

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

Generative models, adversarial examples, and more

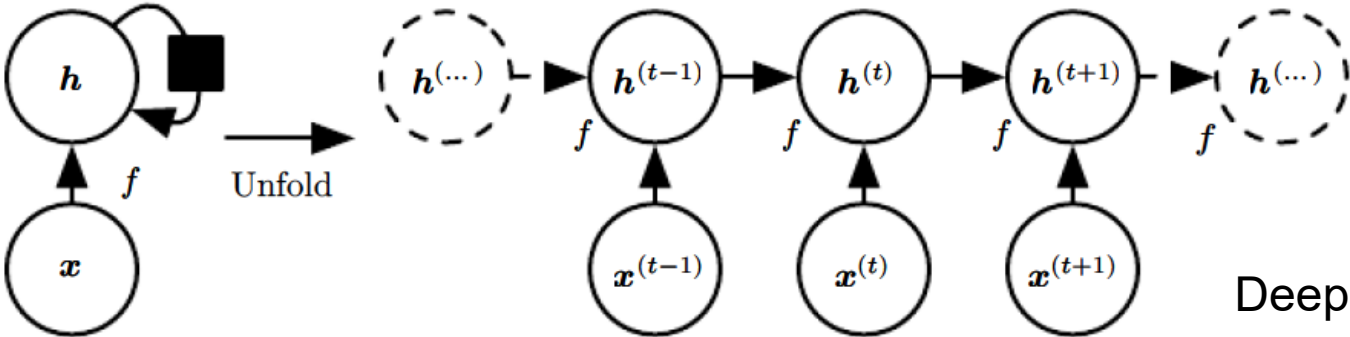
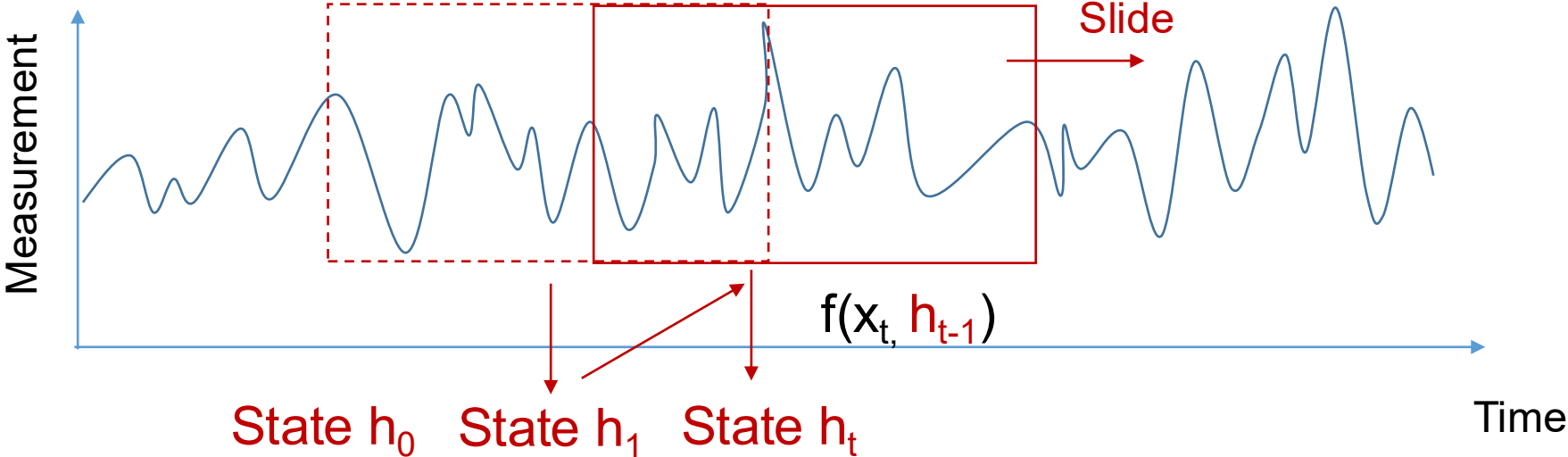
Resources for this lecture:

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

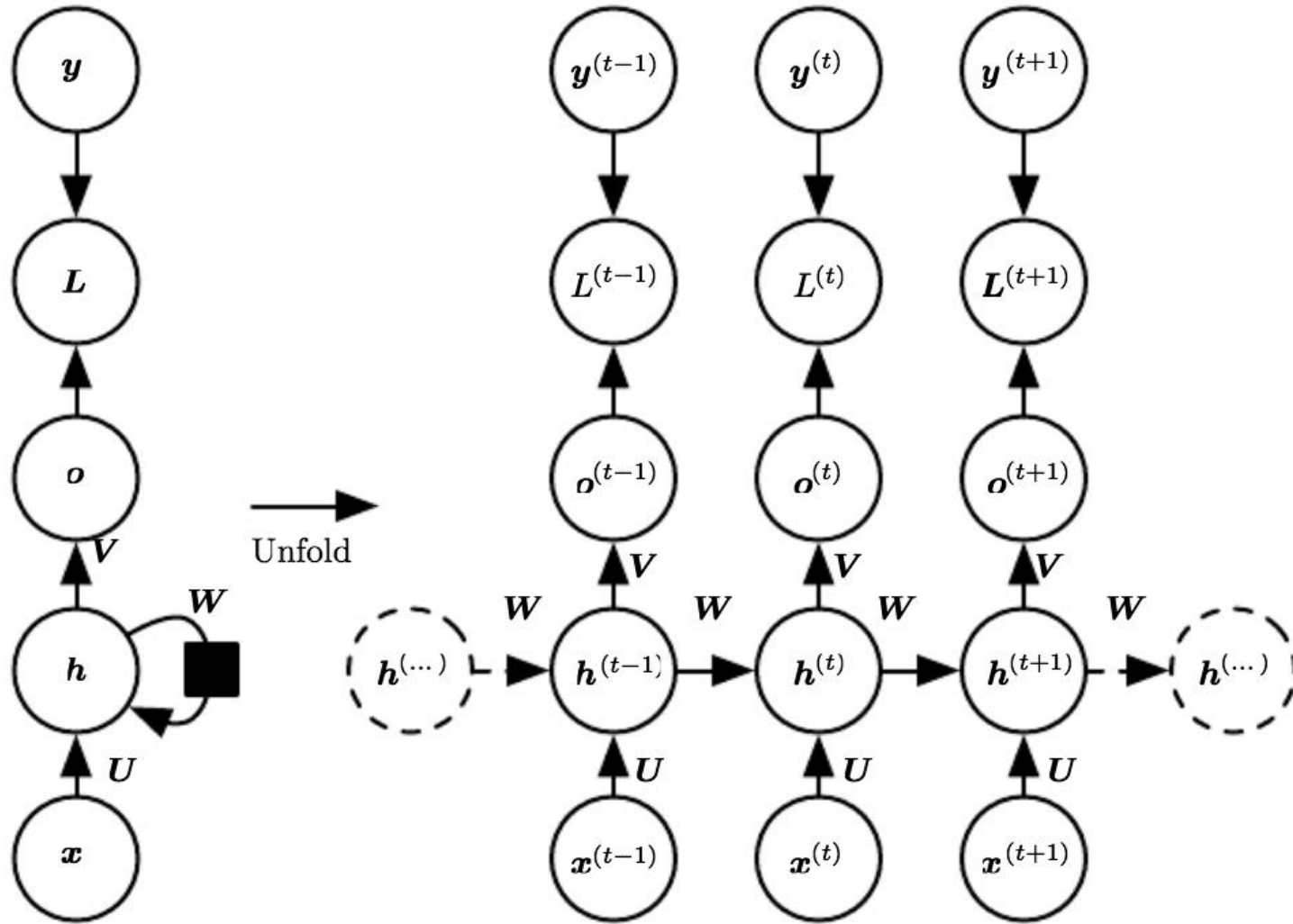
Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

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



Many-to-many recurrent neural network



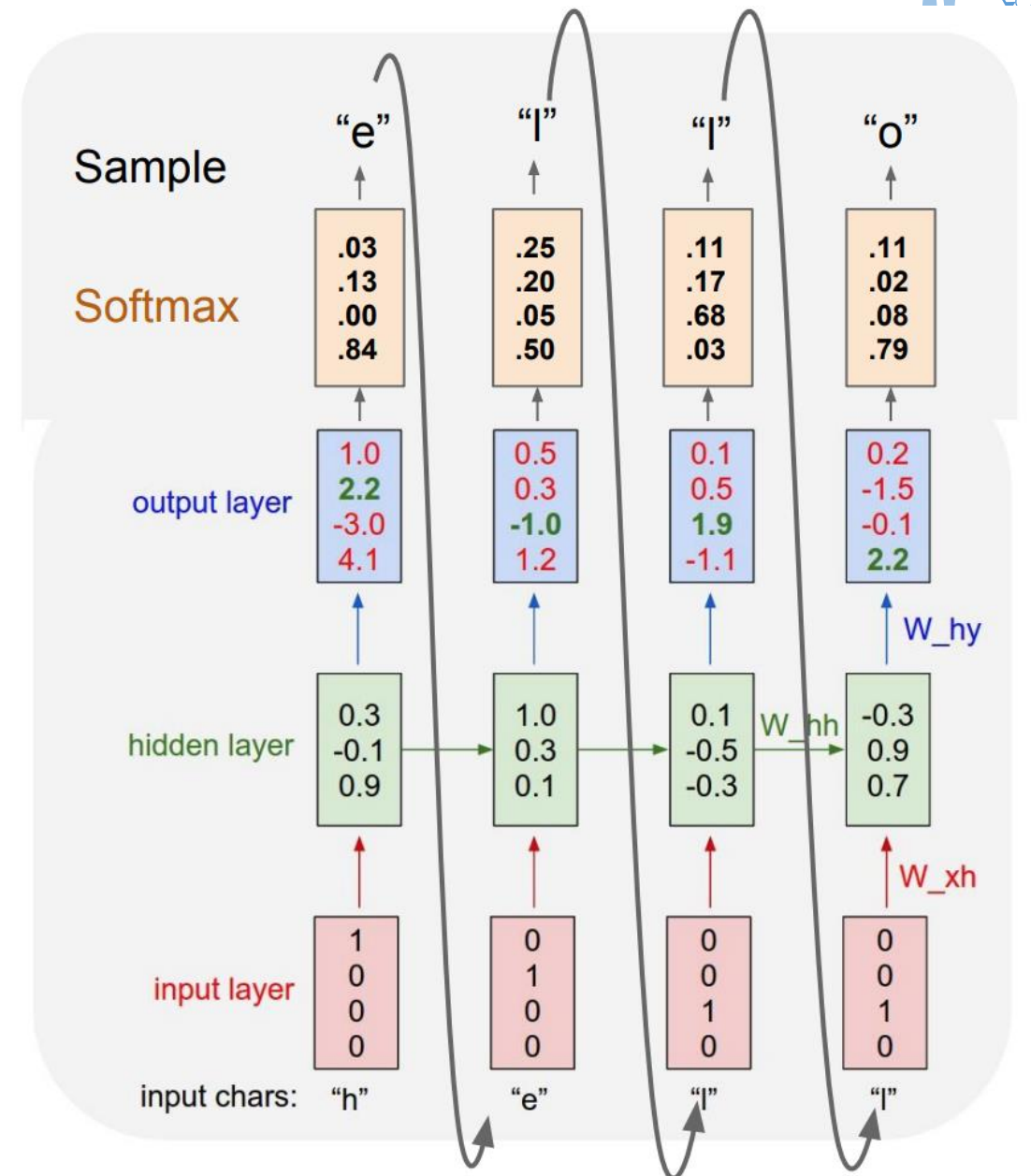
$$\begin{aligned} \mathbf{a}^{(t)} &= \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}, \\ \mathbf{h}^{(t)} &= \tanh(\mathbf{a}^{(t)}), \\ \mathbf{o}^{(t)} &= \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}, \\ \hat{\mathbf{y}}^{(t)} &= \text{softmax}(\mathbf{o}^{(t)}), \end{aligned}$$



