Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

Generative models, adversarial examples, and more

Final Project Details

- - Final code and presentation due via email: Thursday May 2, 5pm
- - Final presentation time slots: Thursday May 2, 7pm - 10pm
- - Final presentation paper write-up, website template and permission form due: Saturday May 4 at 11:59pm
- - Final presentations will occur via Zoom, please coordinate with team members and practice ahead of time!

deep imaging

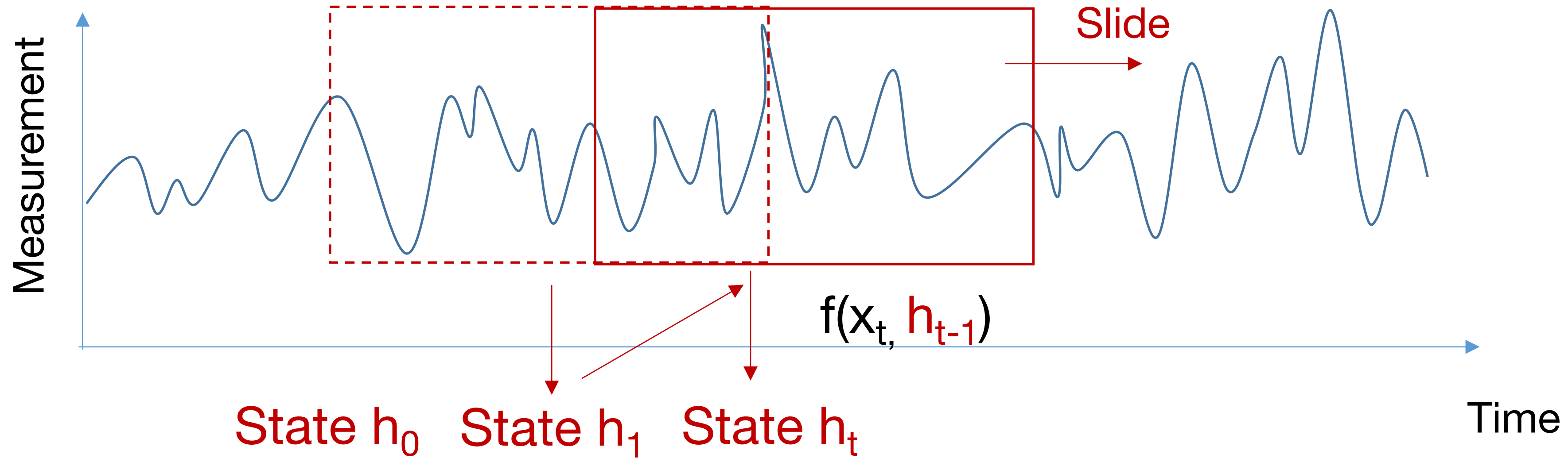
Resources for this lecture:

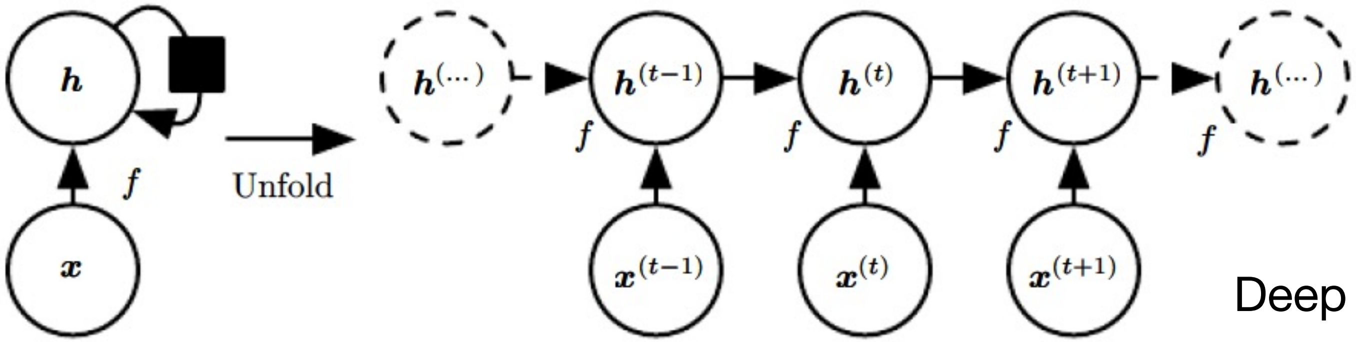
- Stanford CS231n, Lecture 12
- Stanford CS230 course slides
- Deep Learning book, chapter 15
- Number of papers cited throughout slides

Recurrent neural networks in a nutshell

RNN’s: Examine signals as a function of time

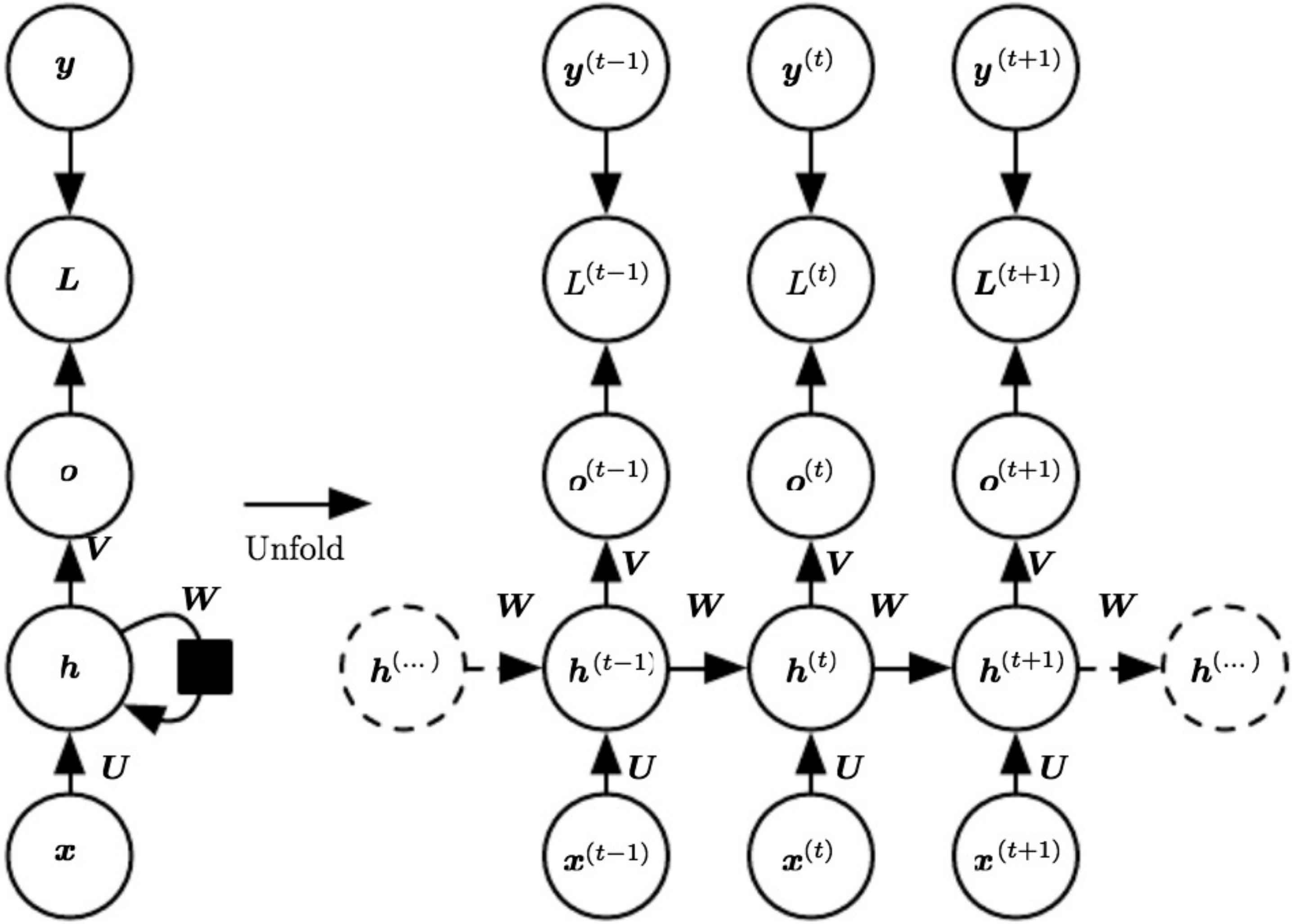
E.g., establish if mouse was scared from this EEG recording





Deep Learning Book, Ch. 10

Many-to-many recurrent neural network

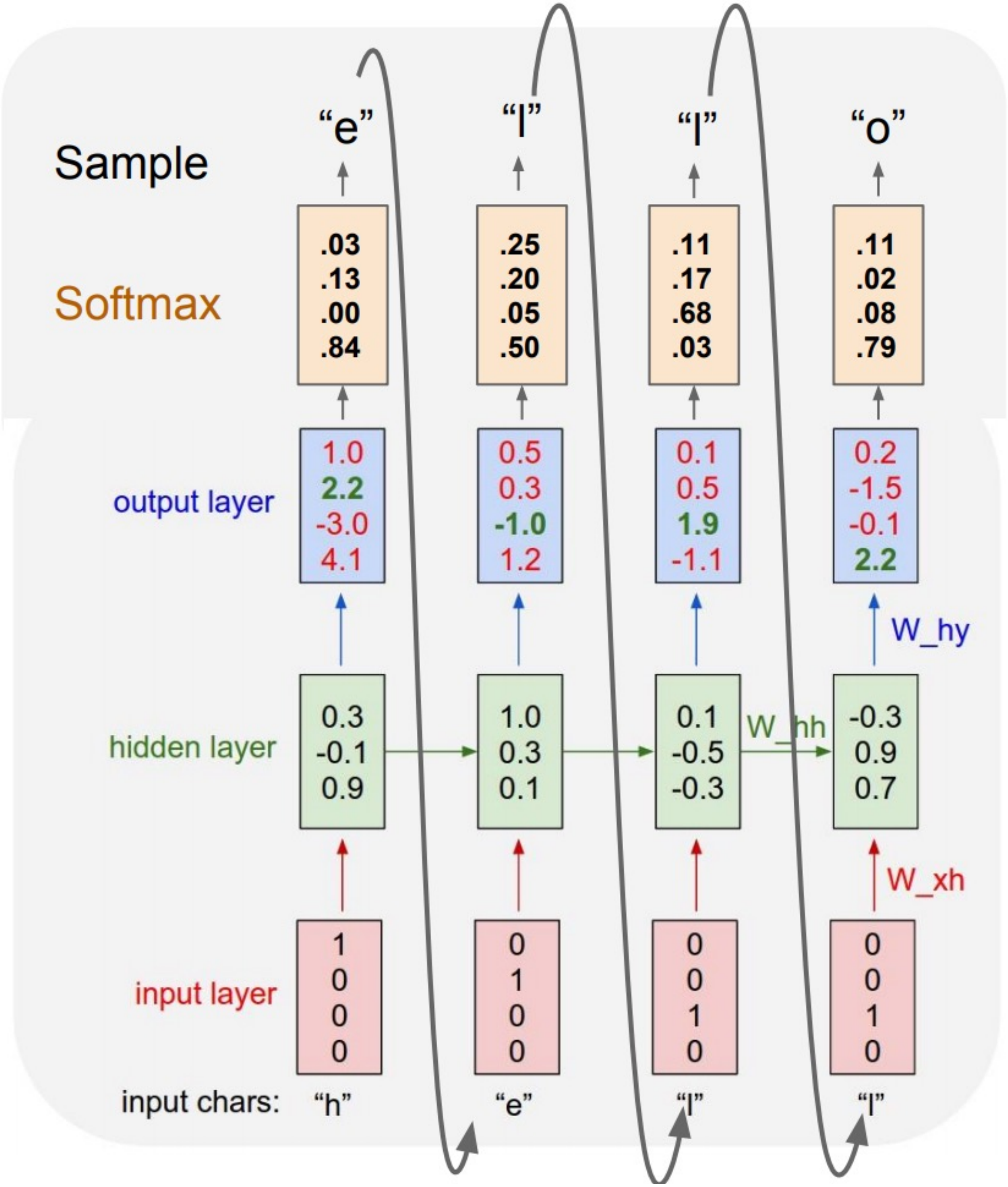


$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

Example: Character-level Language Model Sampling

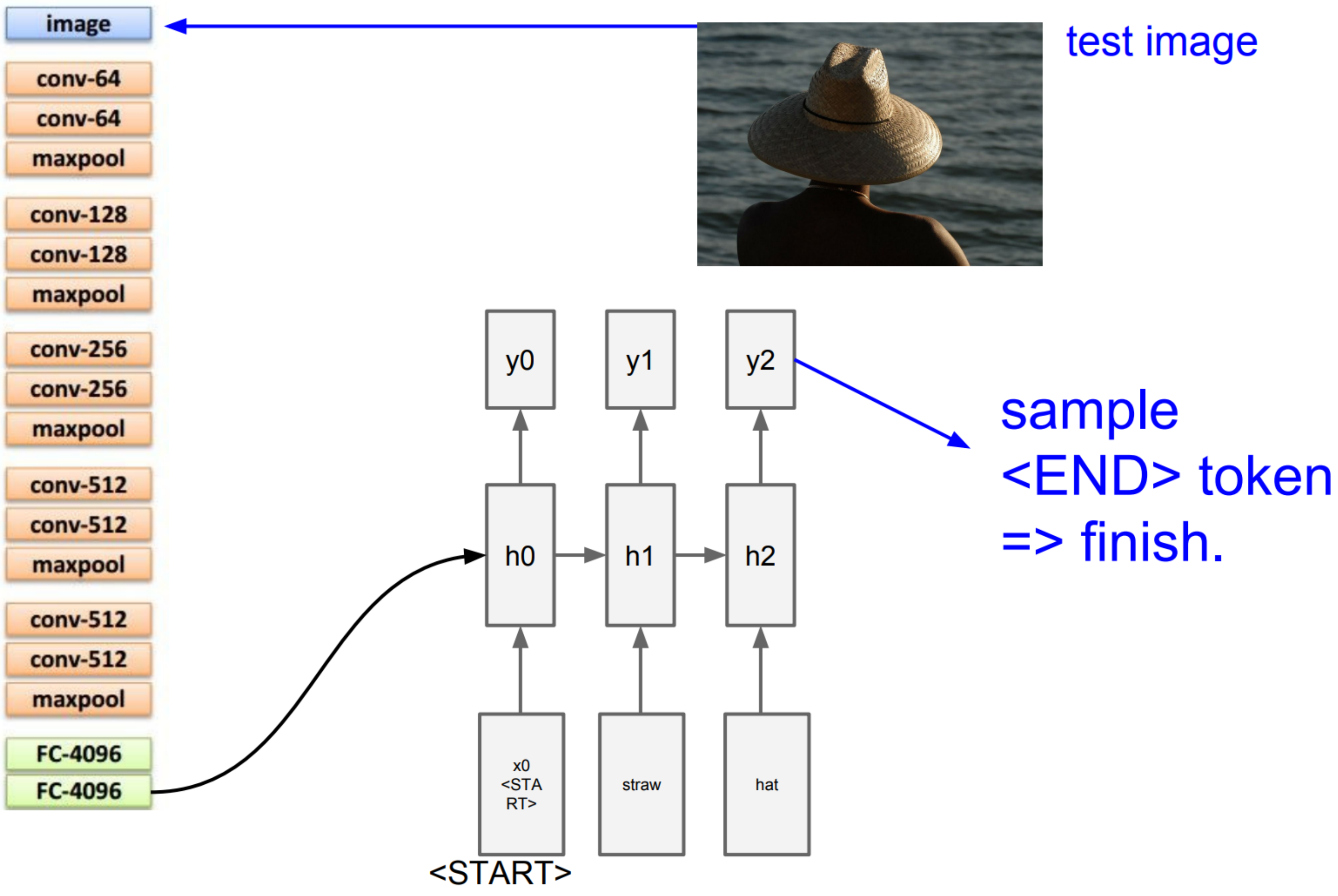
Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



