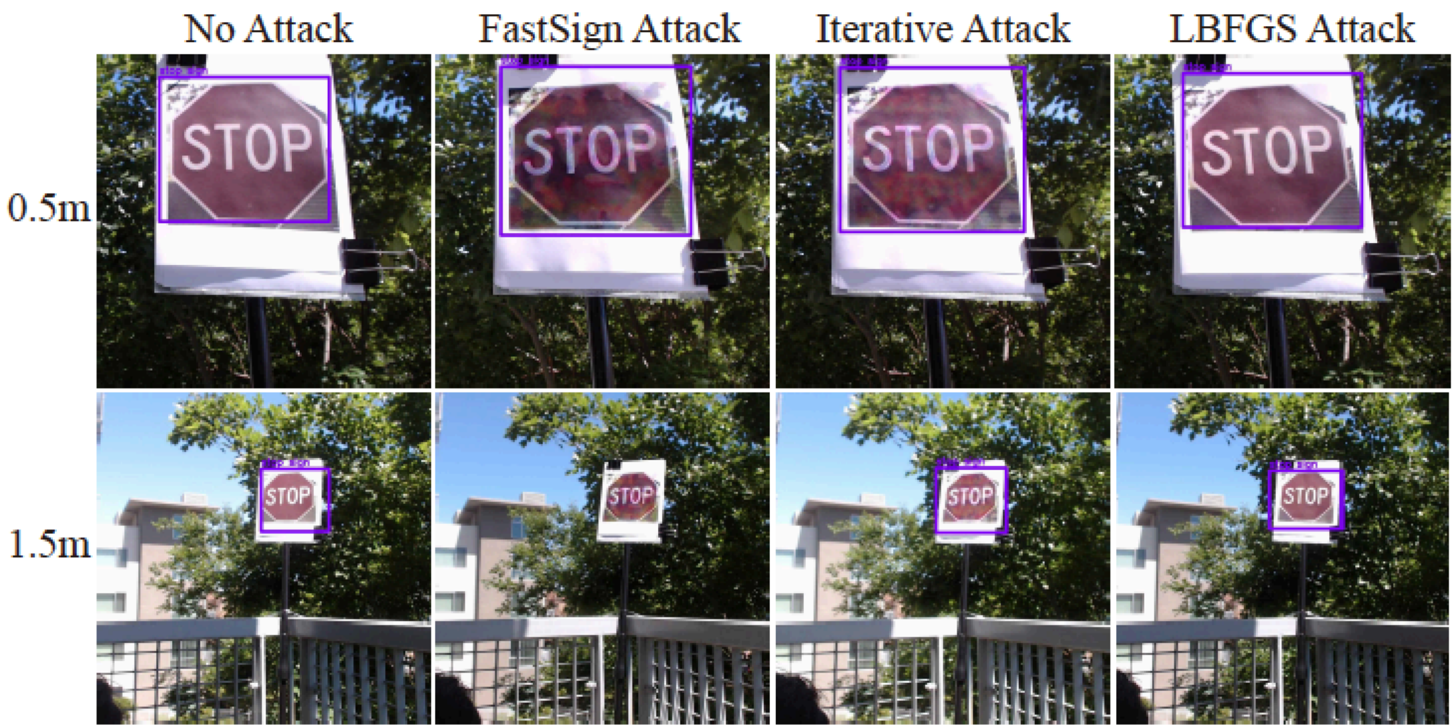


Figure 3: This figure shows experiment setup, and we use the printed stop signs to simulate real stop signs with natural background. These are examples for successful 0.5 meters and 1.5 meters detection: both original images and adversarial examples are detected in both distances. It demonstrates that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