Example: Character-level Language Model Sampling

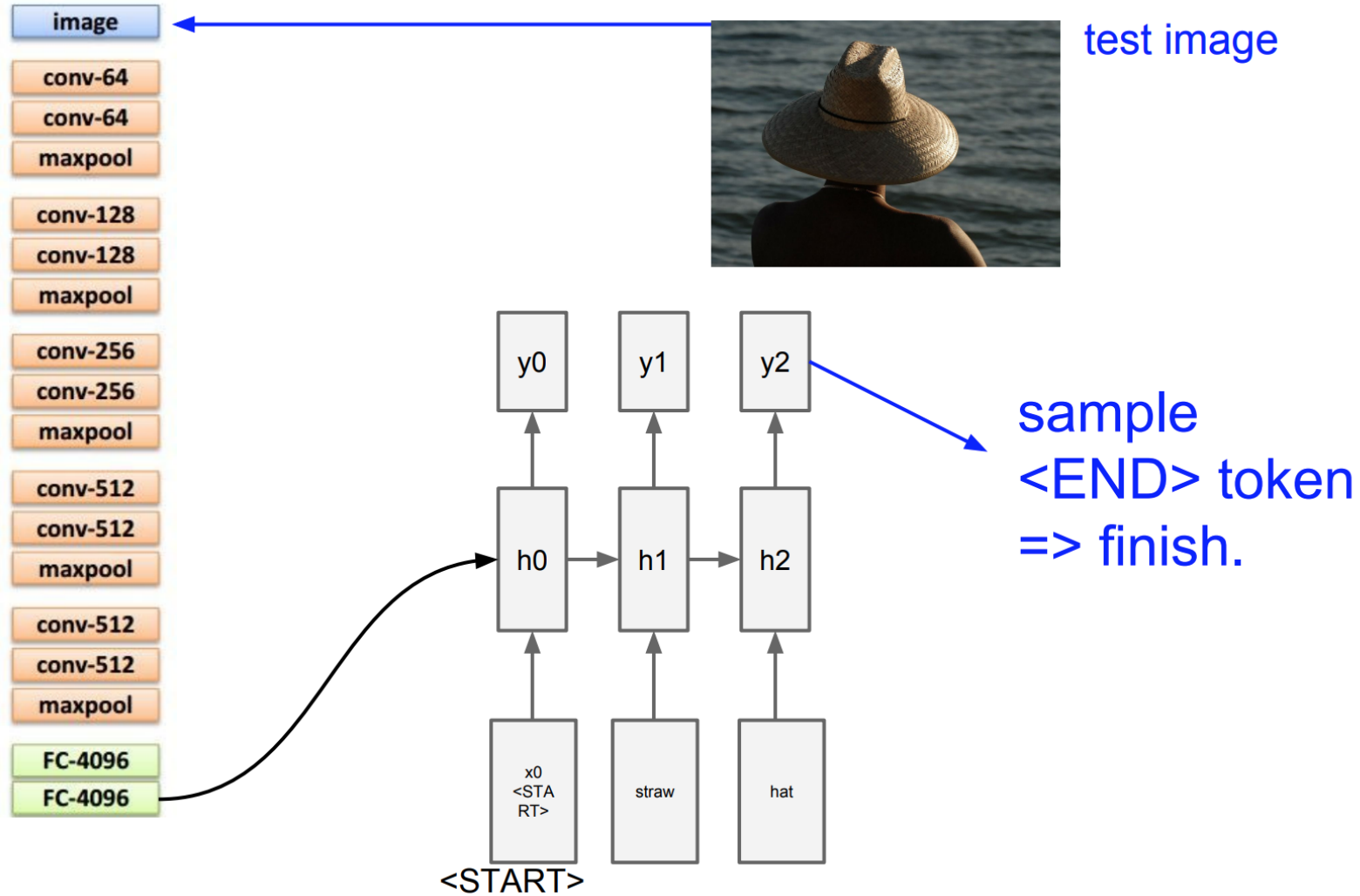
Vocabulary:
[h,e,l,o]

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



From Stanford CS231n Lecture 10 slides

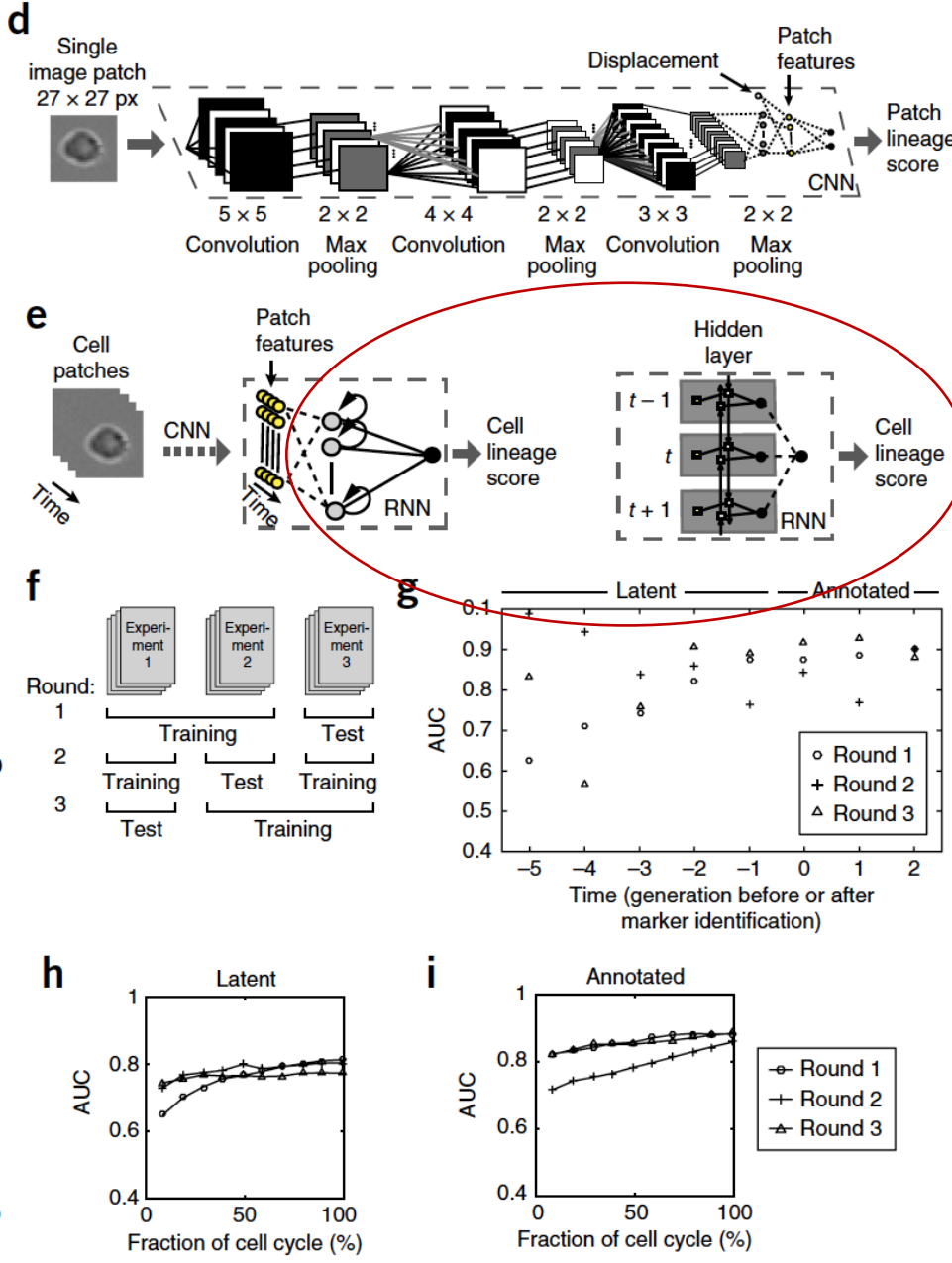
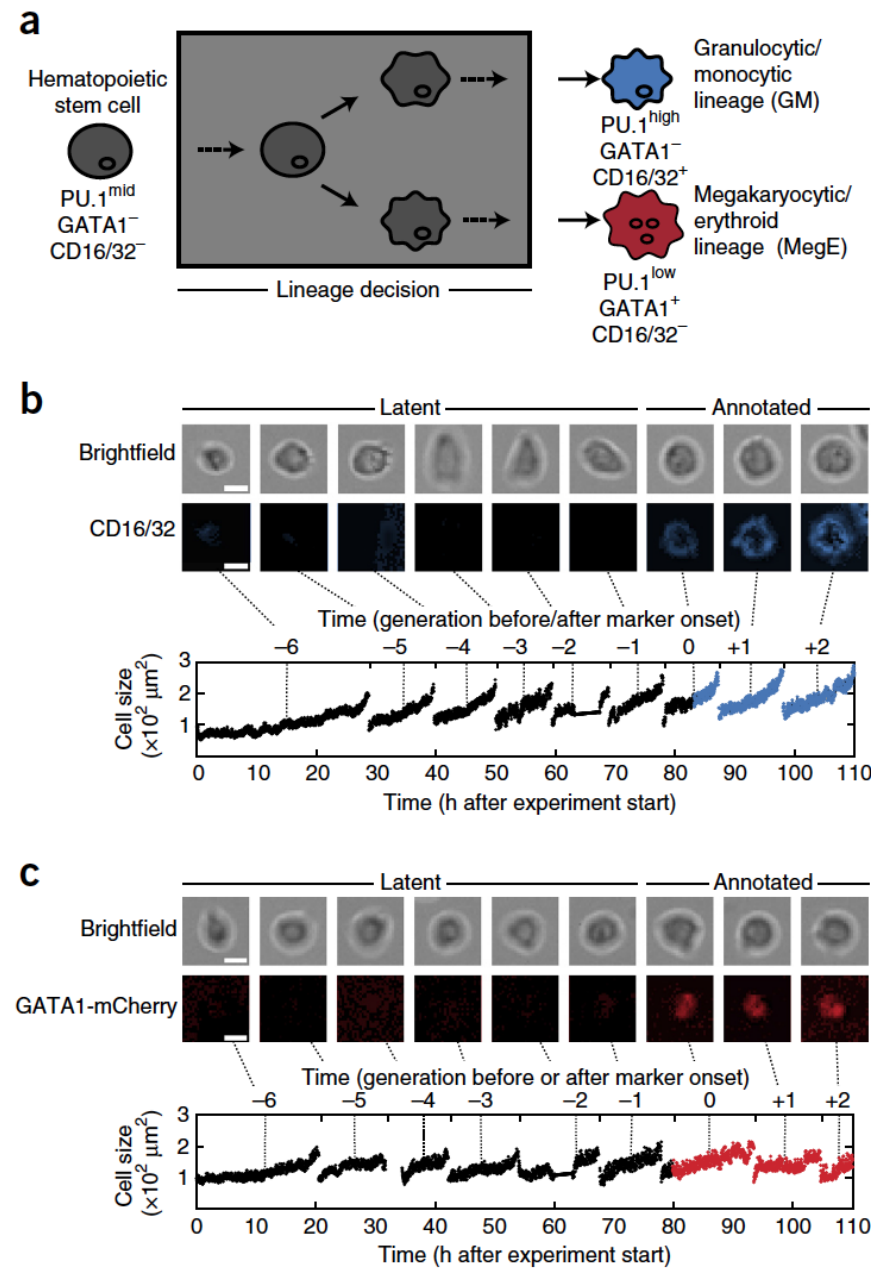
Example: Image captioning