From Stanford CS231n Lecture 10 slides

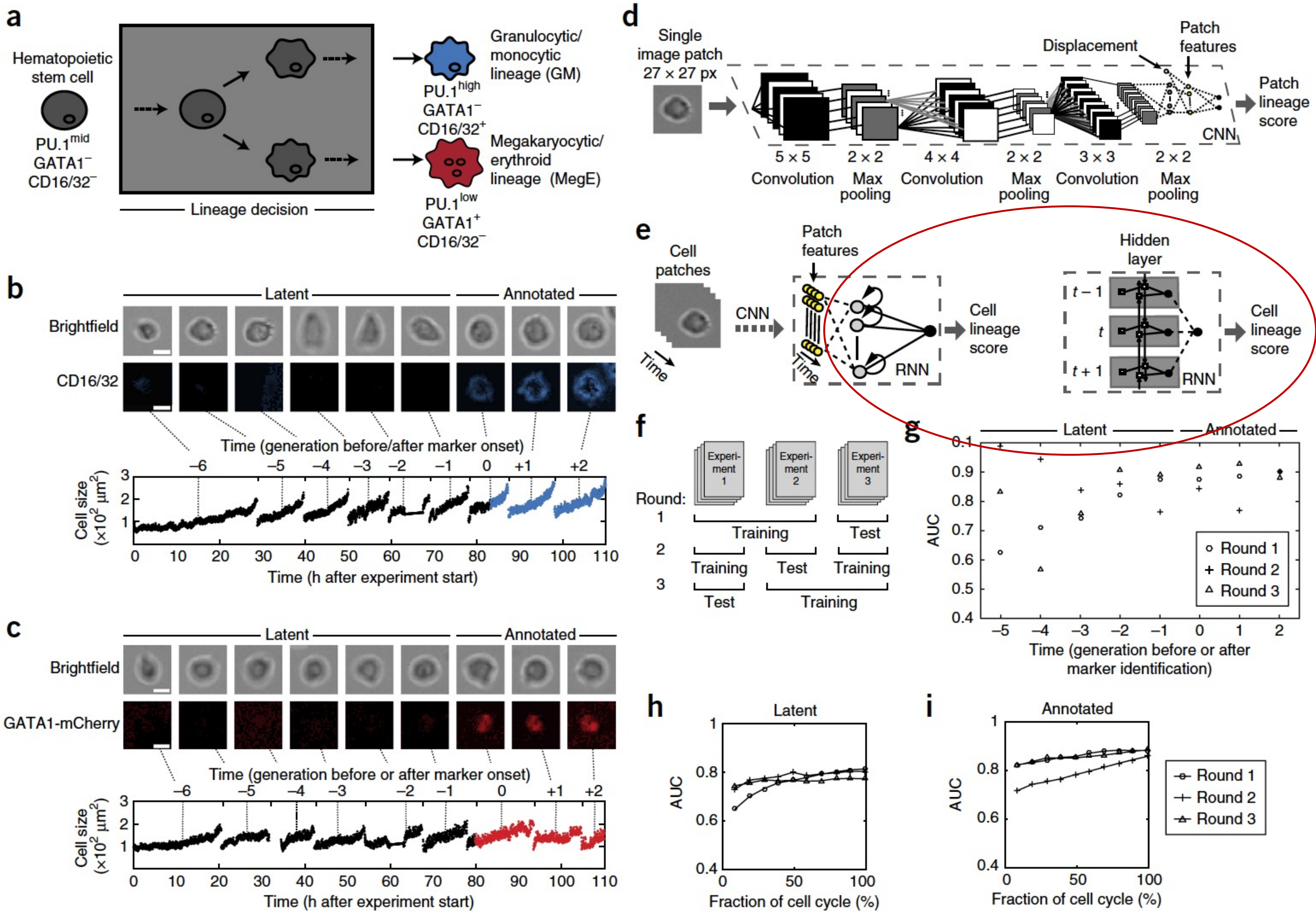
Example: Image captioning



From Stanford CS231n Lecture 10 slides

Prospective identification of hematopoietic lineage choice by deep learning

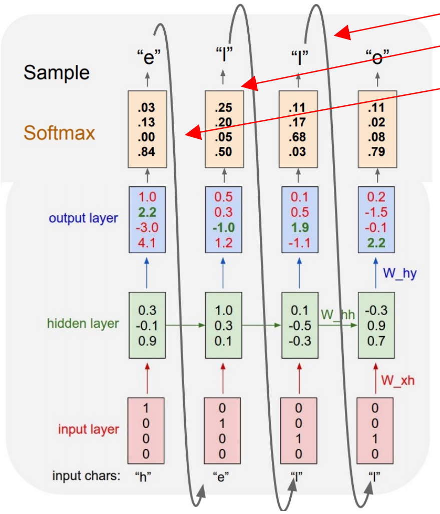
Felix Buggenthin[1,6], Florian Buettner[1,2,6], Philipp S Hoppe[3,4], Max Endele[3], Manuel Kroiss[1,5], Michael Strasser[1], Michael Schwarzfischer[1], Dirk Loeffler[3,4], Konstantinos D Kokkaliaris[3,4], Oliver Hilsenbeck[3,4], Timm Schroeder[3,4], Fabian J Theis[1,5] & Carsten Marr[1]



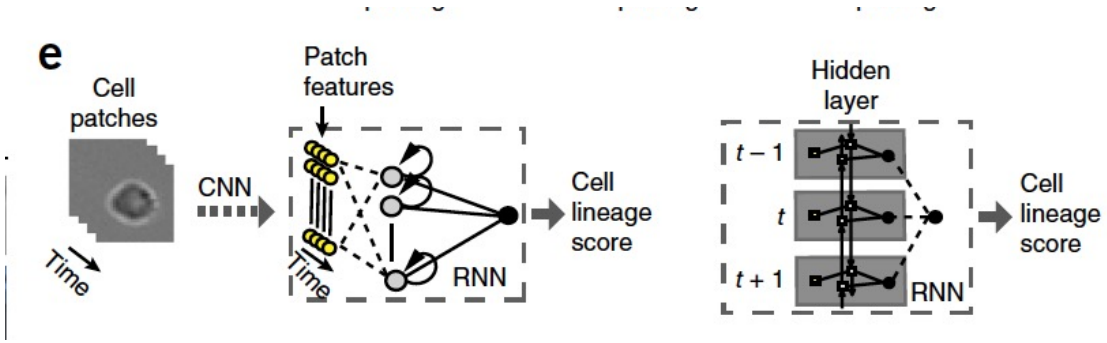
Output generative processing - be careful…

Output-generative: “Let’s figure out the next output based on the other outputs”

“Non-output generative”: “I’ll just use all of my input data to determine my final output”

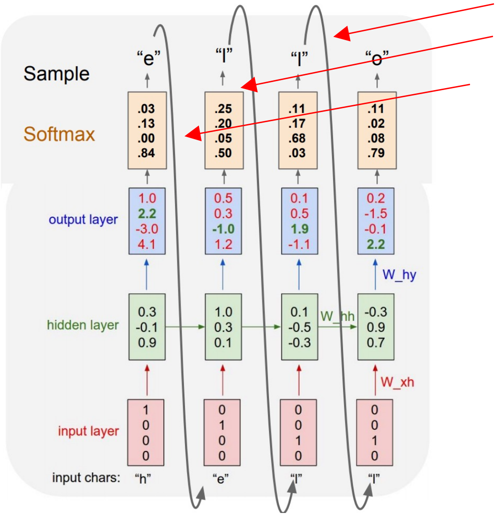


These lines imply a lot!



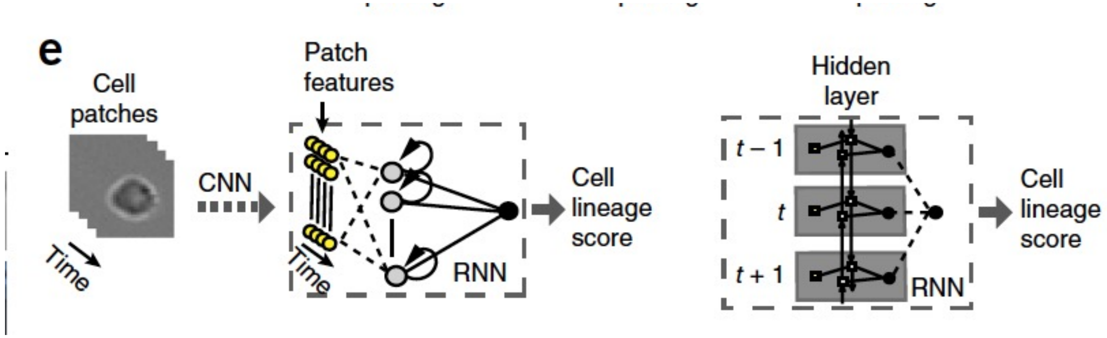
Output generative processing - be careful…

Output-generative: “Let’s figure out the next output based on the other outputs”



These lines imply a lot!

“Non-output generative”: “I’ll just use all of my input data to determine my final output”



Pros: You get a lot more “bang for your buck”

Cons: Fine line between hallucination and trustworthy output…

Pros: More repeatable and interpretable results

Cons: Hard to get more outputs from inputs (and not as “cool”)

deep imaging

What are we possibly missing from the many-to-many model?

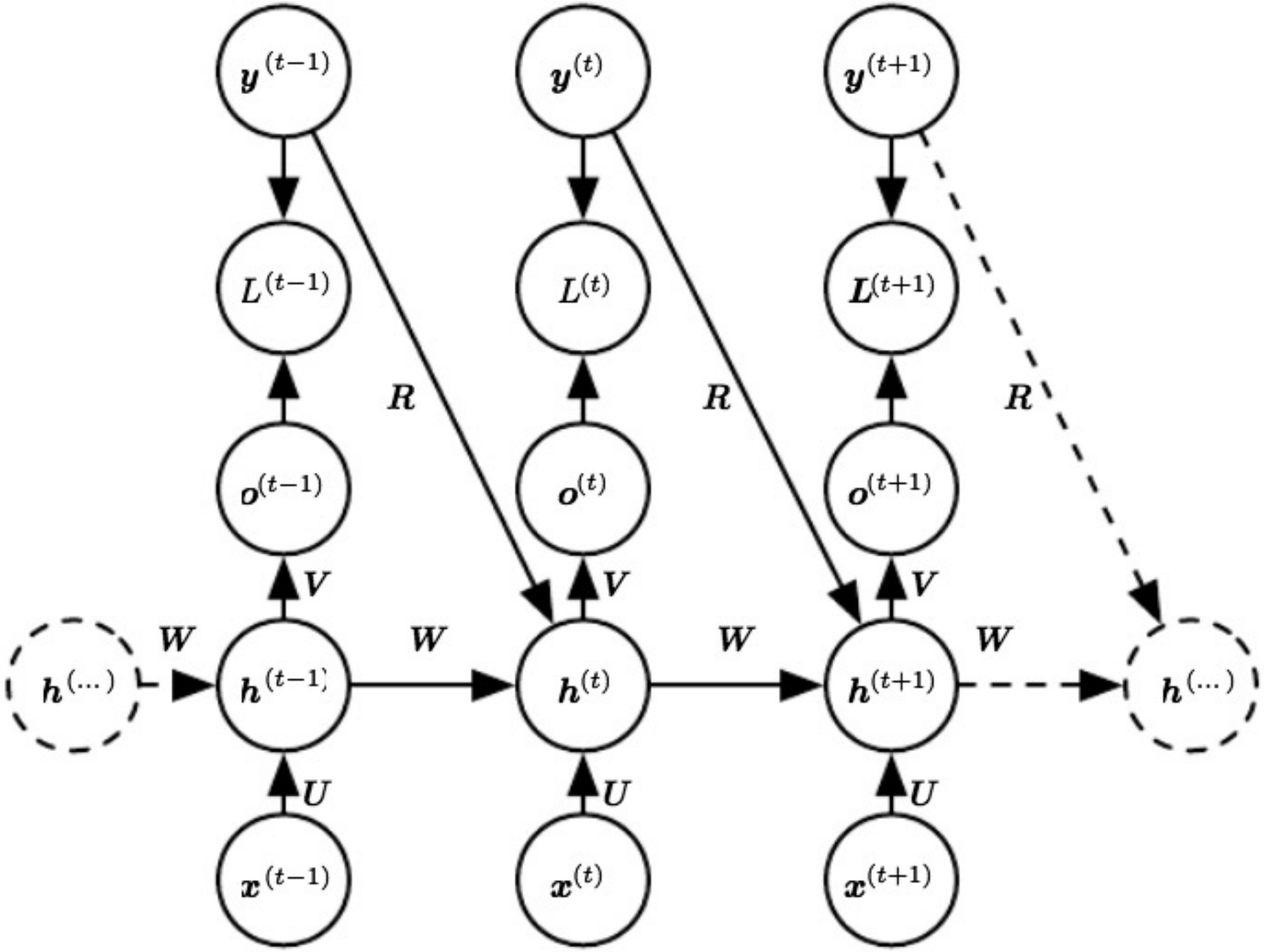
1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

Let the network become dependent on past labels as well:

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t-1)})$$

Conditional recurrent neural network



What are we possibly missing from the many-to-many model?

1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

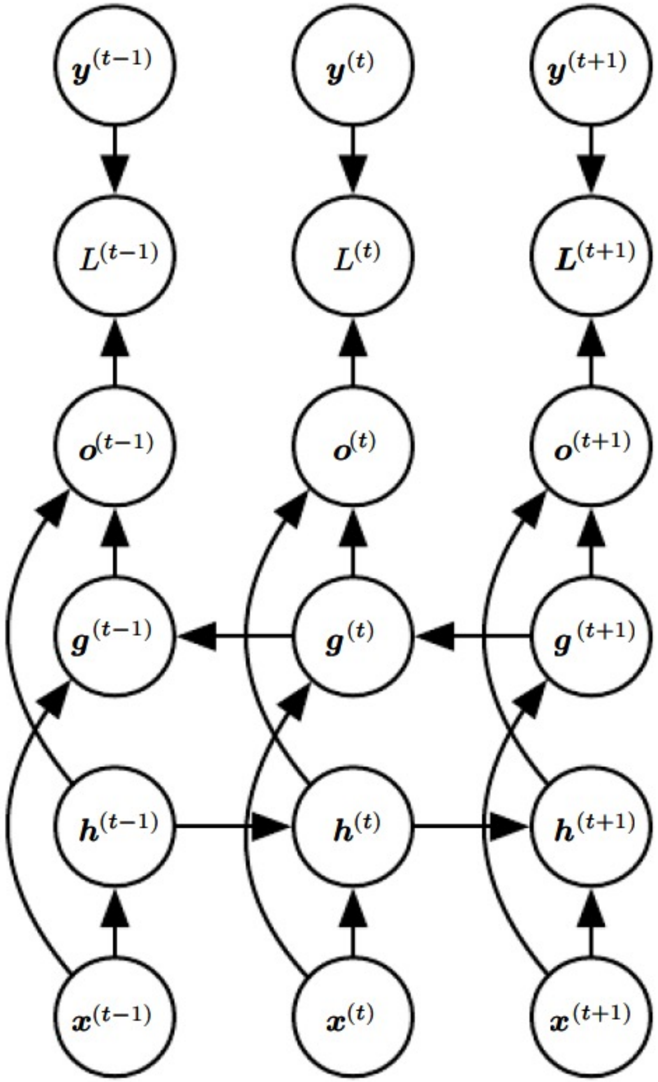
$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

Let the network become dependent on past labels as well:

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t-1)})$$

2. Only considering one direction in sequence/time…

Other extensions: bi-directional analysis



- Consider future and past events jointly
- Add a third matrix that takes future hidden states in as well
- E.g., sentence structure is not purely causal
- Handwriting recognition, speech analysis, etc.

What are we possibly missing from the many-to-many model?