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

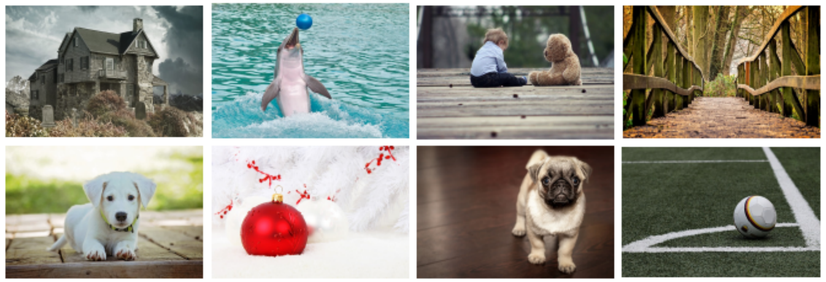
https://blog.openai.com/robust-adversarial-inputs/

Generative adversarial networks

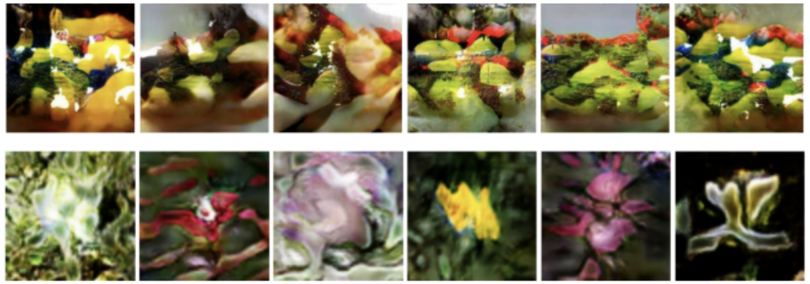
II.A - Motivation

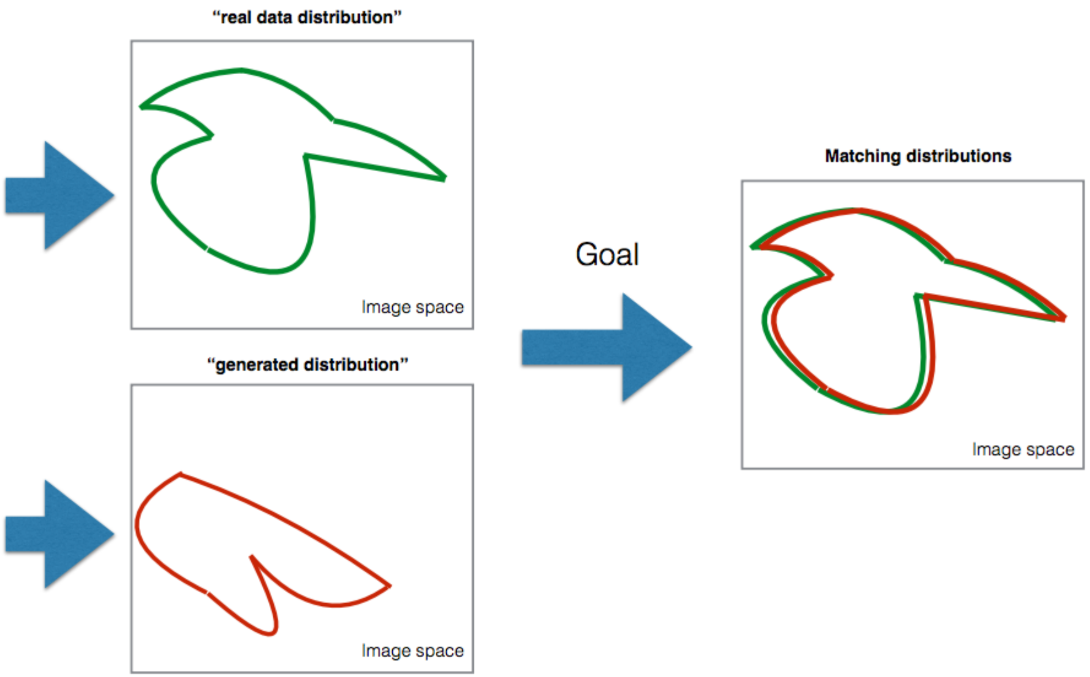
Probability distributions:

Samples from the "real data distribution"



Samples from the "generated distribution"