Prospective identification of hematopoietic lineage choice by deep learning

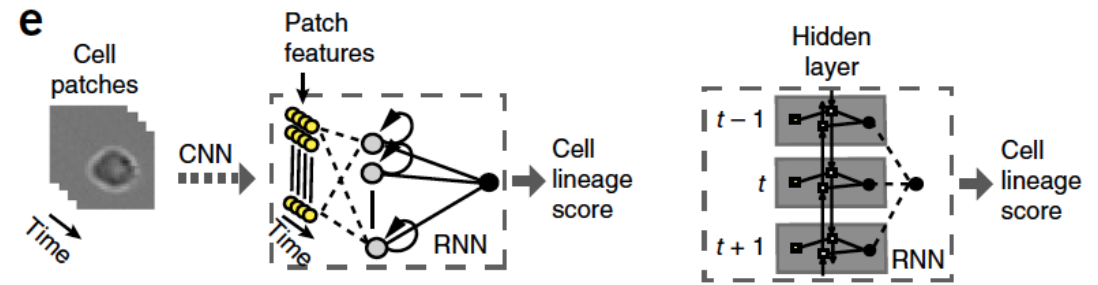
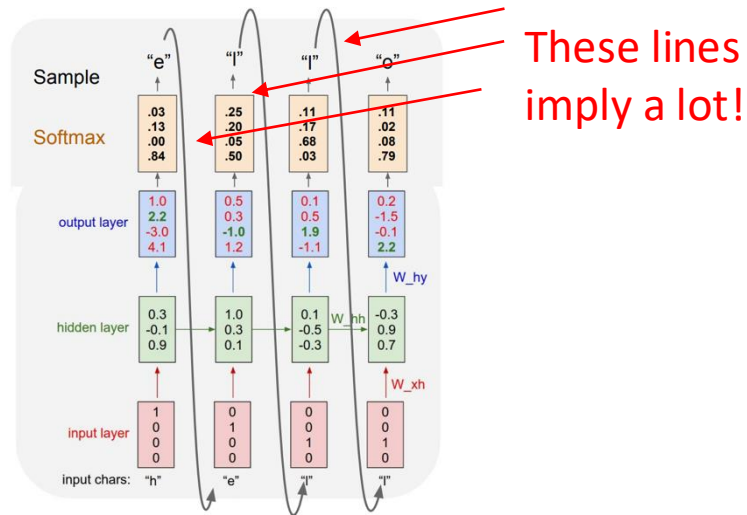
Felix Buggenthin^{1,6}, Florian Buettner^{1,2,6}, Philipp S Hoppe^{3,4}, Max Endeke³, Manuel Kroiss^{1,5}, Michael Strasser¹, Michael Schwarzfischer¹, Dirk Loeffler^{3,4}, Konstantinos D Kokkaliaris^{3,4}, Oliver Hilsenbeck^{3,4}, Timm Schroeder^{3,4}, Fabian J Theis^{1,5} & Carsten Marr¹



Output generative processing - be careful...

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

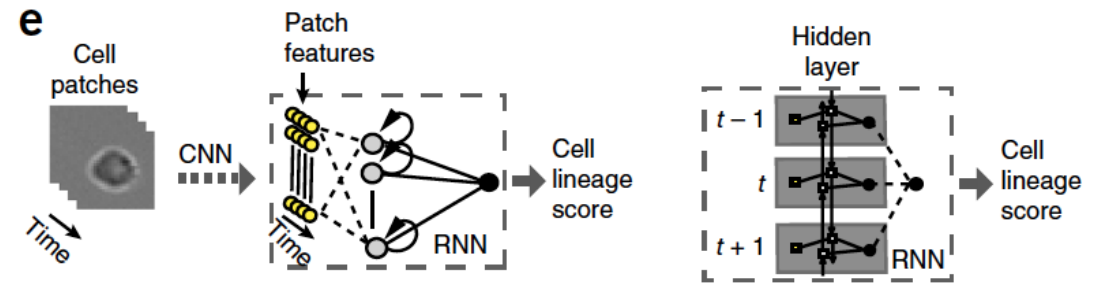
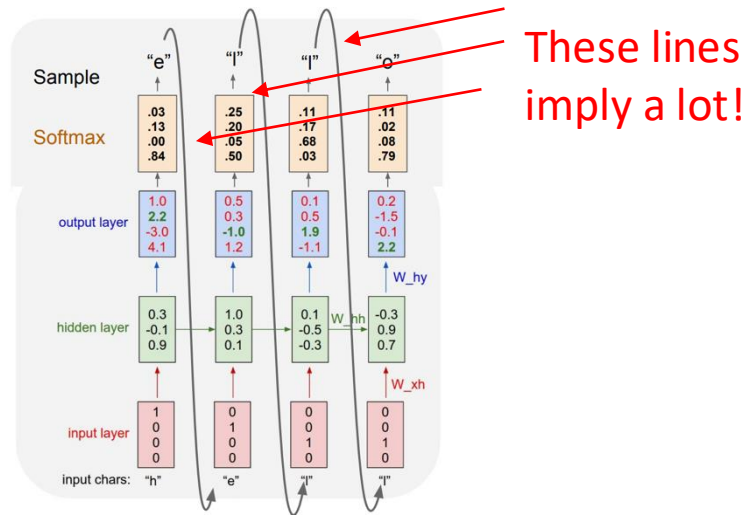
“Non-output generative”: “I’ll just use all of my input data to determine my final output”



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”



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

Pros: More repeatable and interpretable results

Cons: Fine line between hallucination and trustworthy output...