1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

Let the network become dependent on past labels as well:

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t-1)})$$

2. Only considering one direction in sequence/time…

3. Chaining things together is not necessarily the ideal way to maintain long connections

RNN's have limited memory and can suffer from exploding gradients

Hidden weights effectively follow a recursive relationship:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^\top \boldsymbol{h}^{(t-1)} \longrightarrow \boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^t\right)^\top \boldsymbol{h}^{(0)},$$

RNN's have limited memory and can suffer from exploding gradients

Hidden weights effectively follow a recursive relationship:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top}\boldsymbol{h}^{(t-1)} \longrightarrow \boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^{t}\right)^{\top}\boldsymbol{h}^{(0)},$$

If W admits it, can perform eigenvvector decomposition to obtain,

$$\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top},$$

RNN's have limited memory and can suffer from exploding gradients

Hidden weights effectively follow a recursive relationship:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top}\boldsymbol{h}^{(t-1)} \longrightarrow \boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^{t}\right)^{\top}\boldsymbol{h}^{(0)},$$

If W admits it, can perform eigenvvector decomposition to obtain,

$$\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top},$$

In this space, power relationship W^t alters just eigenvalues, does not rotate eigenvectors:

$$\boldsymbol{h}^{(t)} = \boldsymbol{Q}^{\top}\boldsymbol{\Lambda}^{t}\boldsymbol{Q}\boldsymbol{h}^{(0)}$$

Thus, if the eigenvector is large (the largest), it will explode. Remaining eigenvectors eventually vanish

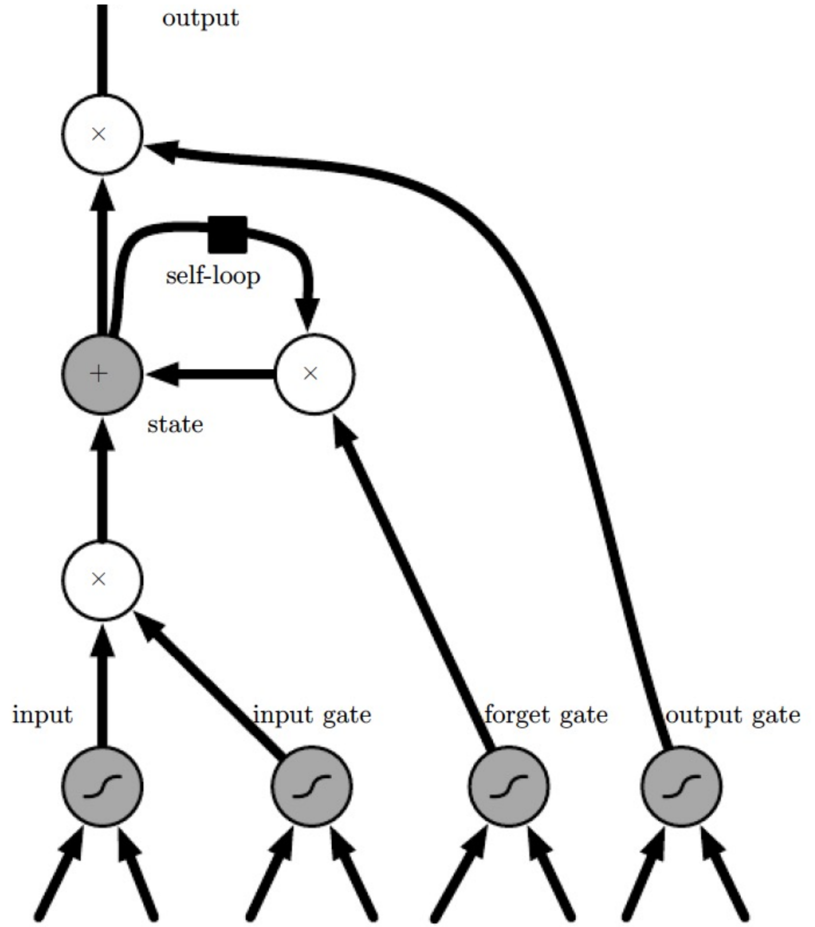
S. Hochreiter and J. Schmidhuber (1997)

The long short-term memory network

Long Short-Term Memory
https://dl.acm.org › citation
by S Hochreiter - 1997 - Cited by 24398 - Related articles
Nov 1, 1997 - Hochreiter, S., & Schmidhuber, J. (1996). Bridging long time lags by weight guessing and "**long short-term memory**." In F. L. Silva, J. C. Principe, ...

Additions:

- Self-loop to maintain “memory”
  - Allow gradients to flow for a long time
- Weight of self-loop gated by “Forget gate”
  - Forgetting depends on data
  - Memory time scale is thus dynamic
- Output gate
  - Can turn on/shut off everything



S. Hochreiter and J. Schmidhuber (1997)

Long Short-Term Memory
https://dl.acm.org › citation
by S Hochreiter - 1997 - Cited by 24398 - Related articles
Nov 1, 1997 - Hochreiter, S., & Schmidhuber, J. (1996). Bridging long time lags by weight guessing and "**long short-term memory**." In F. L. Silva, J. C. Principe, ...

The long short-term memory network

Forget gate:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

Internal state:

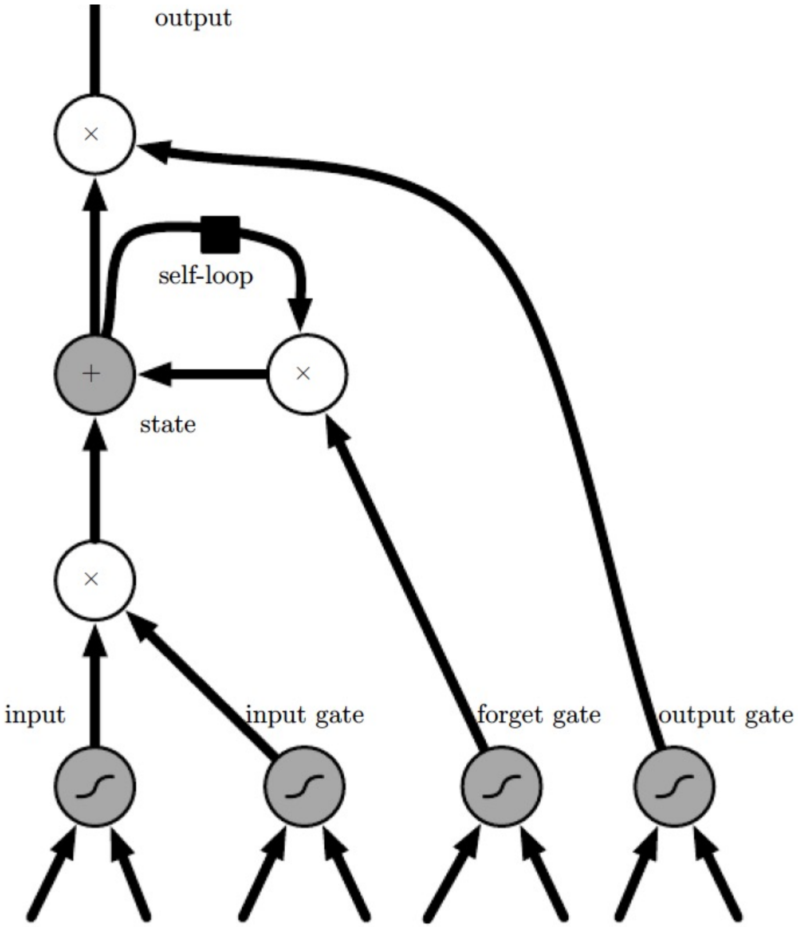
$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

External input gate:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

LSTM output:

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}$$



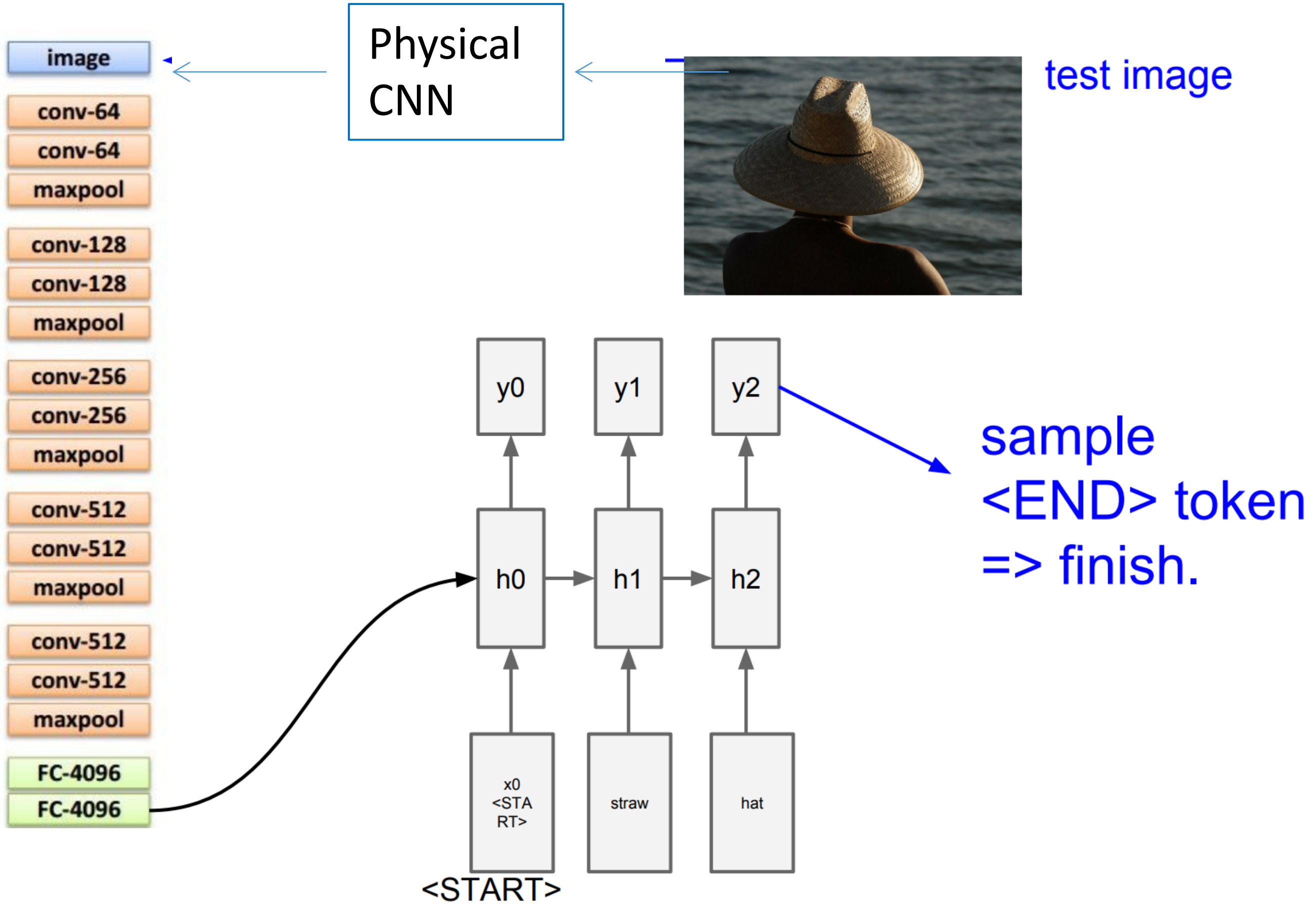
deep imaging

Brainstorming time – physical layers in an RNN???

Brainstorming time – physical layers in an RNN???

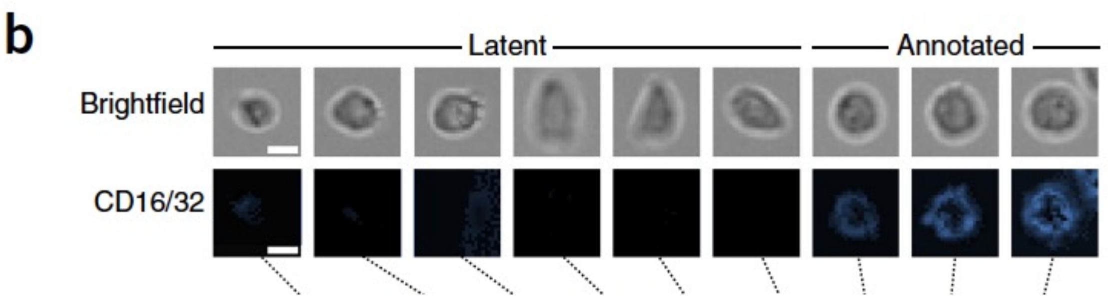
Simple example, “Output generative” flavor

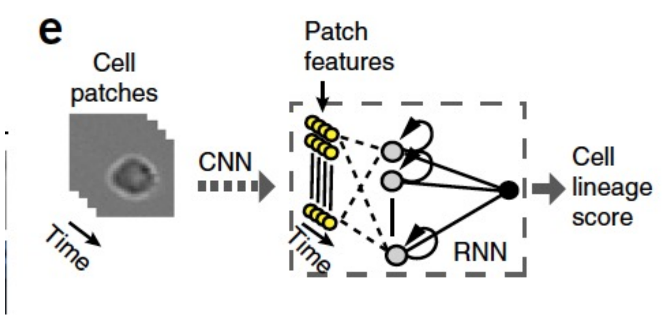
Design an optimal X to produce the best image captions



Brainstorming time – physical layers in an RNN???

Simple example, "Non-output generative" flavor





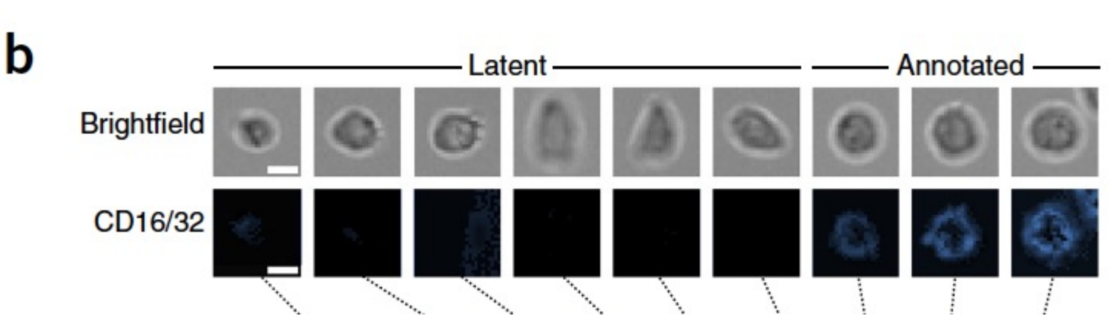
Brainstorming time – physical layers in an RNN???

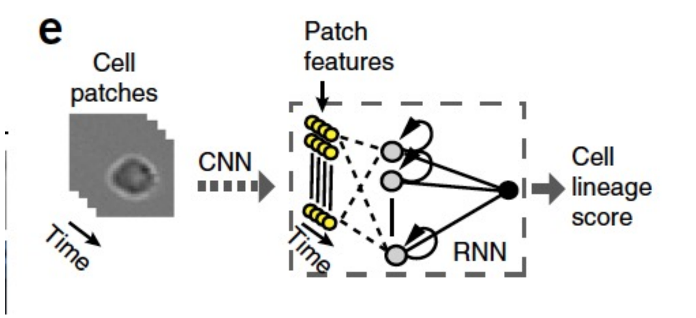
Simple example, “Non-output generative” flavor

Physical model down-samples movie sequence

1 0 0 1 0 0 0 0 1 ……

Optimize selected frames to reduce overhead





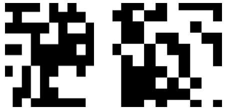
Brainstorming time – physical layers in an RNN???

Simple example, “Non-output generative” flavor

Physical model down-samples movie sequence

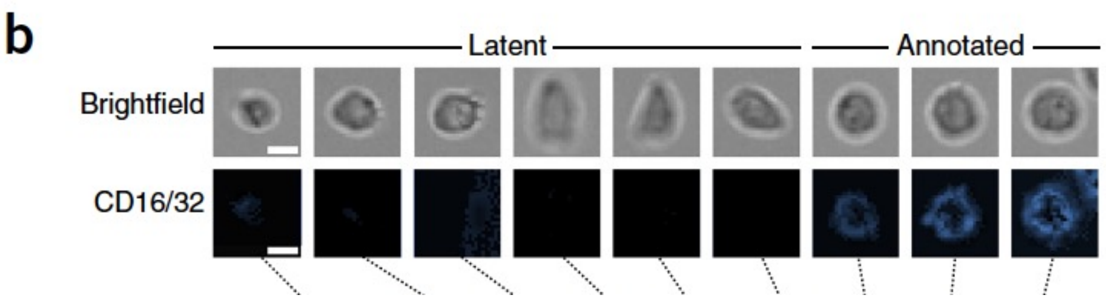
1 0 0 1 0 0 0 0 1 ……

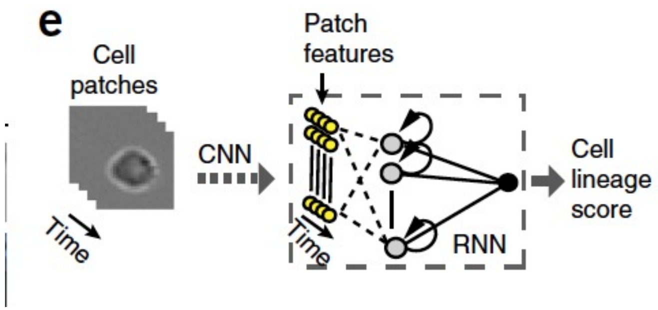
Optimize selected frames to reduce overhead



…..