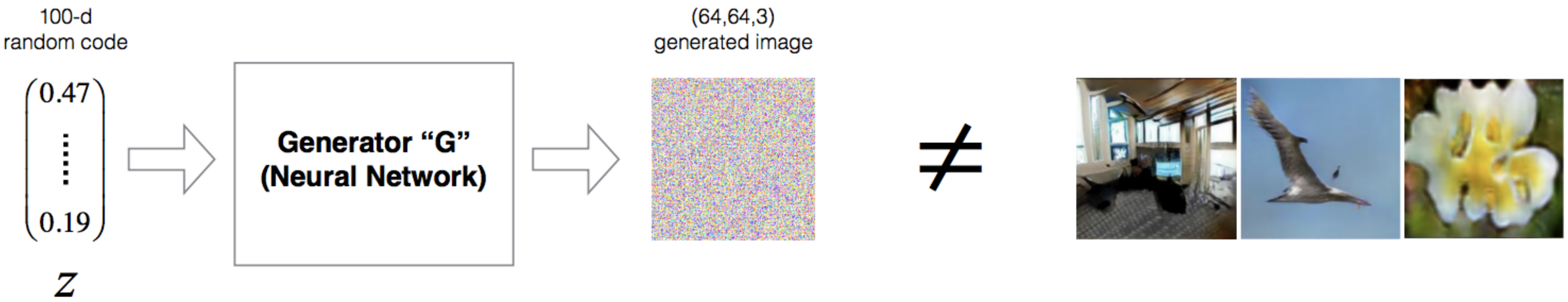


[Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas (2017): StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

II.B - G/D Game



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

[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

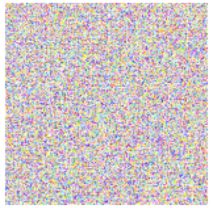
II.B - G/D Game

100-d random code

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

Generator "G" (Neural Network)

(64,64,3) generated image



Run Adam simultaneously on two minibatches (true data / generated data)

x

Discriminator "D" (Neural Network)