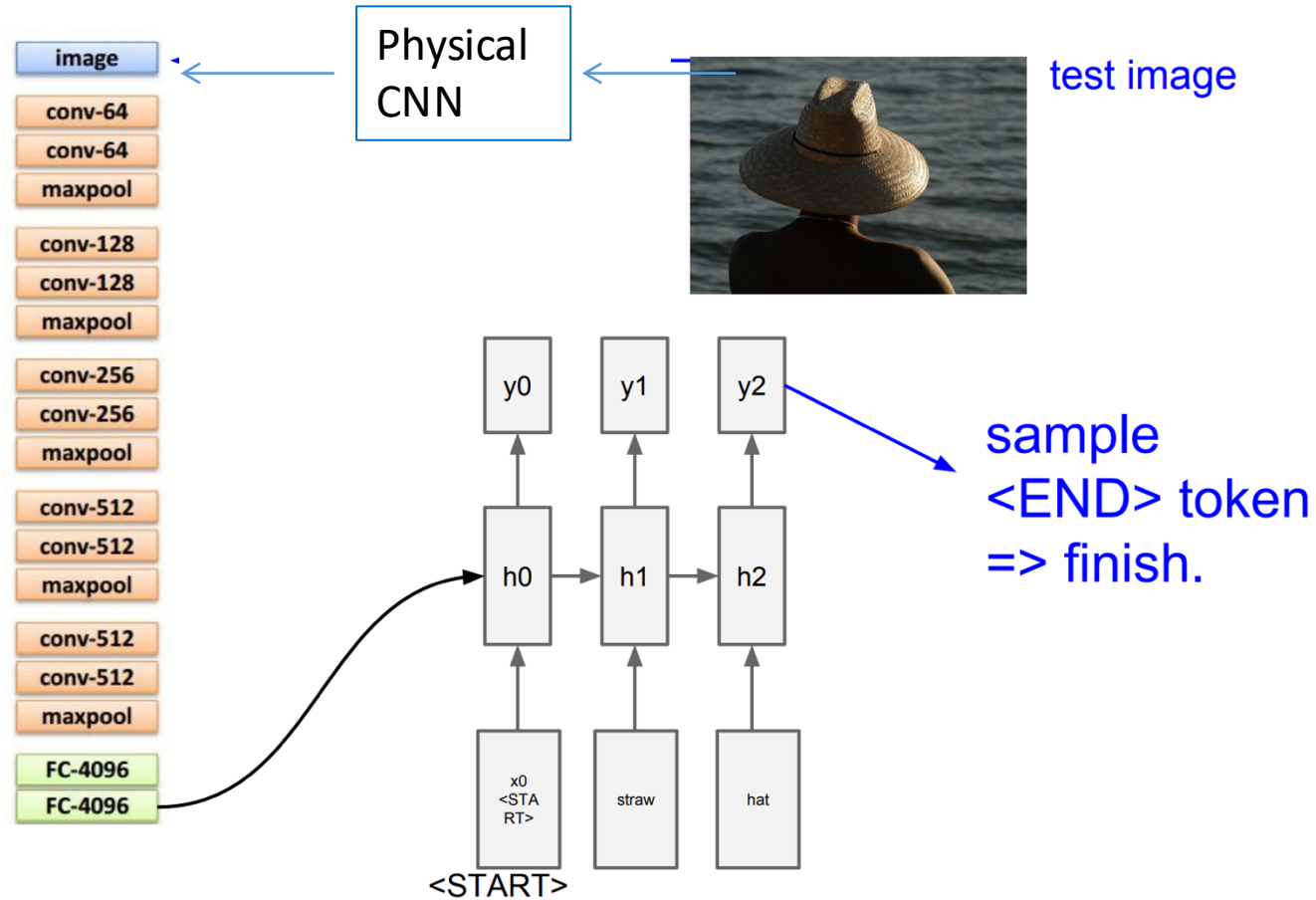
Cons: Hard to get more outputs from inputs (and not as “cool”)

Brainstorming time – physical layers in an RNN???

Brainstorming time – physical layers in an RNN???

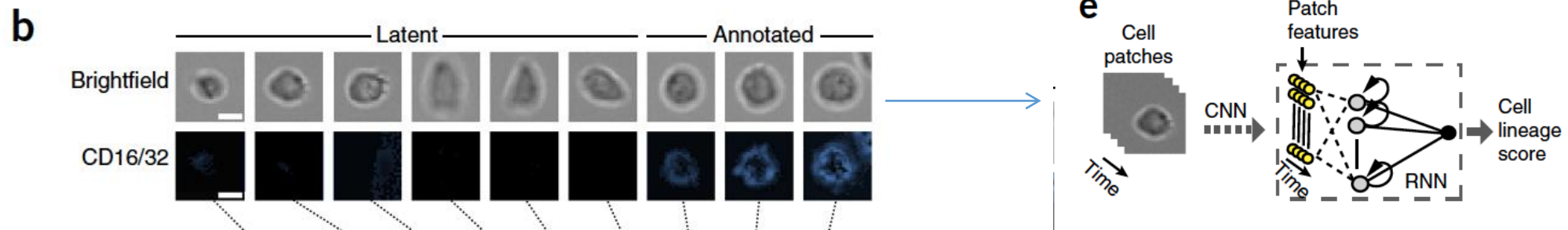
Simple example, “Output generative” flavor

Design an optimal X to produce the best image captions



Brainstorming time – physical layers in an RNN???

Simple example, “Non-output generative” flavor



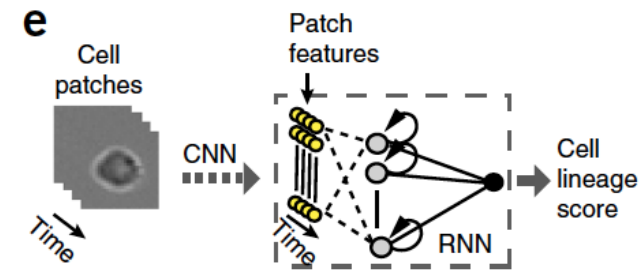
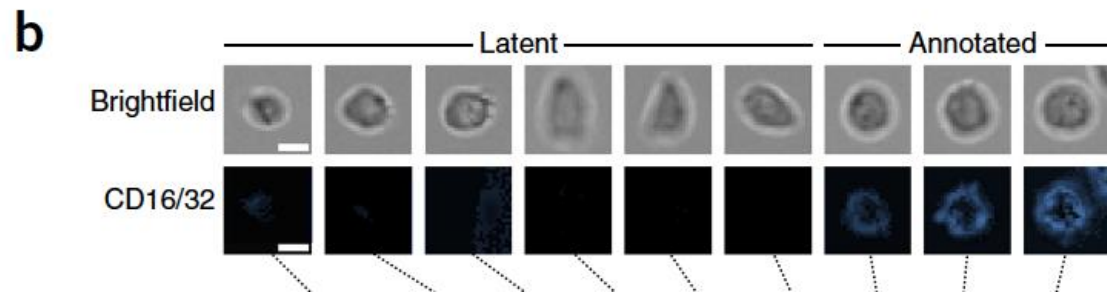
Brainstorming time – physical layers in an RNN???

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

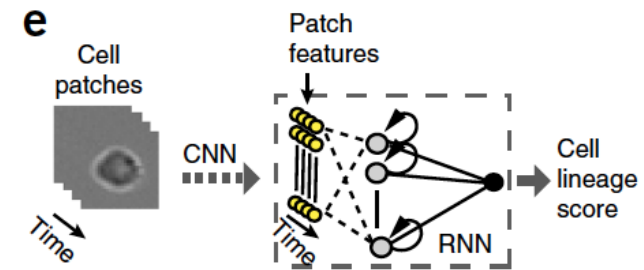
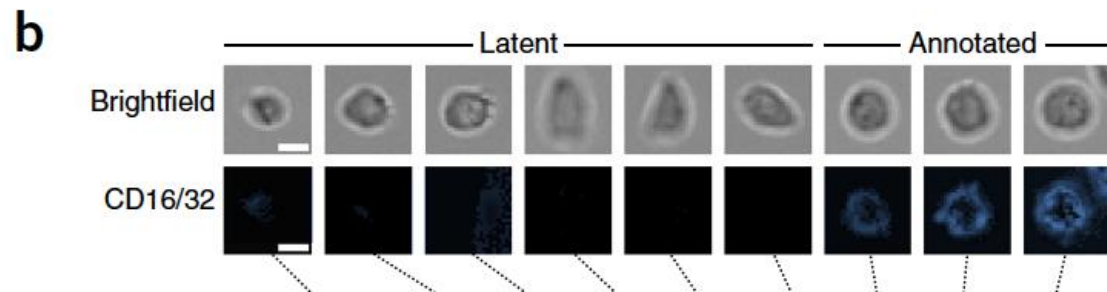
Physical model down-samples movie sequence

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 \mathbf{x} with labels \mathbf{y}
- Goal is to learn function $f(\mathbf{x}) = \mathbf{y}$

Unsupervised

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

Supervised versus unsupervised learning

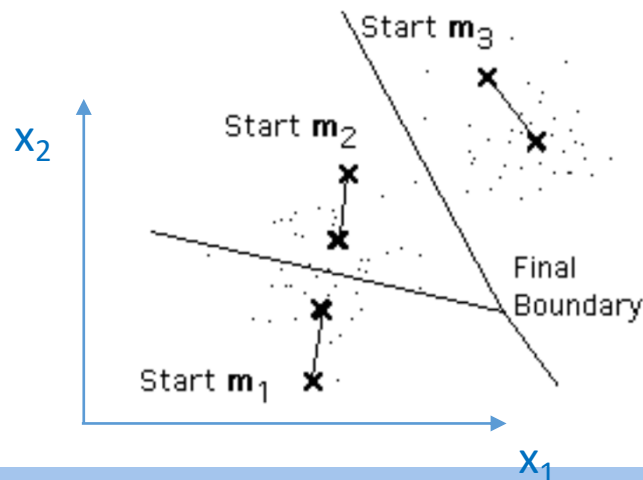
Supervised

- Have data \mathbf{x} with labels \mathbf{y}
- Goal is to learn function $f(\mathbf{x}) = \mathbf{y}$