Or, optimize illumination/sampling pattern





Brainstorming time – physical layers in an RNN???

Take a bit of time and try to think about/write down the following:

- With your image data (or some data that you are interested), what do you measure over time?
- What would your input data for an RNN be, and what might be a useful output?
- What physical parameters influence how you measure data over time?
- What physical parameters might be useful to tweak to improve your output?
- Can you think of a way to model that parameter in an RNN?

Supervised versus unsupervised learning

Supervised

- Have data **x** with labels **y**
- Goal is to learn function f(**x**) = **y**

Unsupervised

- Just have data **x** with *no* labels
- Figure out and exploit underlying structure of data

Supervised versus unsupervised learning

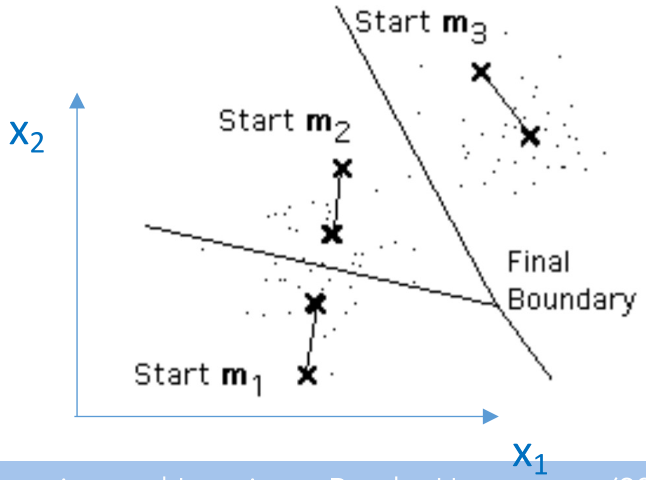
Supervised

- Have data **x** with labels **y**
- Goal is to learn function f(**x**) = **y**

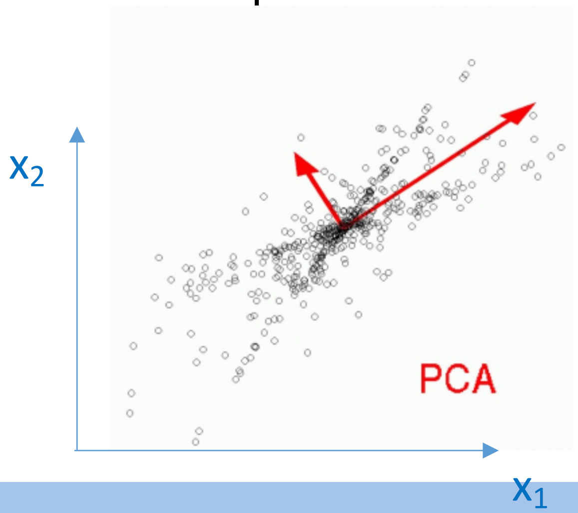
Unsupervised

- Just have data **x** with *no* labels
- Figure out and exploit underlying structure of data

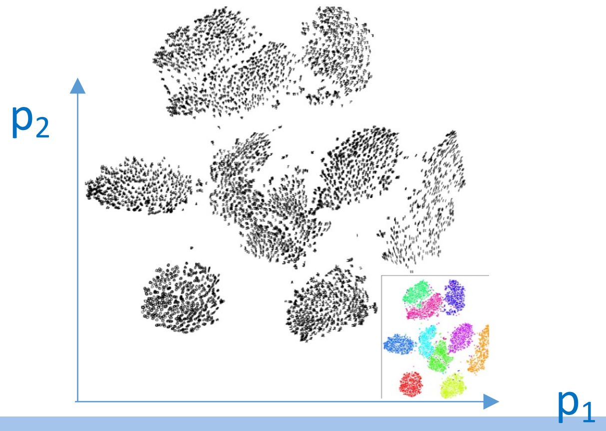
Example: K-means clustering



Example: PCA



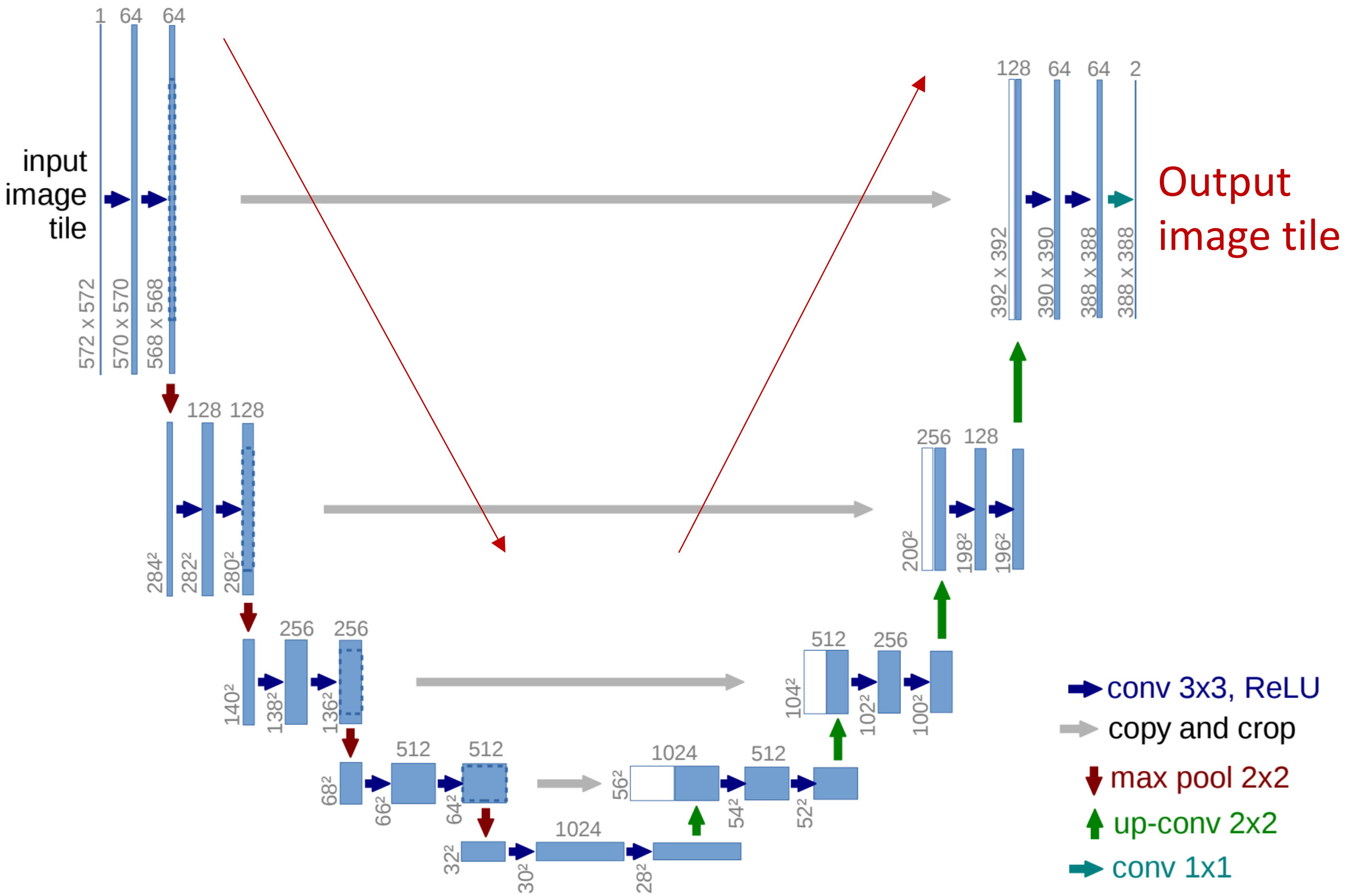
Example: tSNE



Unsupervised learning example: autoencoder

deep imaging

U-Net Architecture



U-Net: Convolutional Networks for Biomedical Image Segmentation

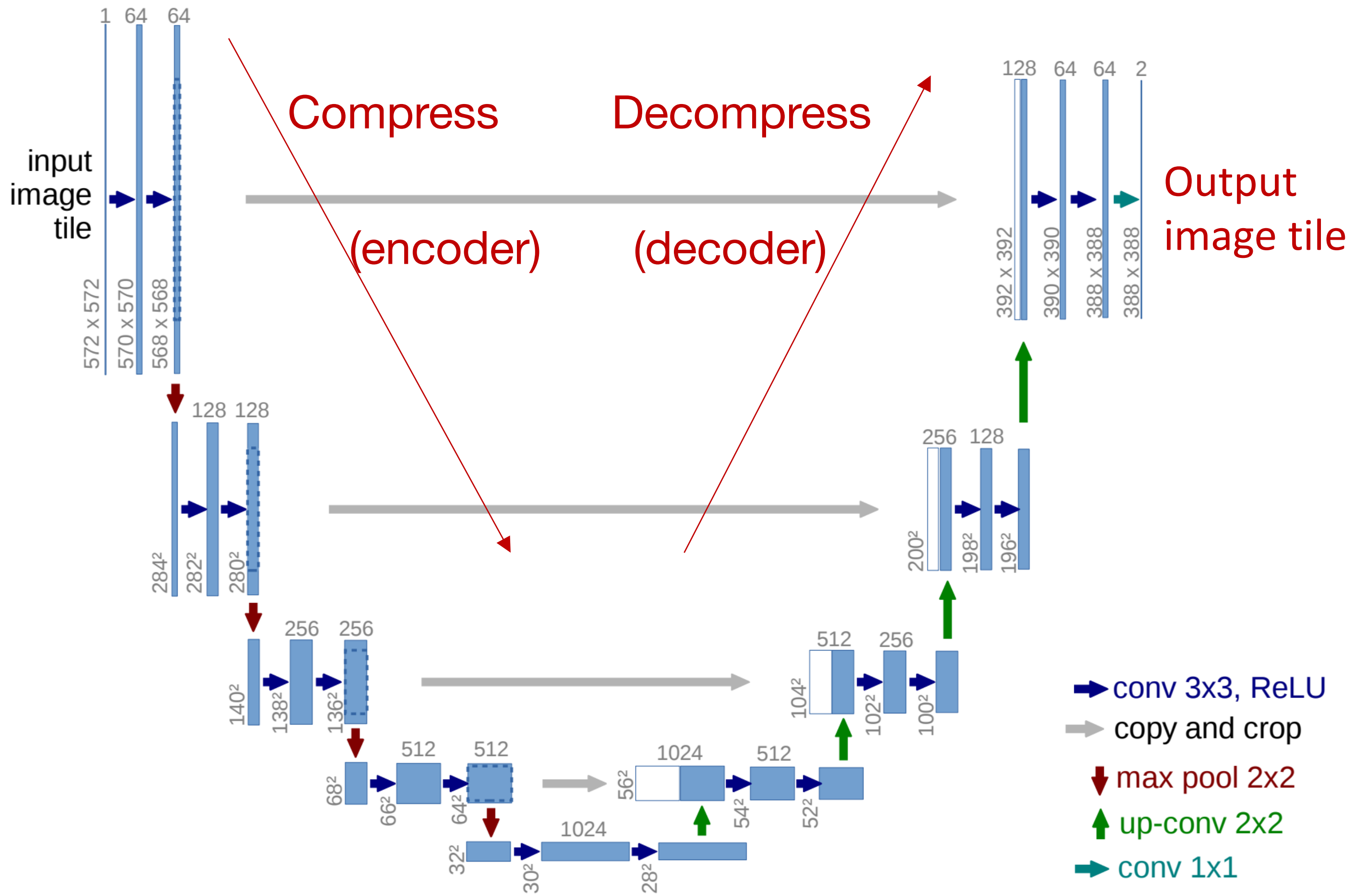
Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Unsupervised learning example: autoencoder

- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions

U-Net Architecture



U-Net: Convolutional Networks for Biomedical Image Segmentation

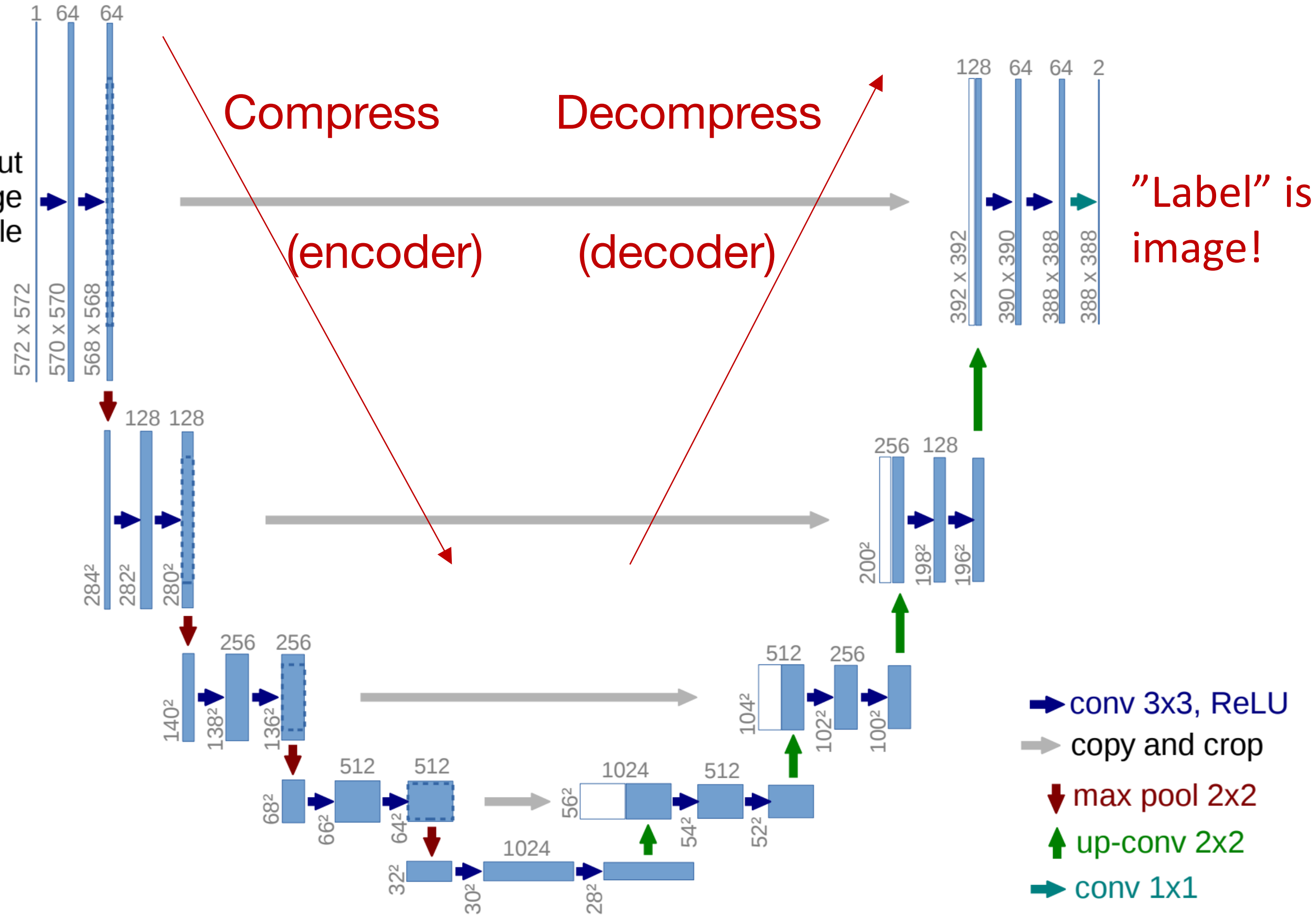
Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Example: Learned compression

Input image

U-Net Architecture



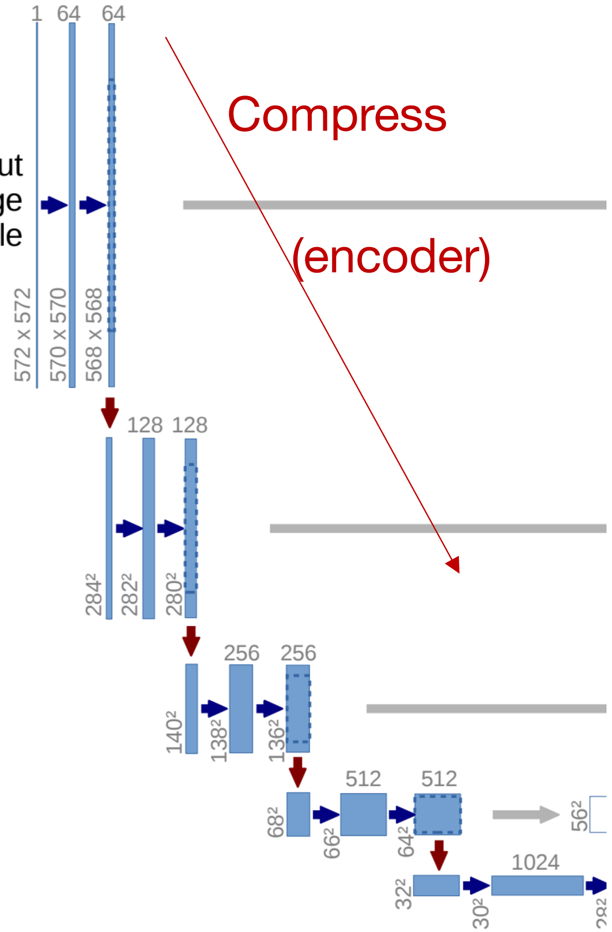
”Label” is image!