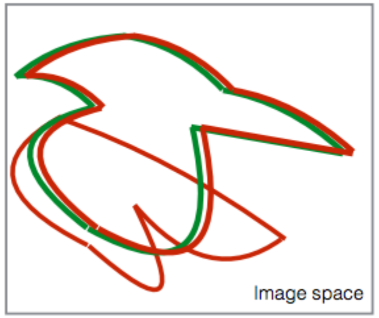
Gradients

Binary classification

$$\begin{cases} y = 0 & if \quad x = G(z) \\ y = 1 & otherwise \end{cases}$$

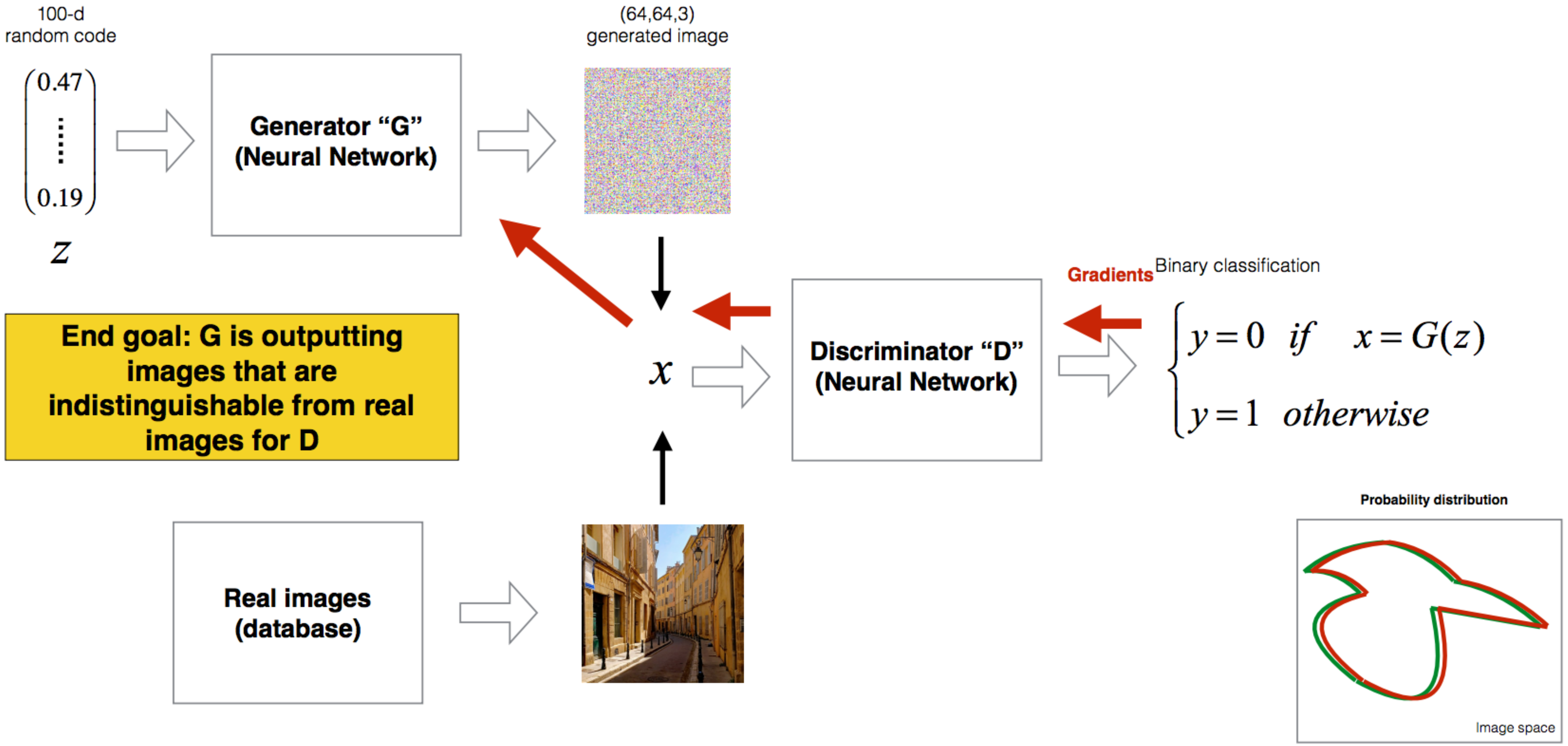
Real images (database)

Probability distributions



Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

II.B - G/D Game



100-d
random code
$\begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix}$
z
Generator "G"
(Neural Network)
(64,64,3)
generated image
x
Real images
(database)
Discriminator "D"
(Neural Network)
Gradients
Binary classification
$\begin{cases} y=0 & if \quad x=G(z) \\ y=1 & otherwise \end{cases}$
End goal: G is outputting images that are indistinguishable from real images for D
Probability distribution
Image space

II.B - G/D Game

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

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

- 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

Stanford CS230, Lecture 3

II. D. In terms of code

imaging

```python
# 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)
```

```python
def build_discriminator(self):

    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)
```

Erik Linder-Norén (Github): eriklindernoren/Keras-GAN: link]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

II.E - Nice results

Operation on codes

Code 1

$\begin{pmatrix}0.12\\ \vdots\\ 0.92\end{pmatrix} \Rightarrow$ **Generator "G" (Neural Network)** $\Rightarrow$

(64,64,3)
generated image

Code 2

$\begin{pmatrix}0.47\\ \vdots\\ 0.19\end{pmatrix} \Rightarrow$ **Generator "G" (Neural Network)** $\Rightarrow$

(64,64,3)
generated image

Code 3

$\begin{pmatrix}0.42\\ \vdots\\ 0.07\end{pmatrix} \Rightarrow$ **Generator "G" (Neural Network)** $\Rightarrow$

(64,64,3)
generated image

Code 1 Code 2 Code 3

$\begin{pmatrix}0.12\\ \vdots\\ 0.92\end{pmatrix} - \begin{pmatrix}0.47\\ \vdots\\ 0.19\end{pmatrix} + \begin{pmatrix}0.42\\ \vdots\\ 0.07\end{pmatrix} \Rightarrow$ **Generator "G" (Neural Network)** $\Rightarrow$

Man with glasses - man + woman = woman with glasses

[Radford et al. (2015): UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS]

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...