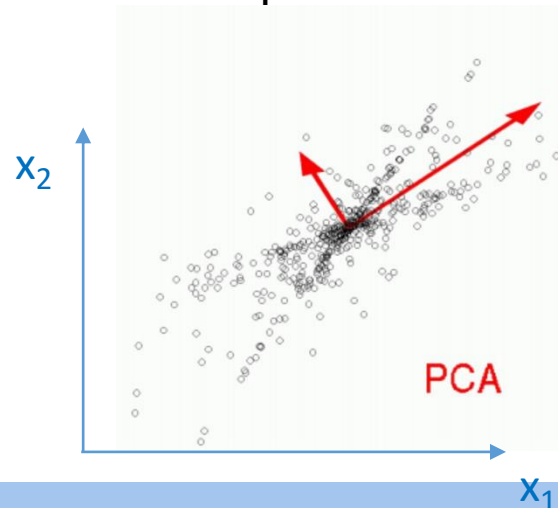
Unsupervised

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

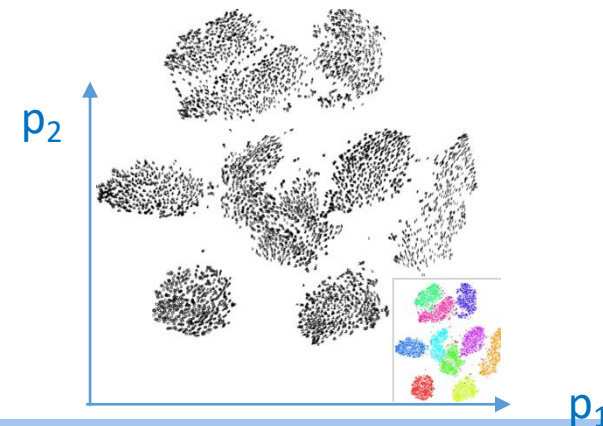
Example: K-means clustering



Example: PCA

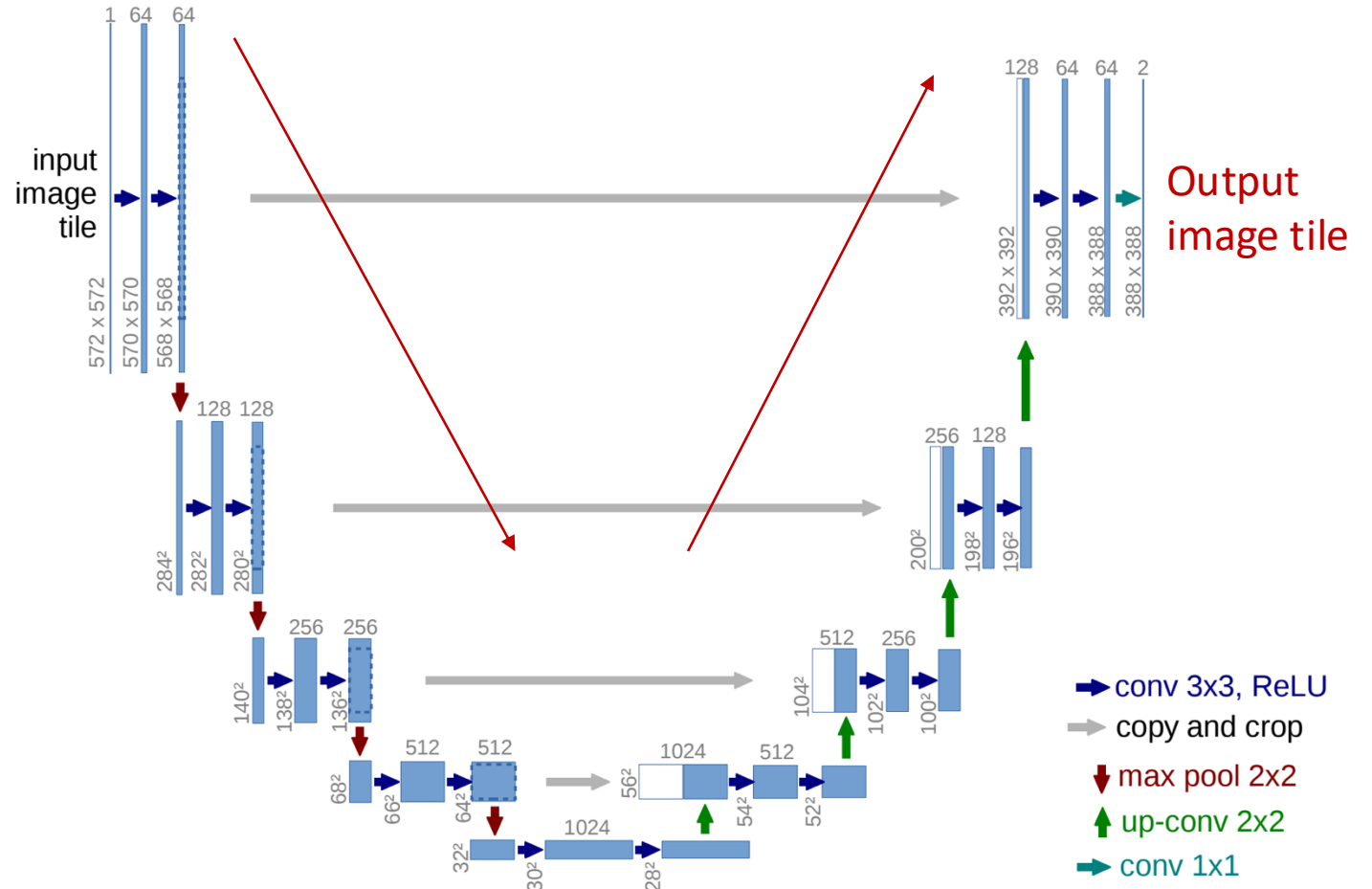


Example: tSNE



Unsupervised learning example: autoencoder

U-Net Architecture



U-Net: Convolutional Networks for Biomedical Image Segmentation

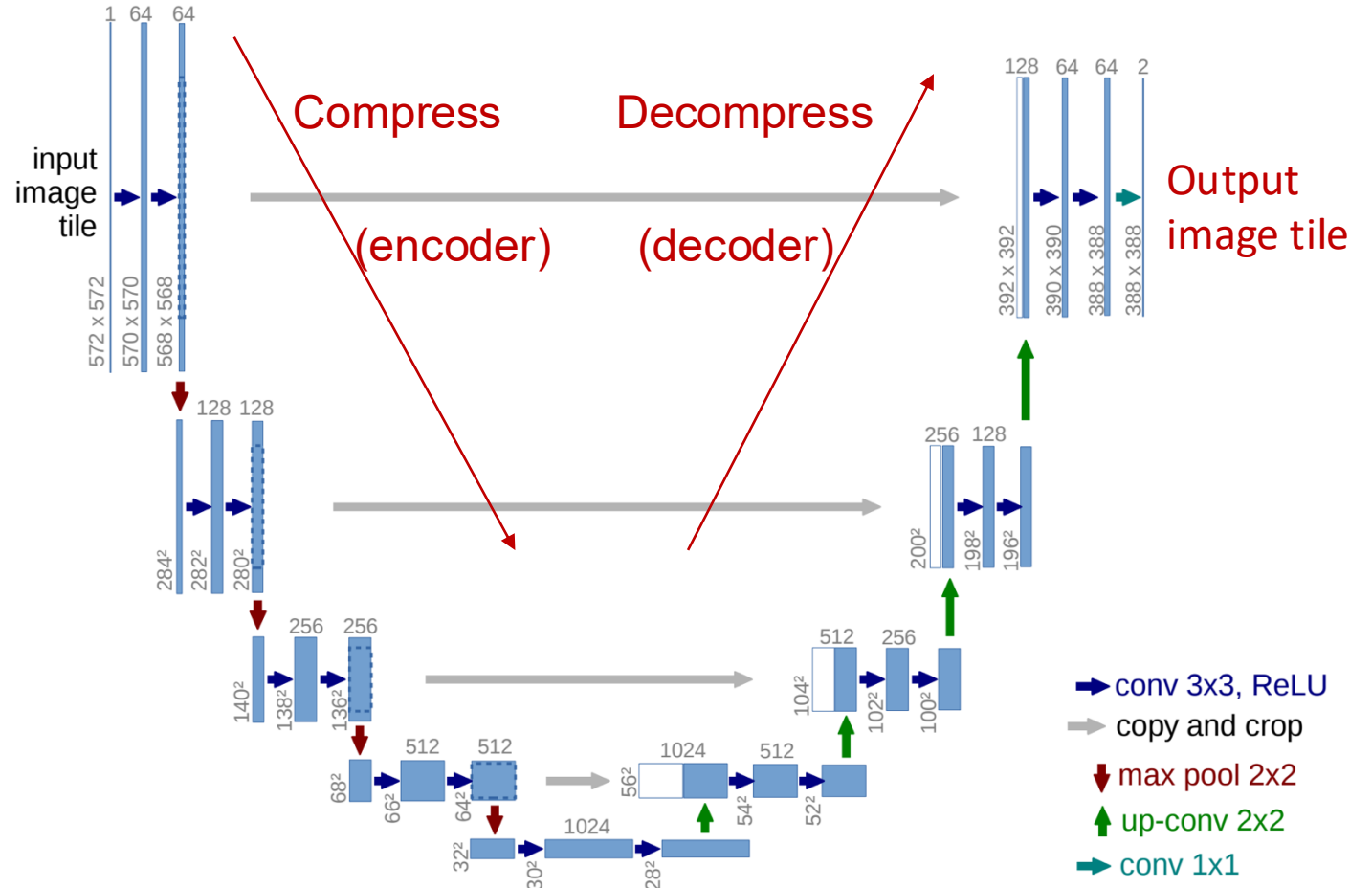
Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOS 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