Input image

Example: Learned compression

U-Net Architecture

New image



Compress

(encoder)

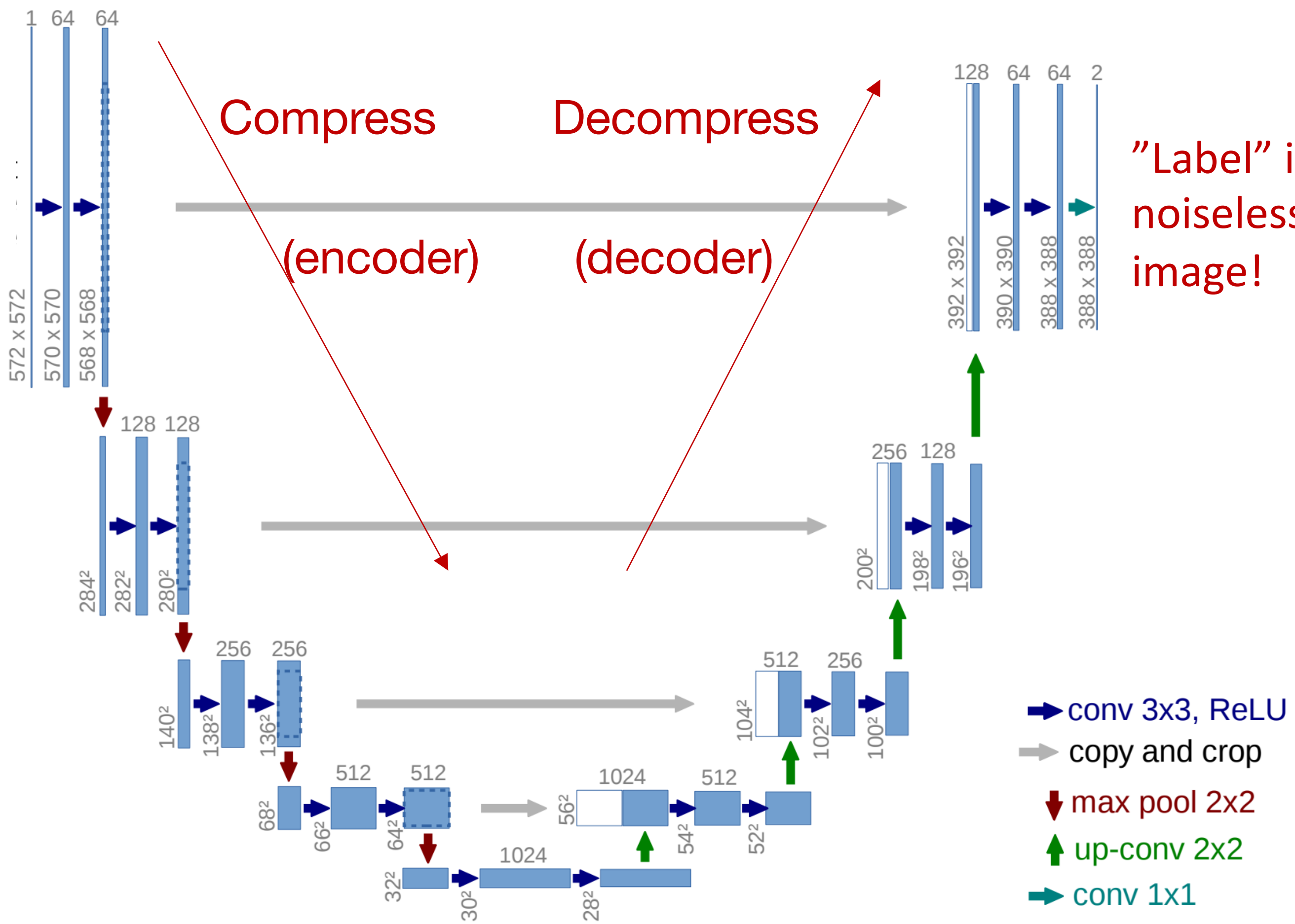
Smaller, compressed representation

Smartphone memory

Example: Denoising Autoencoder

U-Net Architecture

Input noisy image



”Label” is noiseless image!

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

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

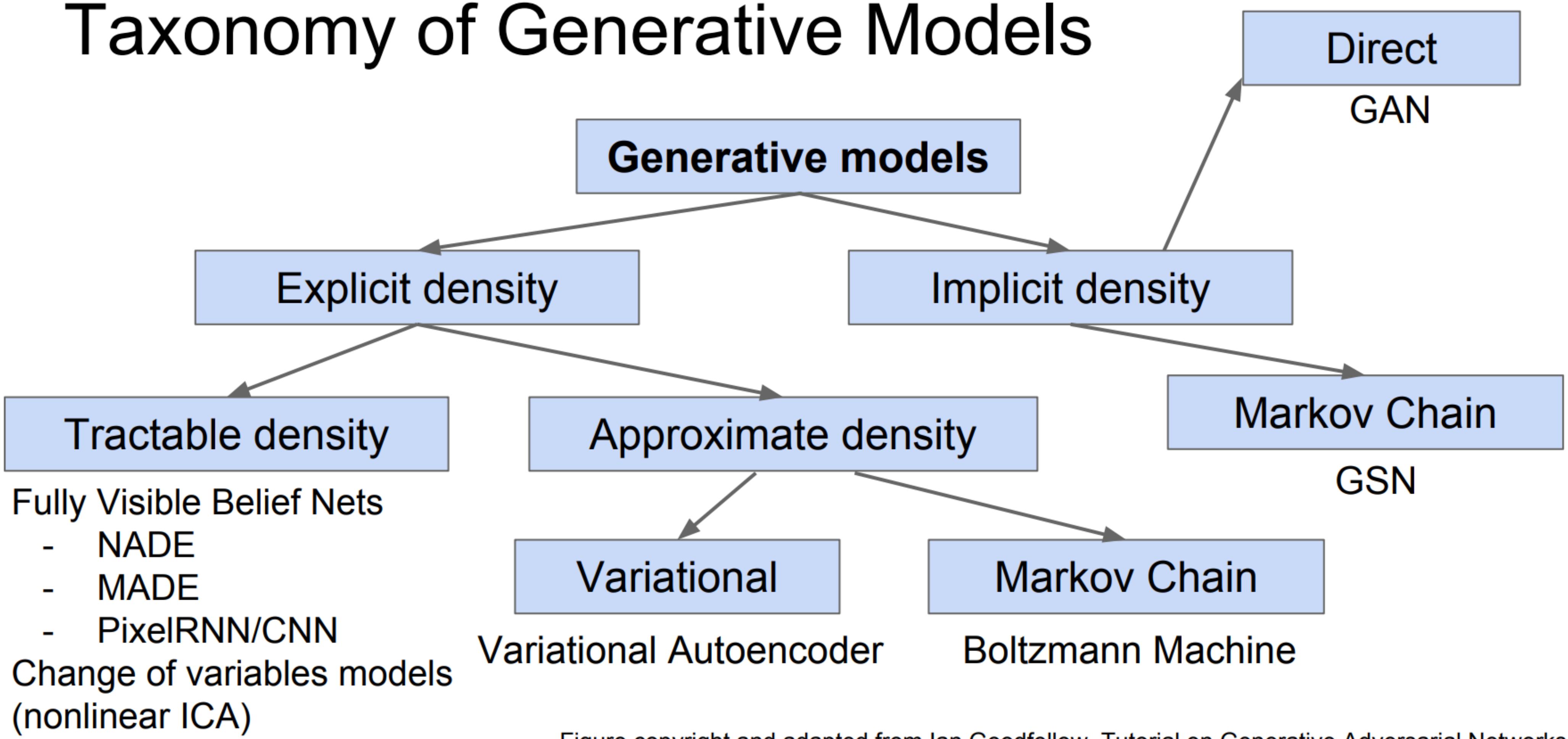


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Stanford CS231n, Lecture 12

Taxonomy of Generative Models

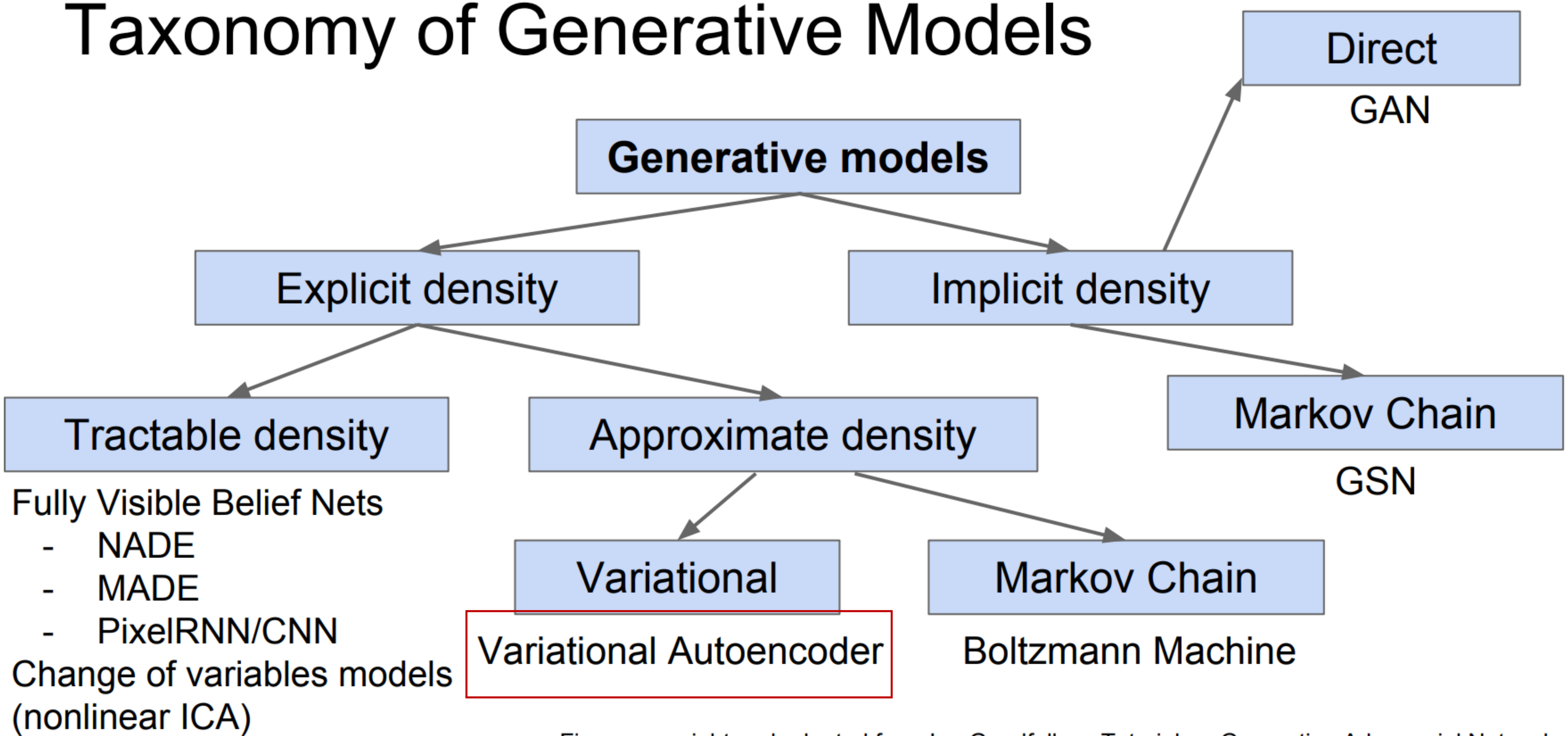
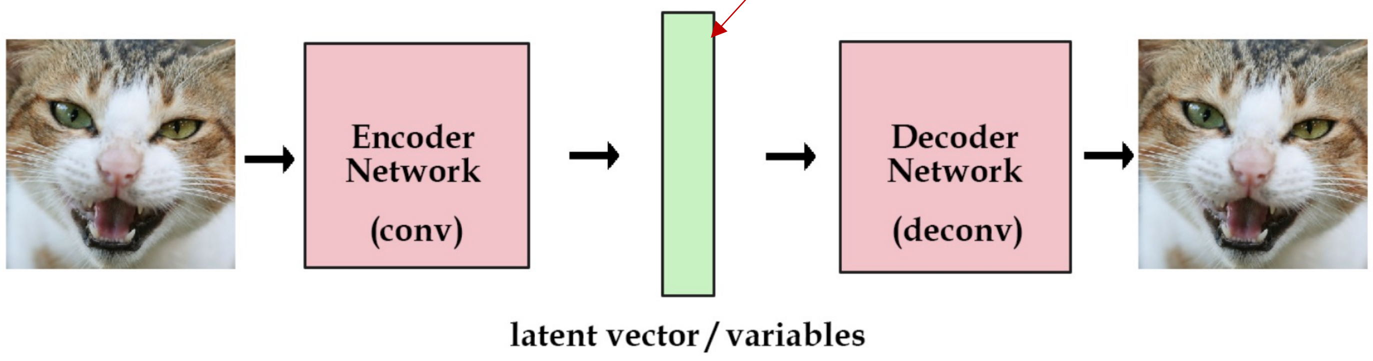


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Stanford CS231n, Lecture 12

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

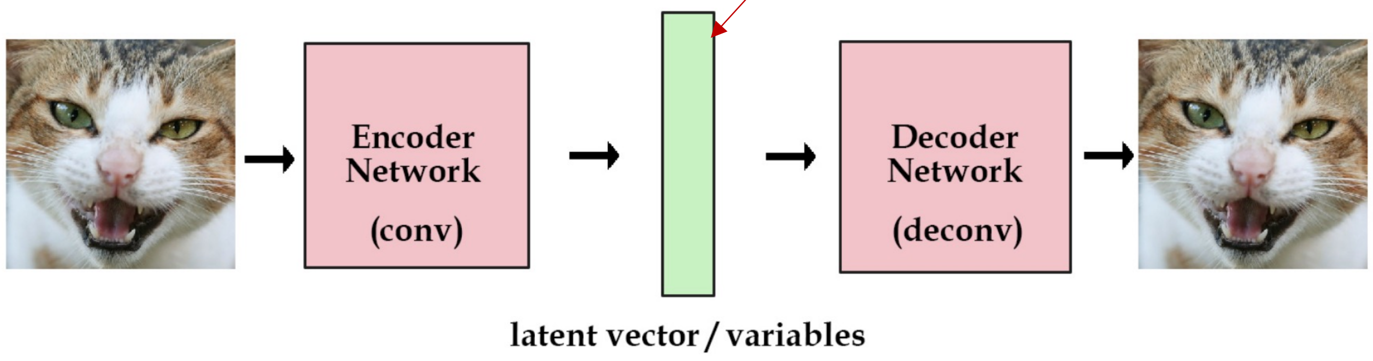


- Good *generative model*
- Have a clean probability distribution to select from to generate new examples

Minimize (KL) distance between latent vector and Gaussian normal

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

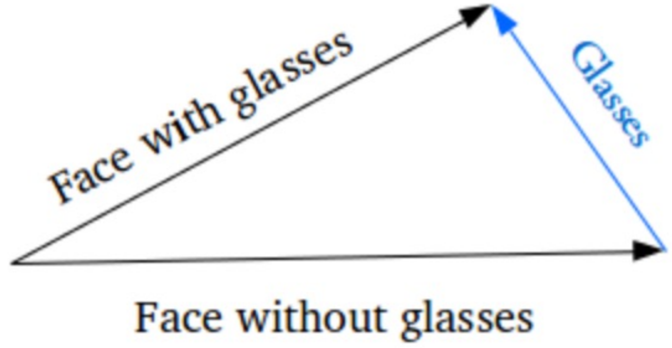


Minimize (KL) distance between latent vector and Gaussian normal

Generative Example (once trained):

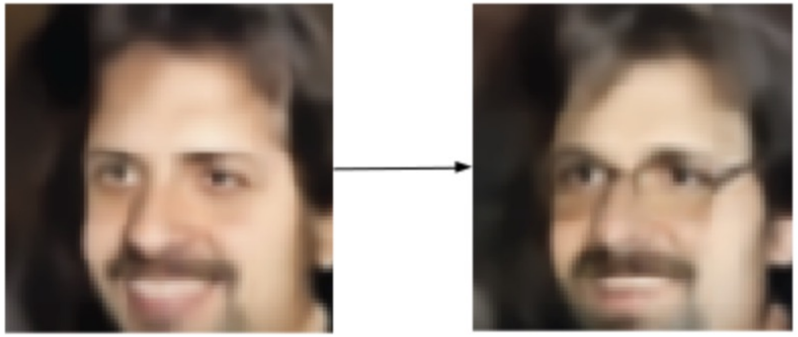
- Encode image with glasses, obtain latent vector PDF $\mathbf{P_g}$
- Encode image without glasses, obtain PDF $\mathbf{P_{ng}}$
- Compute **diff** = $\mathbf{P_g}$ - $\mathbf{P_{ng}}$
- Encode new image to obtain $\mathbf{P_{new}}$, add in **diff**
- Decode $\mathbf{P_{new}}$ + **diff** to get guy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

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



Adding new features to samples

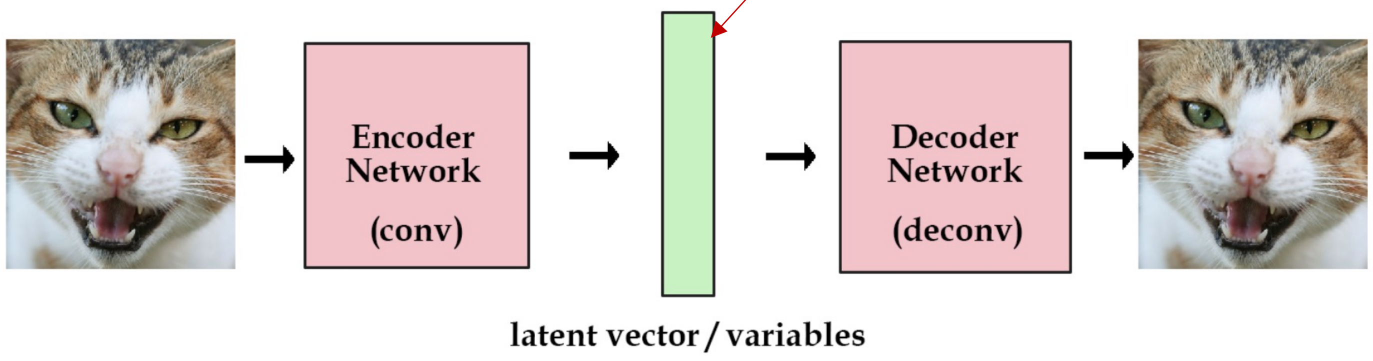
Glasses

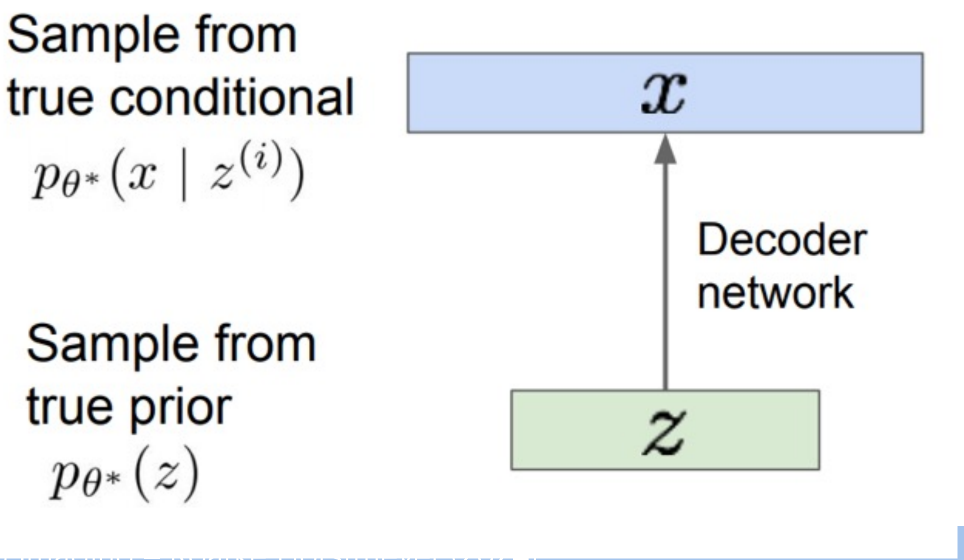


Exploring a specific variation of input data[1]

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF





A very simple example – MNIST digits

See https://deepimaging.github.io/data/Simple_Autoencoder.ipynb

On the way towards generating "fake" images of handwritten digits?

Taxonomy of Generative Models

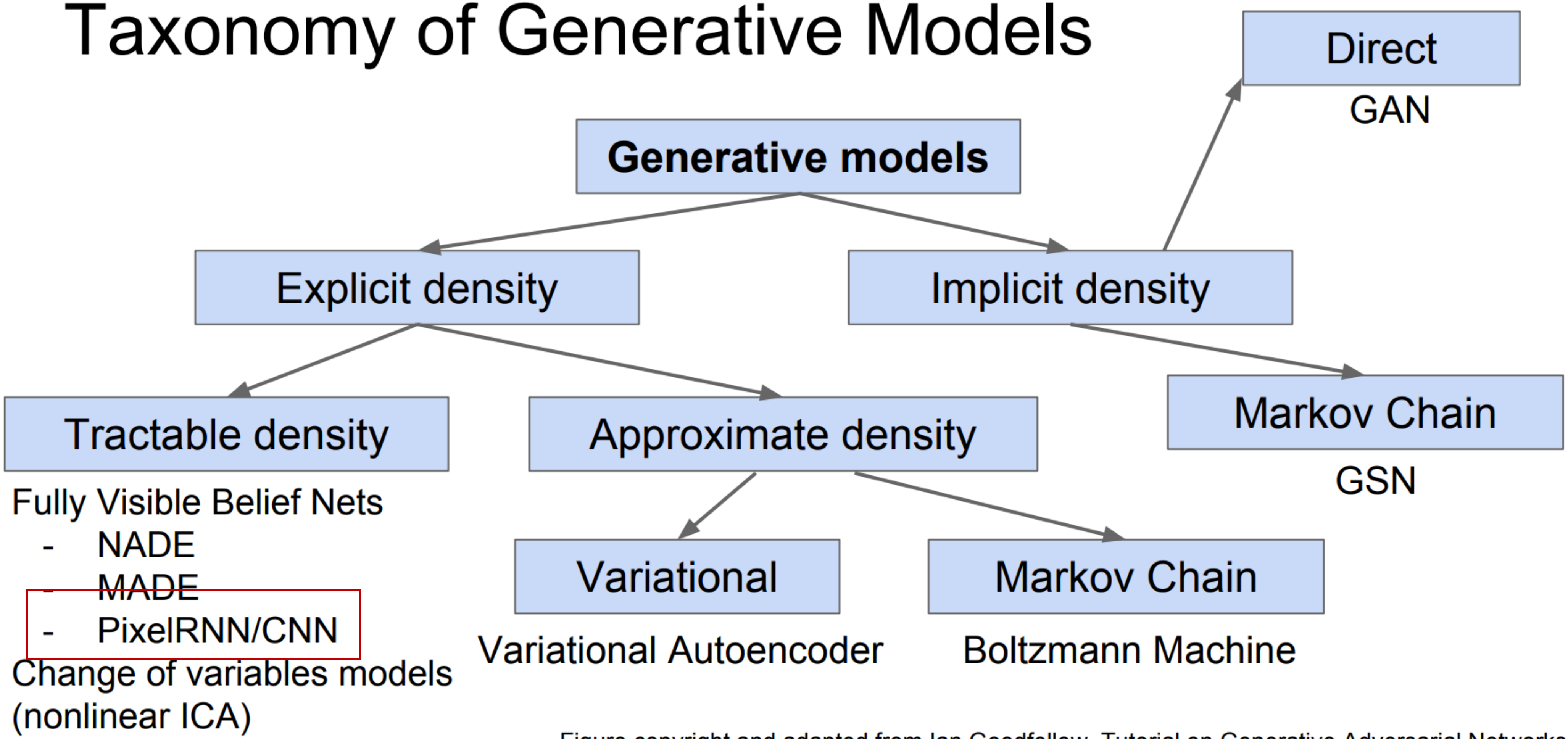


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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:

$$\underbrace{p(x)}_{\text{Likelihood of image x}} = \prod_{i=1}^{n} \underbrace{p(x_i | x_1, ..., x_{i-1})}_{\text{Probability of i'th pixel value given all previous pixels}}$$

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:

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

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

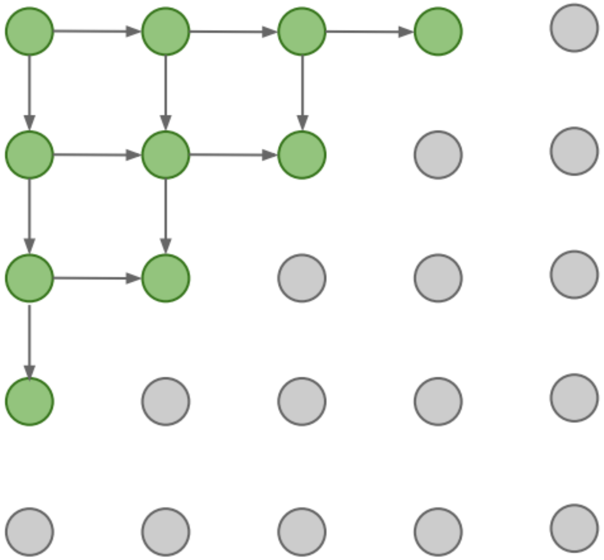
Then maximize likelihood of training data

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!

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]*

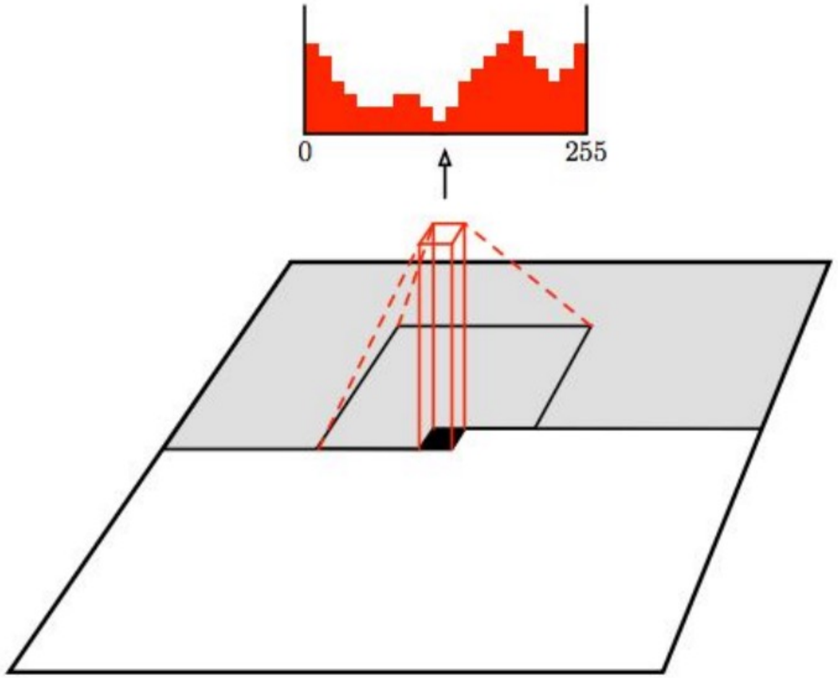
Still generate image pixels starting from corner

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

Training: maximize likelihood of training images

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

Softmax loss at each pixel

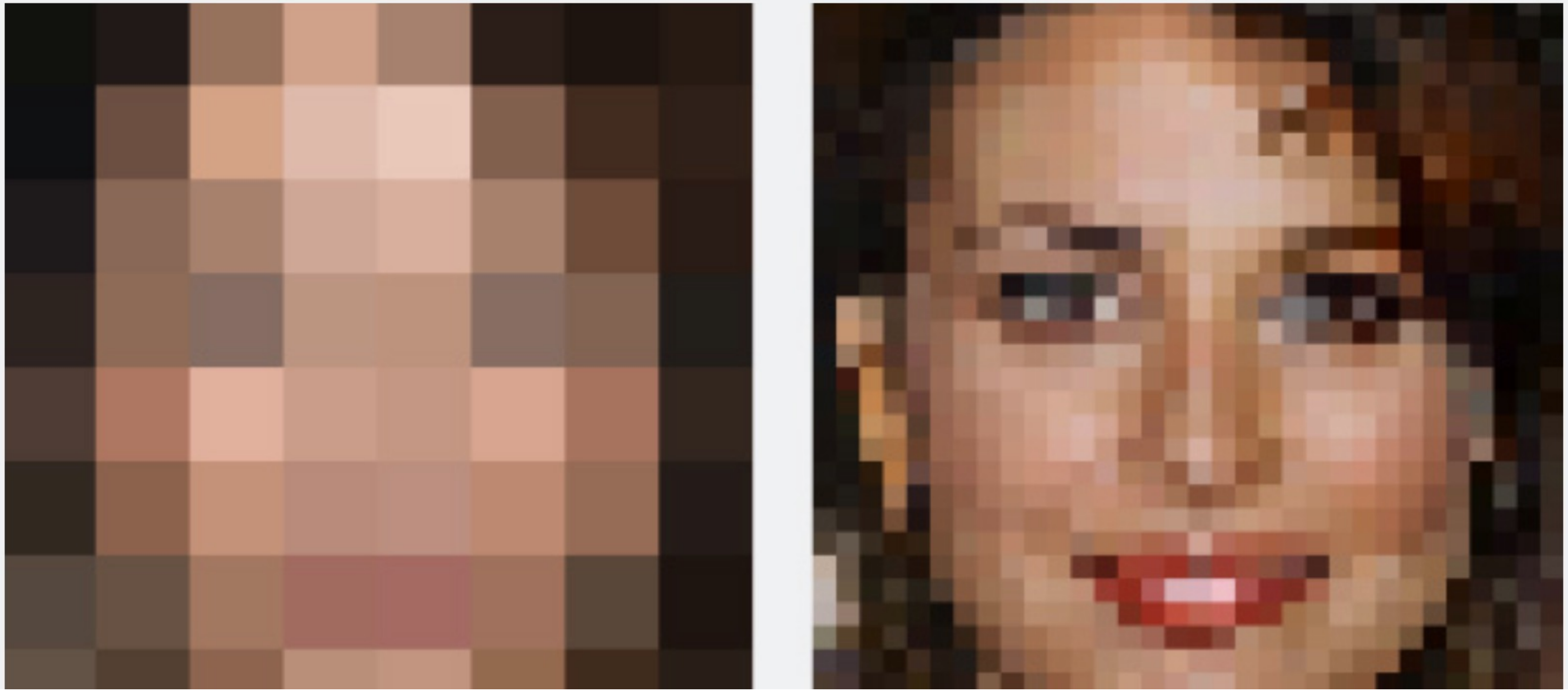


GARBAGE IN, NON-GARBAGE OUT —

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

deep imaging

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) ⇨

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

1. Rephrasing what we want:

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

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

?