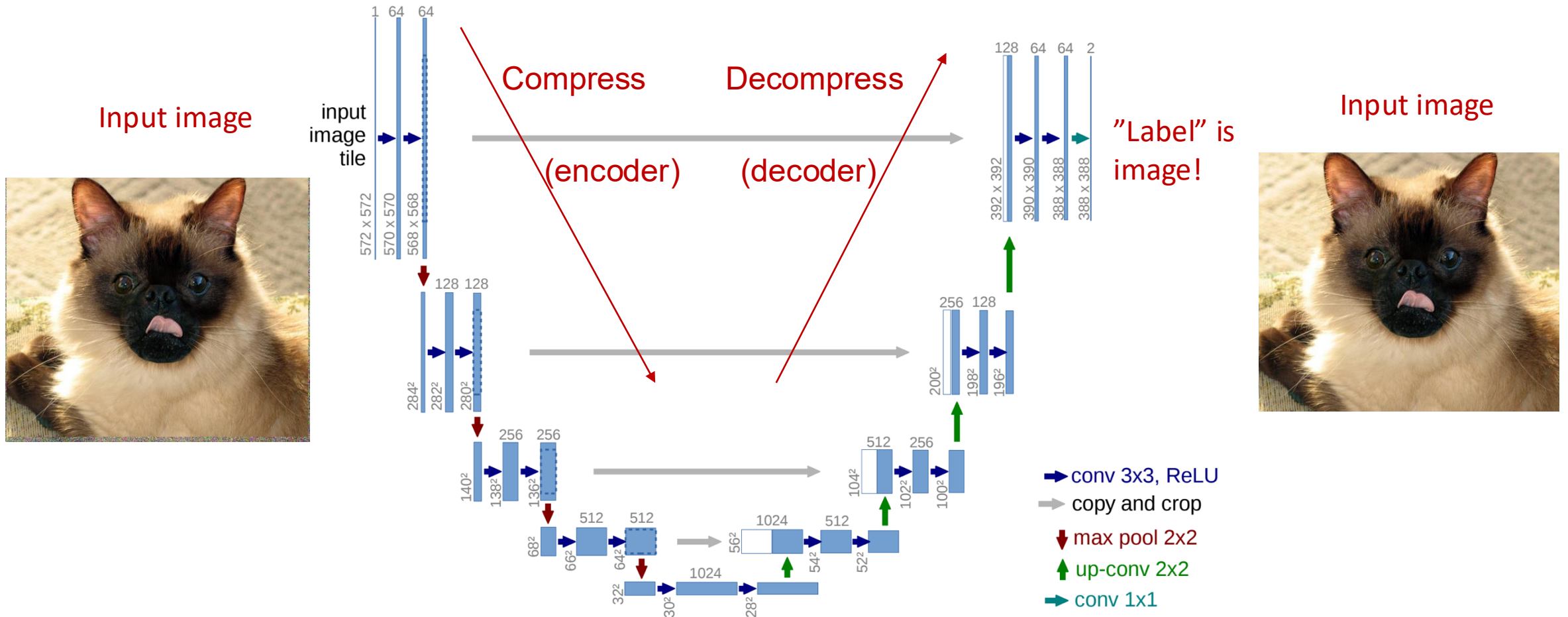
U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