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

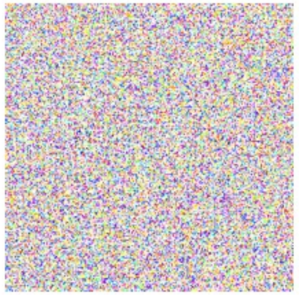
[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

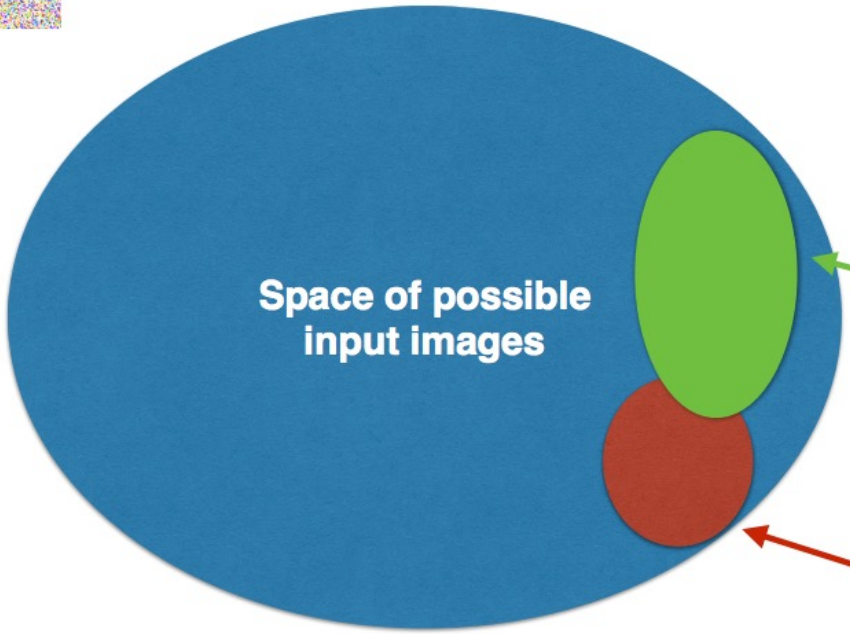
I. A. Attacking a network with adversarial examples

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

deep imaging

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)

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

1. Rephrasing what we want:

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

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

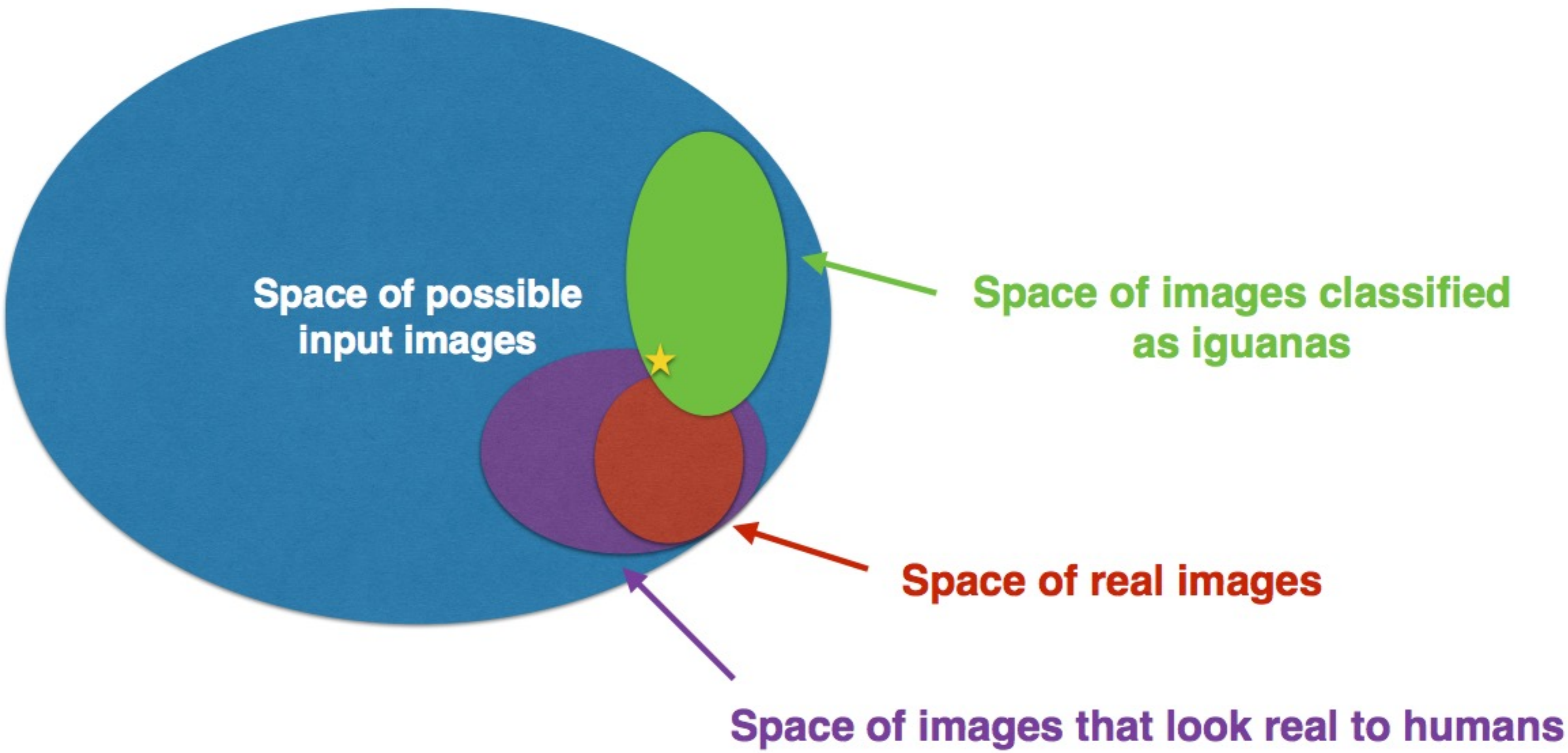
$$\frac{\partial L}{\partial x} \qquad x = x - \alpha \frac{\partial L}{\partial x}$$

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

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

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

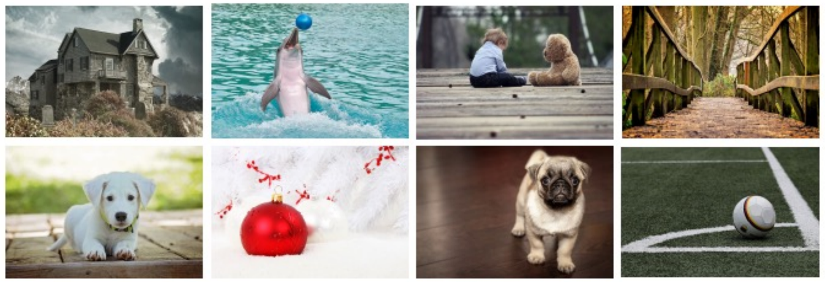
[Lu et al. (2017): SafetyNet: Detecting and Rejecting Adversarial Examples Robustly]
[Harini Kannan et al. (2018): Adversarial Logit Pairing]

deep imaging

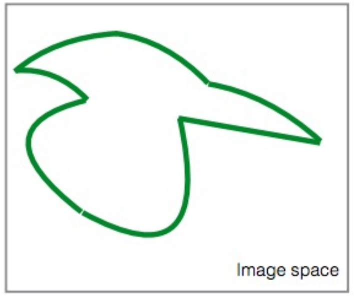
II.A - Motivation

Probability distributions:

Samples from the "real data distribution"

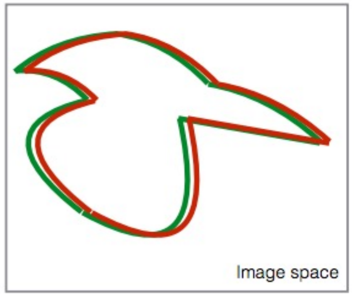


"real data distribution"

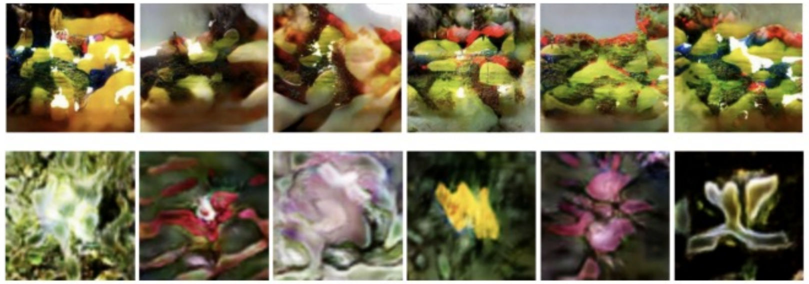


Goal

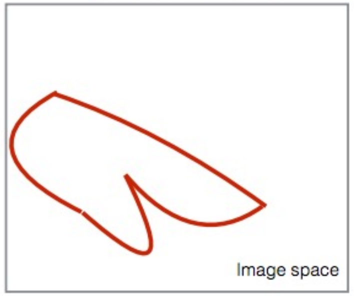
Matching distributions



Samples from the "generated distribution"



"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

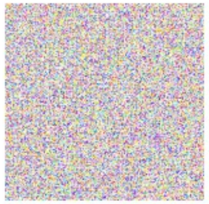
II.B - G/D Game

100-d
random code

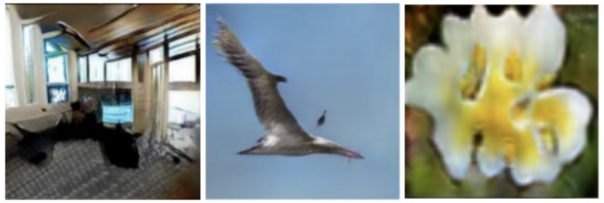
$$z = \begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix}$$

**Generator "G"
(Neural Network)**

(64,64,3)
generated image



$\neq$



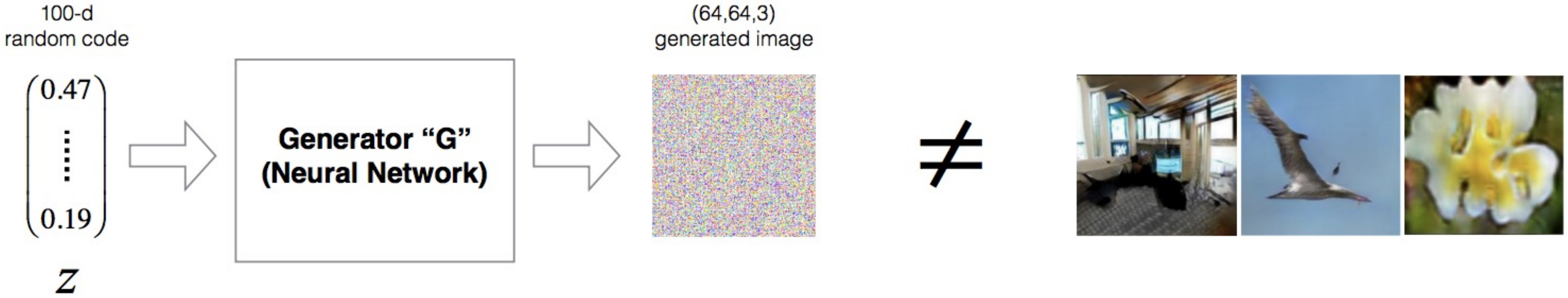
How can we train G to generate images from the true data distributions?

[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]

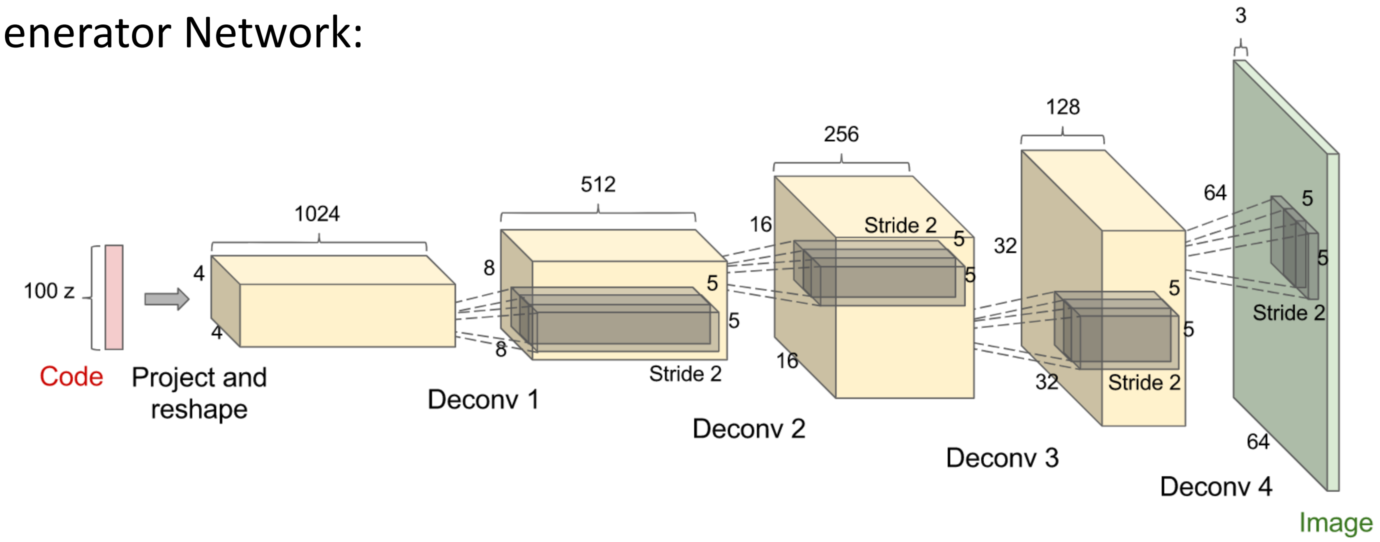
Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

II.B - G/D Game



DCGAN Generator Network:



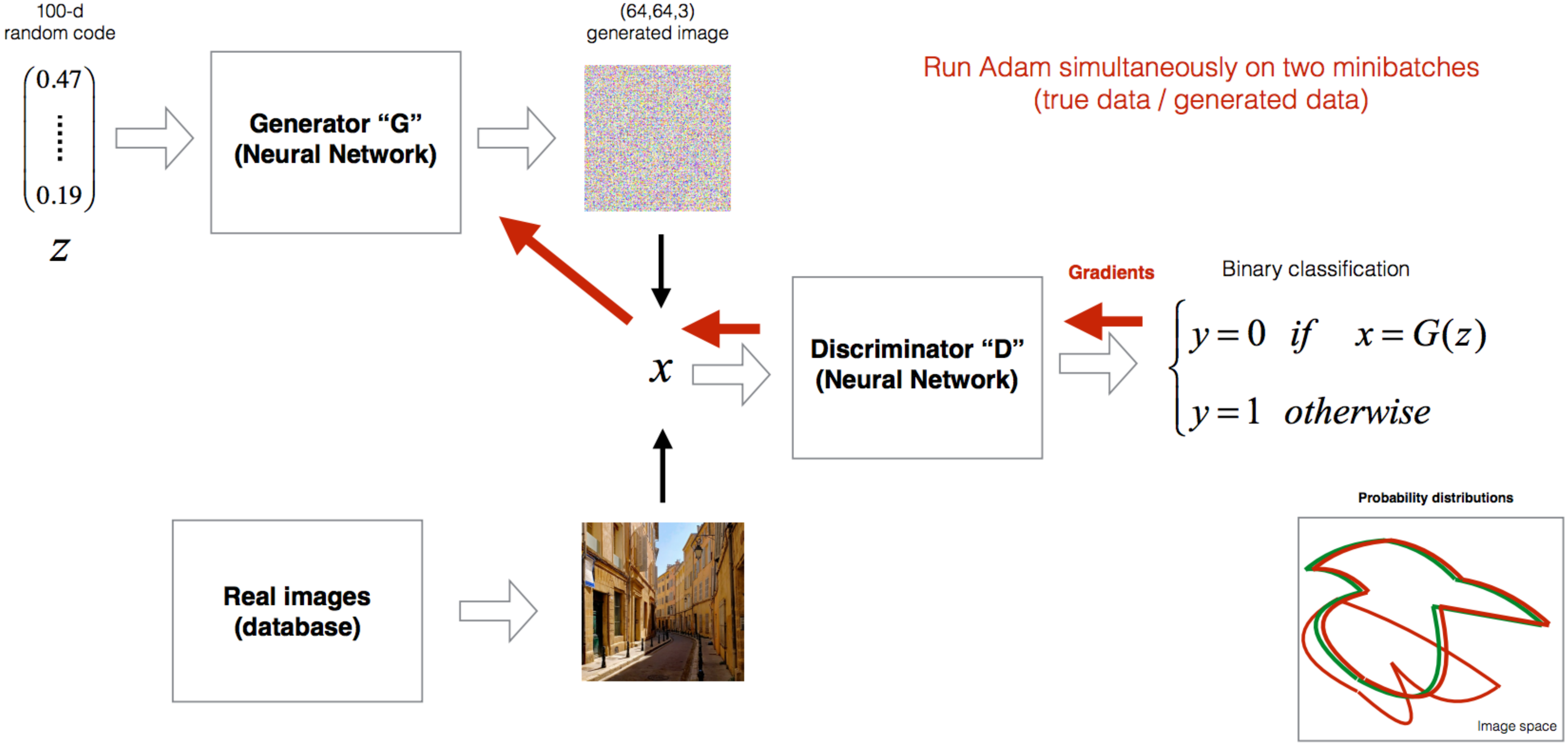
[Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas (2017): StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