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

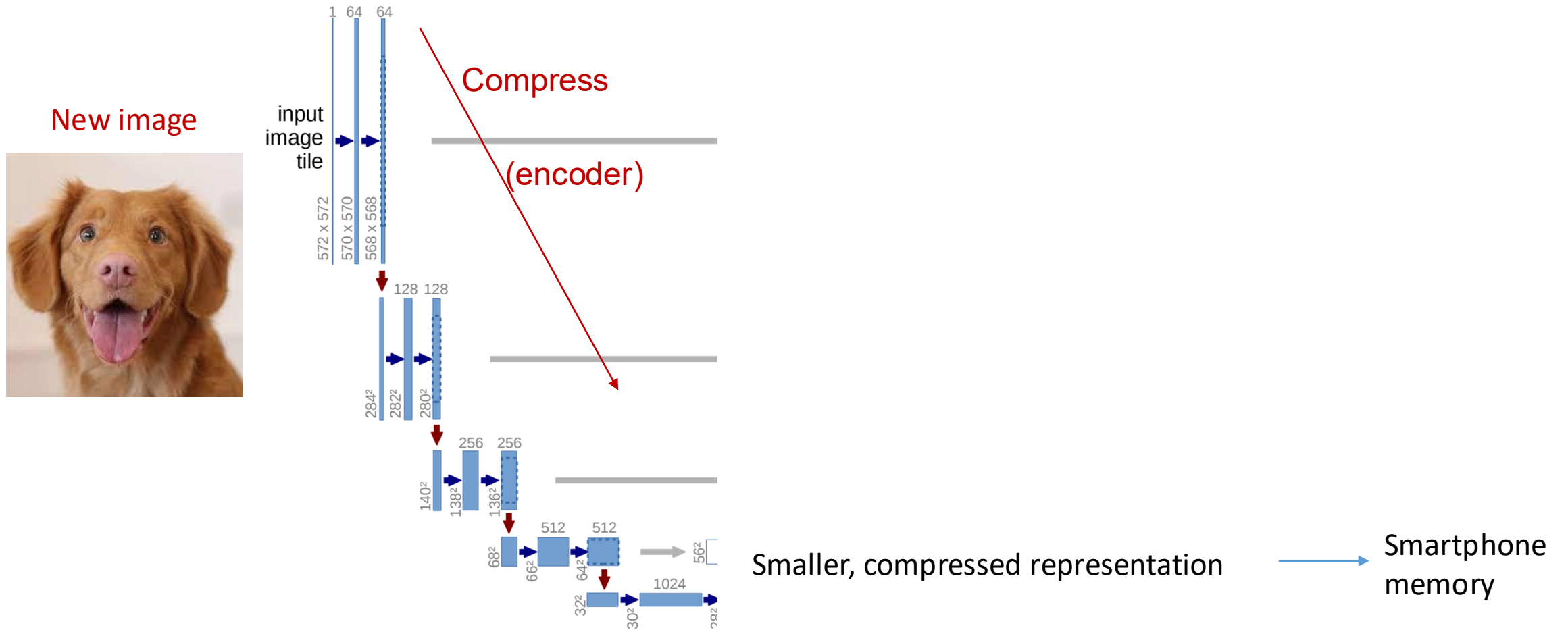
Example: Learned compression

U-Net Architecture



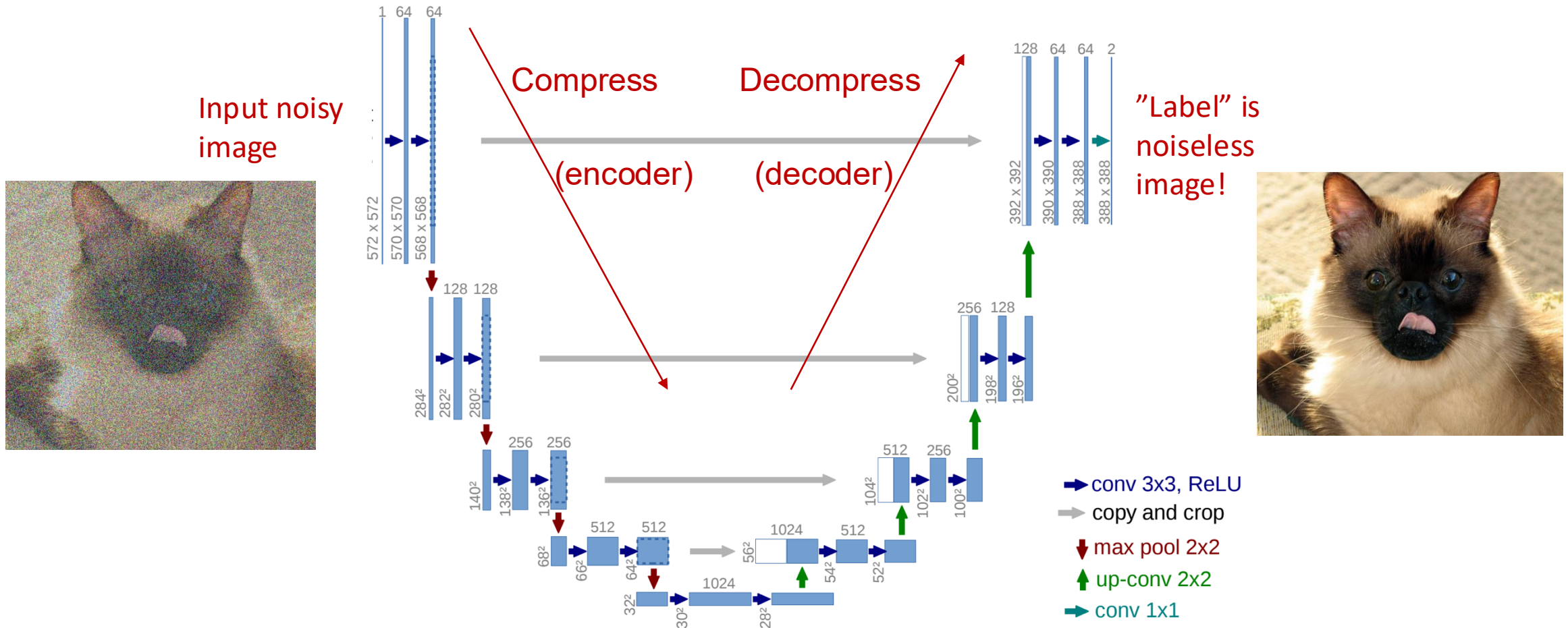
Example: Learned compression

U-Net Architecture



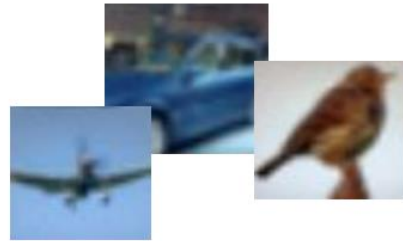
Example: Denoising Autoencoder

U-Net Architecture



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

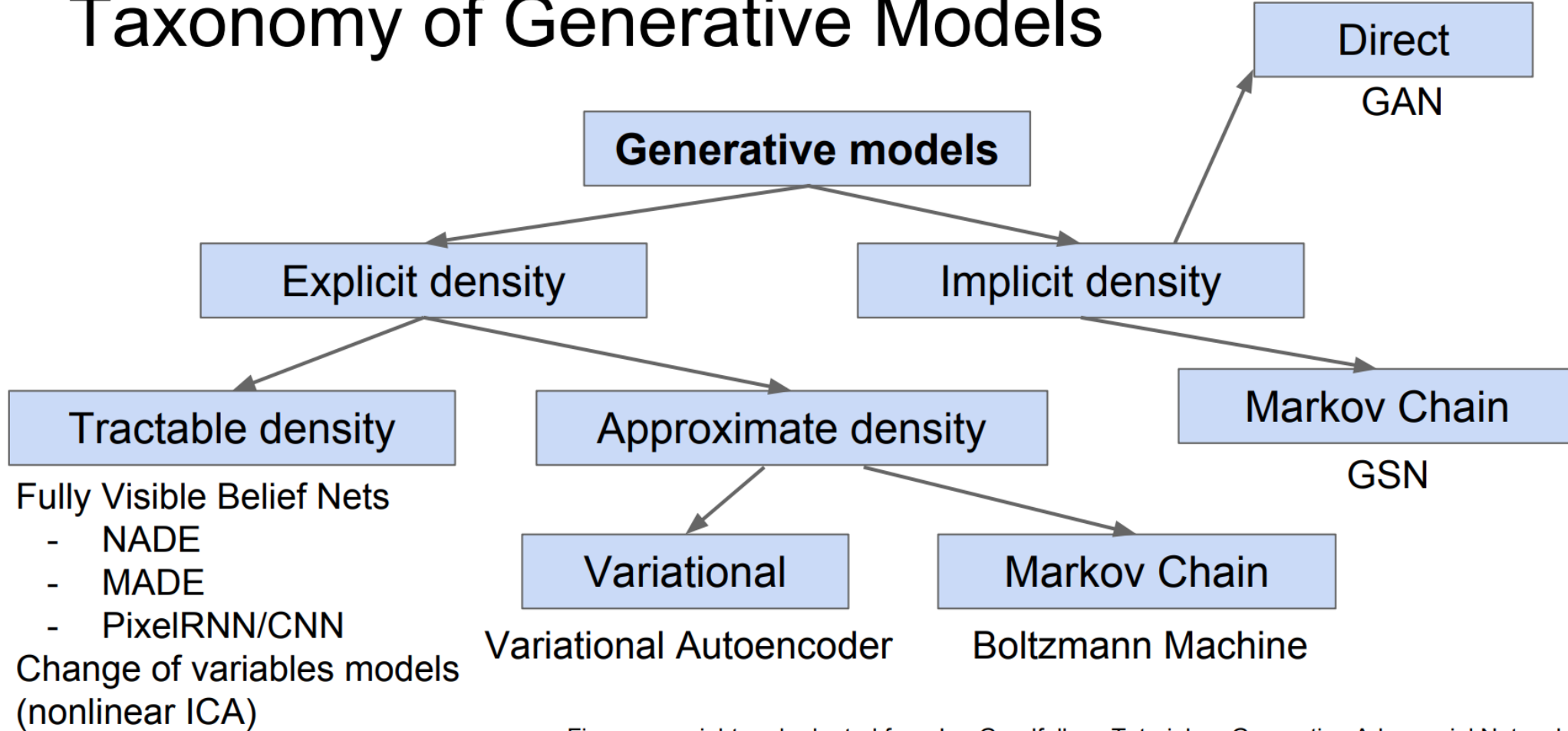


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

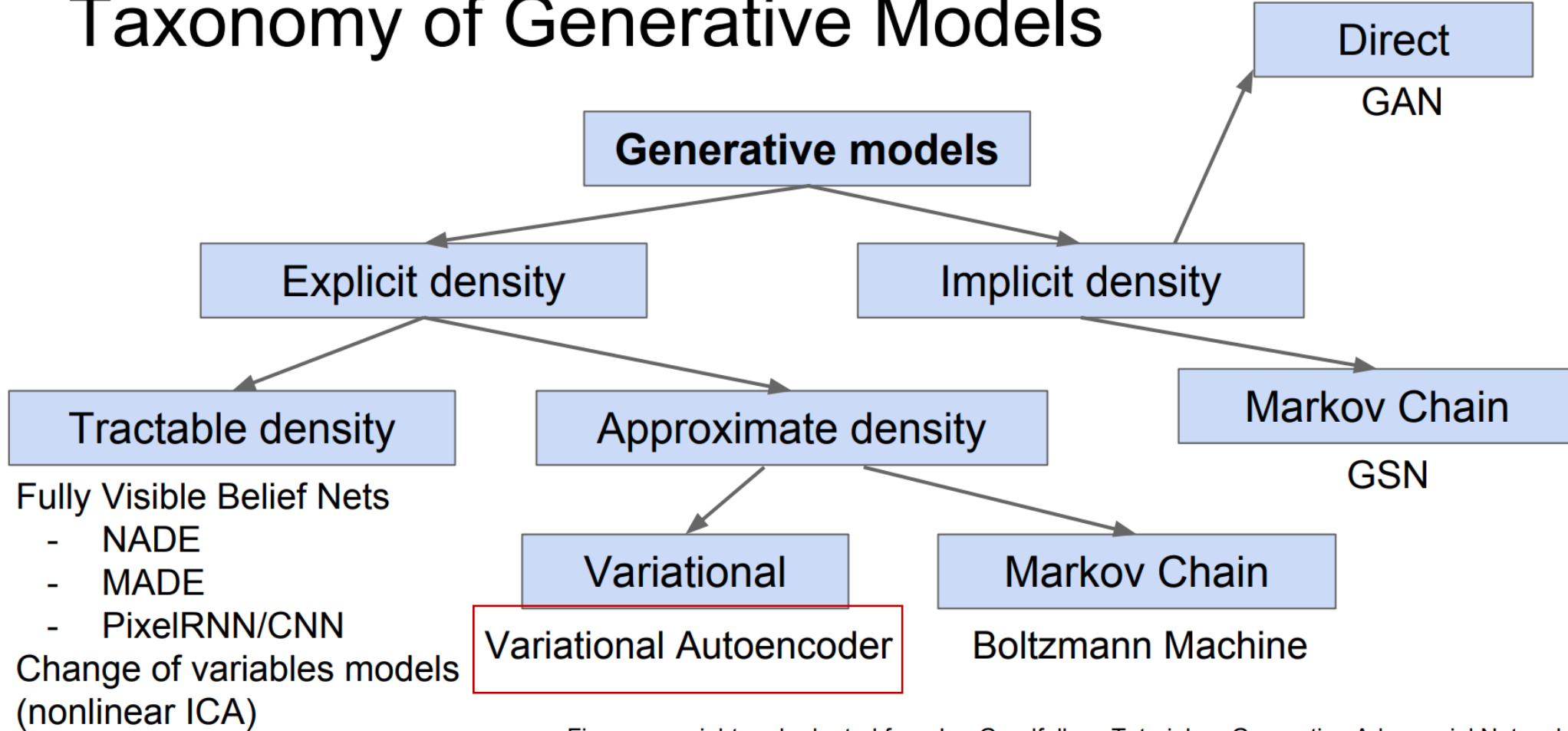
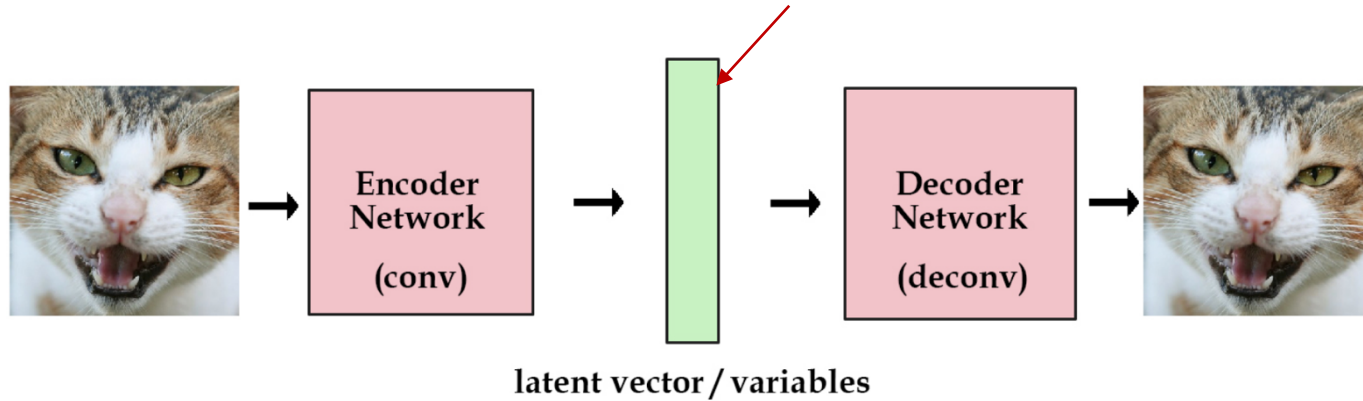


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



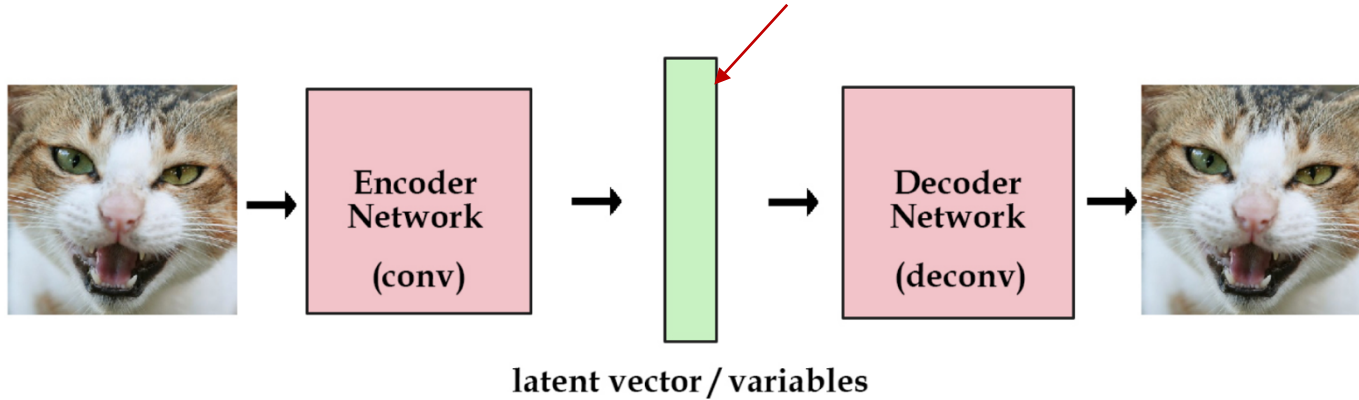
- Good *generative model*
- Have a clean probability distribution to select from to generate new examples