II.B - G/D Game

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



Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

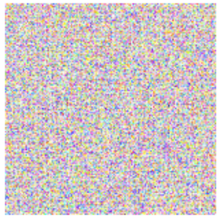
II.B - G/D Game

100-d
random code

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

**Generator "G"
(Neural Network)**

(64,64,3)
generated image



End goal: G is outputting images that are indistinguishable from real images for D

x

**Discriminator "D"
(Neural Network)**

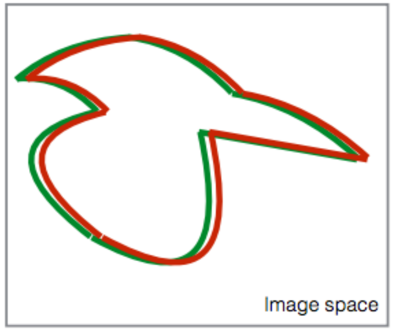
Gradients

Binary classification

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

**Real images
(database)**

Probability distribution



Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

II.B - G/D Game

naging

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

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

- 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"

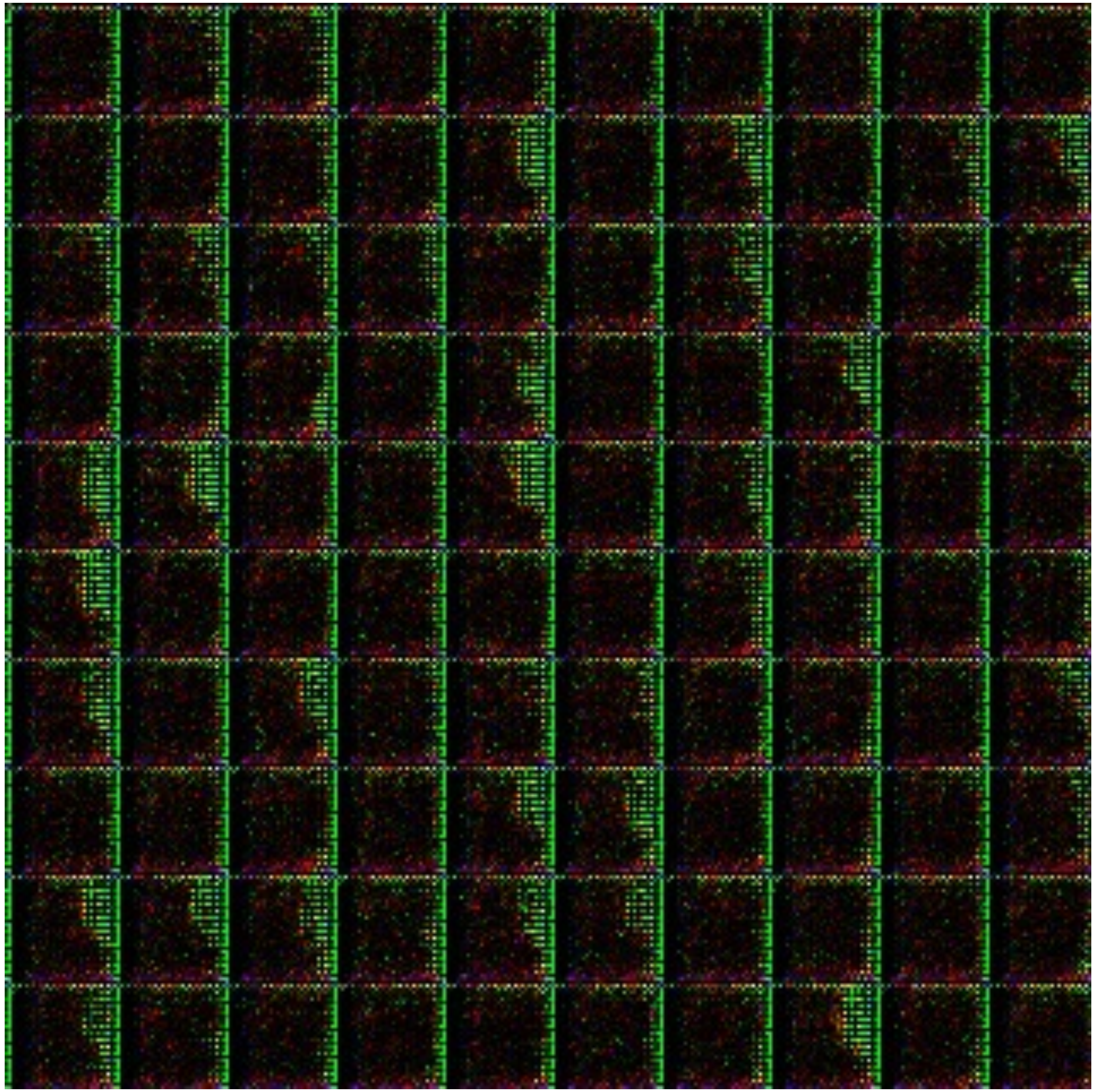
"Maximize probability that the discriminator is wrong and labels the fake example as a real example"

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

Stanford CS230, Lecture 3

Typical # of weights:
100 Million

Typical training dataset:
20GB (e.g., Imagenet)



II. D. In terms of code

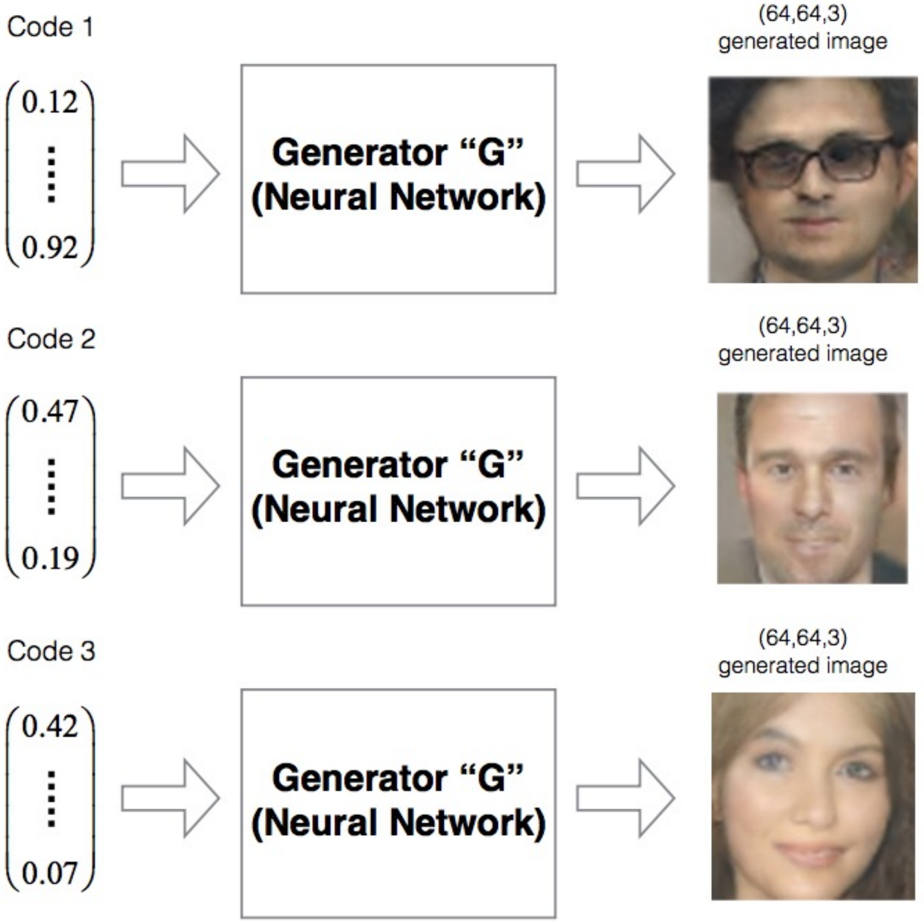
```python
# Build and compile the discriminator
self.discriminator = self.build_discriminator()
self.discriminator.compile(loss='binary_crossentropy',
    optimizer=optimizer,
    metrics=['accuracy'])

# 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)
```

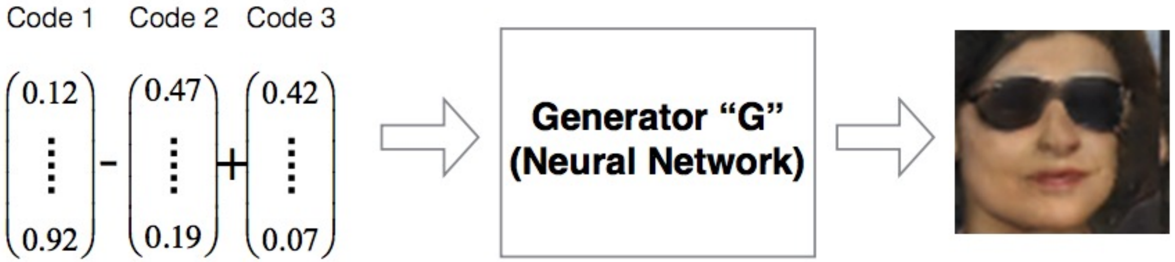
```python
def build_discriminator(self):

    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]

II.E - Nice results

Operation on codes





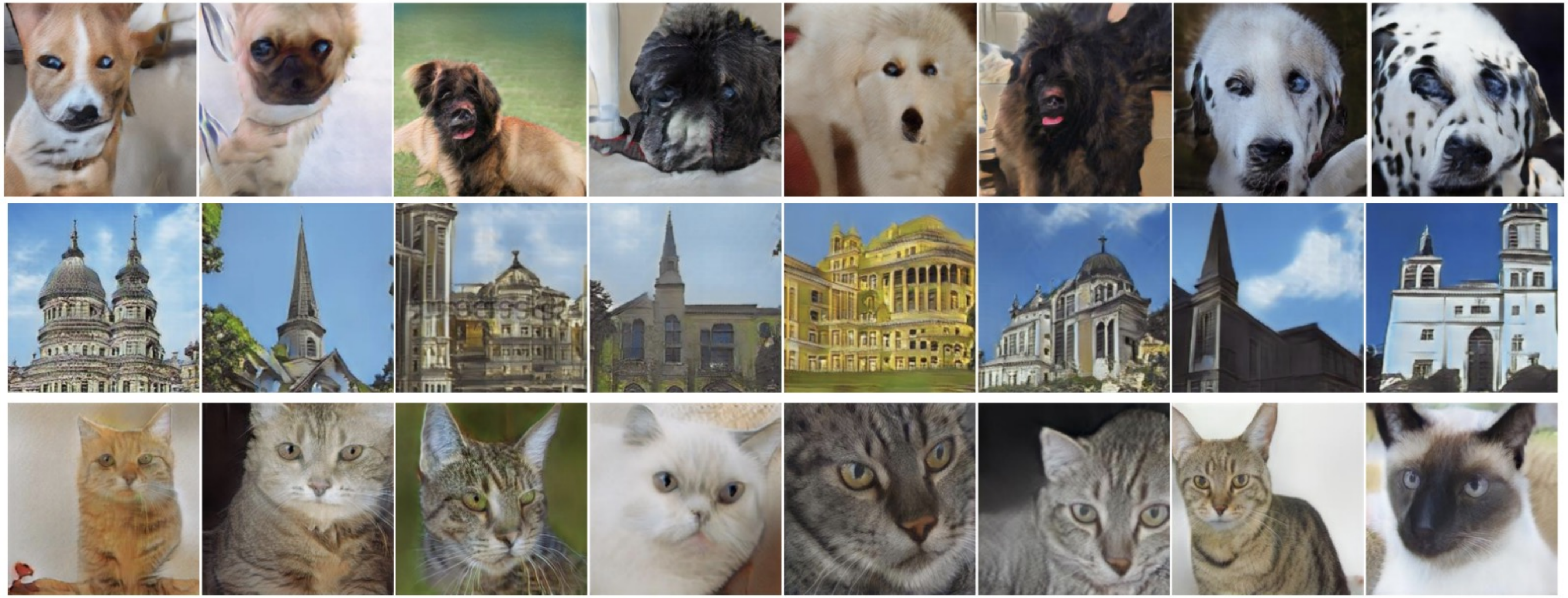
Man with glasses - man + woman = woman with glasses

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

II.E - Nice results

Image Generation:

Samples from the "generated distribution"



[Zhang et al. (2017): StackGAN++]

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?"

<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:

No Attack | FastSign Attack | Iterative Attack | LBFGS Attack

4 different adversarial examples for object classifier:

No Attack | FastSign Attack | Iterative Attack | LBFGS Attack

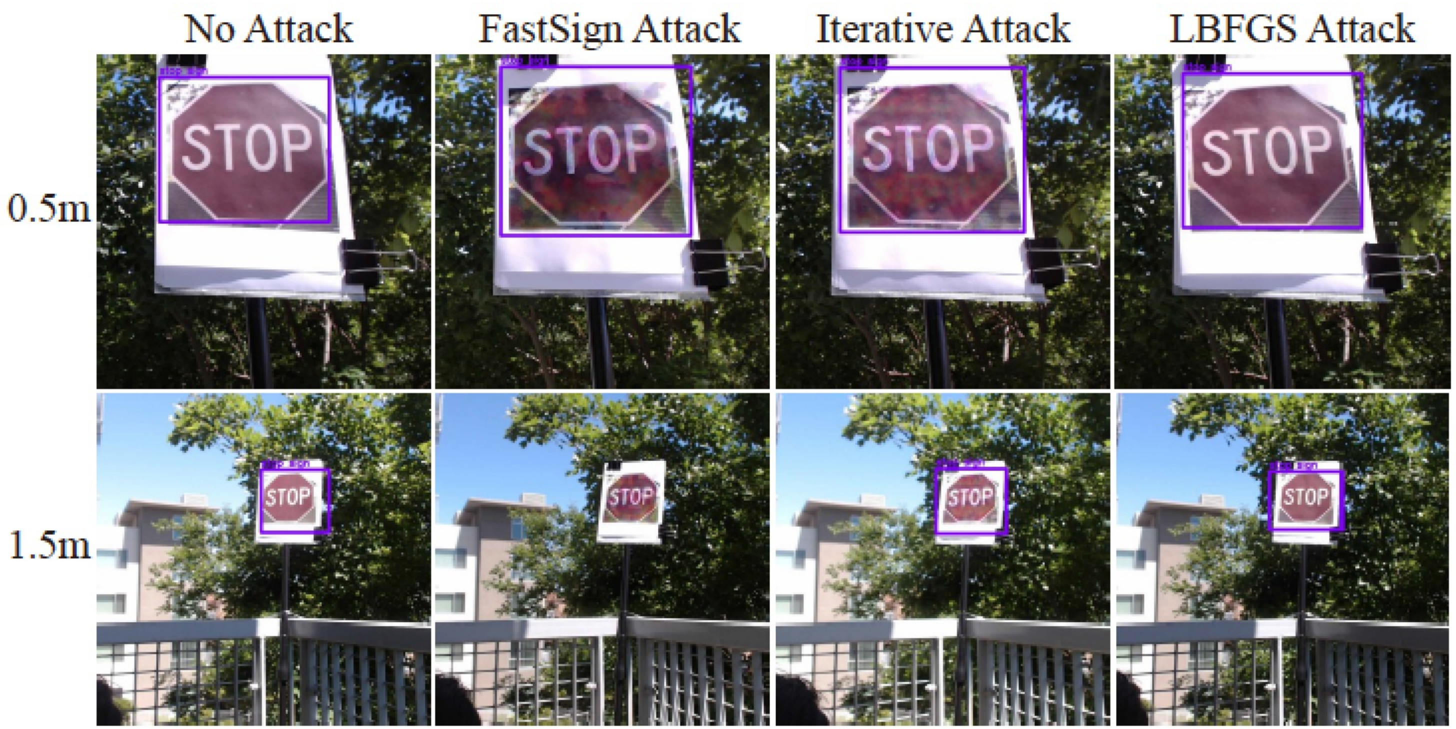


Figure 3: This figure shows experiment setup, and we use the printed stop signs to simulate real stop signs with natural background. These are examples for successful 0.5 meters and 1.5 meters detection: both original images and adversarial examples are detected in both distances. It demonstrates that 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/