Minimize (KL) distance between latent vector and Gaussian normal



Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

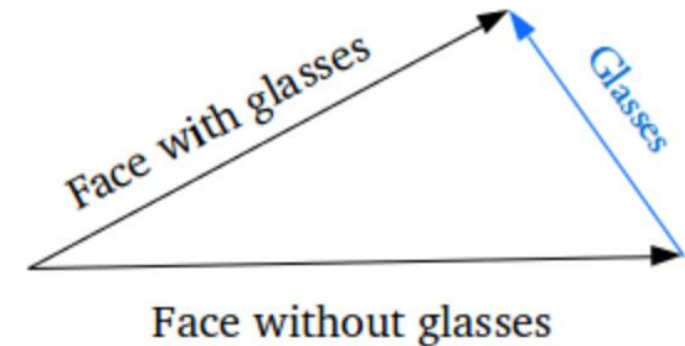


Minimize (KL) distance between latent vector and Gaussian normal

Generative Example (once trained):

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

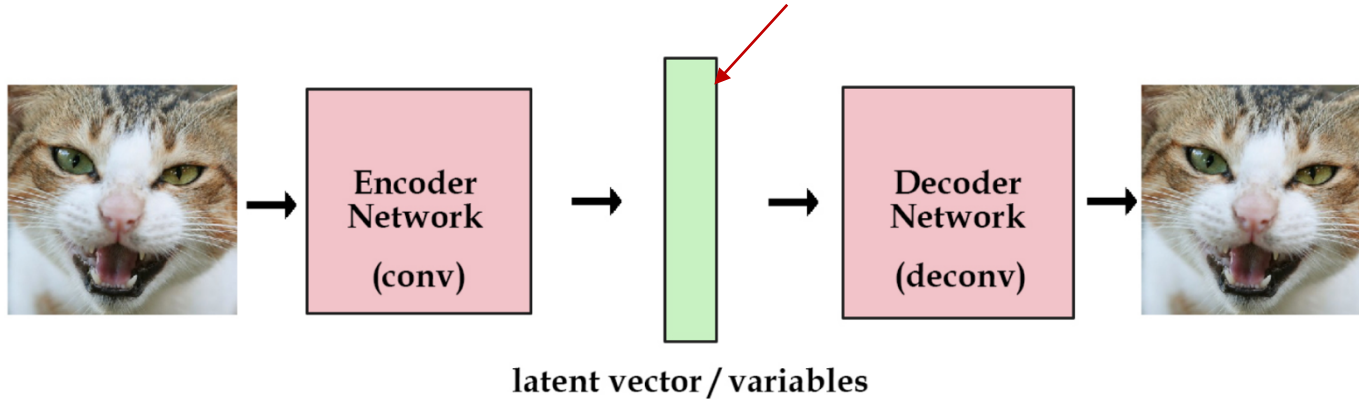
Glasses



Exploring a specific variation of input data[1]

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

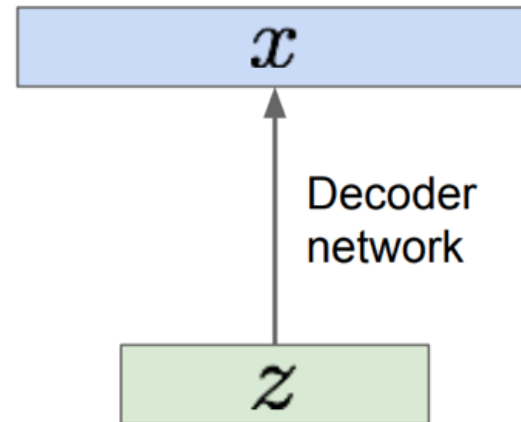


Sample from true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from true prior

$$p_{\theta^*}(z)$$



Taxonomy of Generative Models

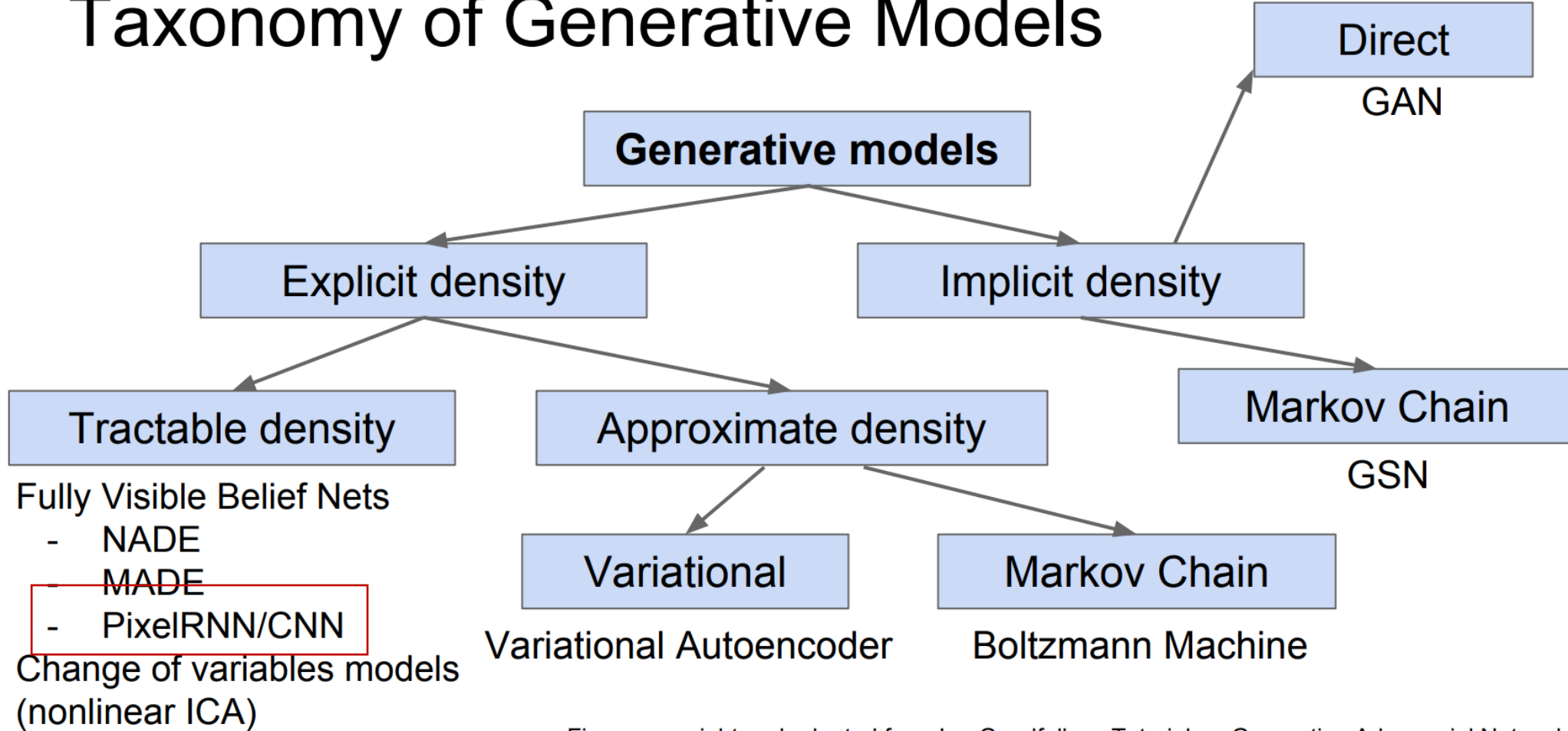


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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!

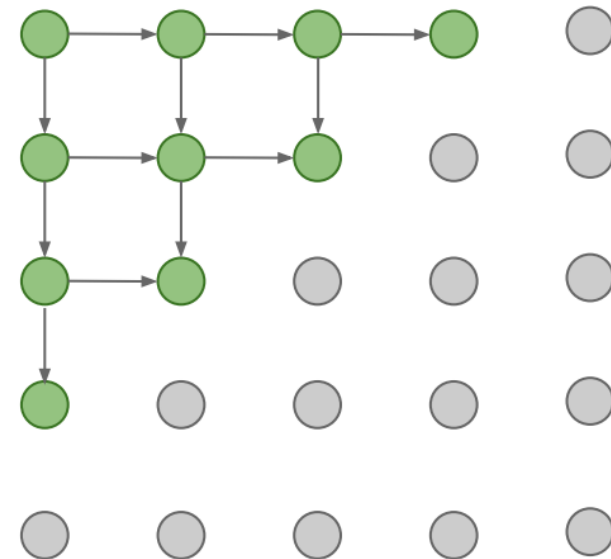




Figure 1. Image completions sampled from a PixelRNN.

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

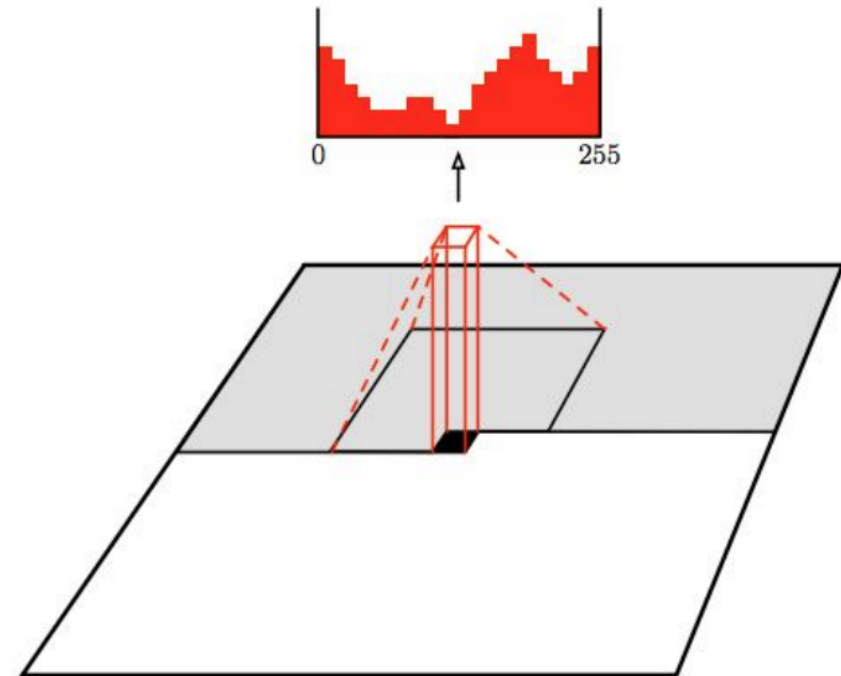


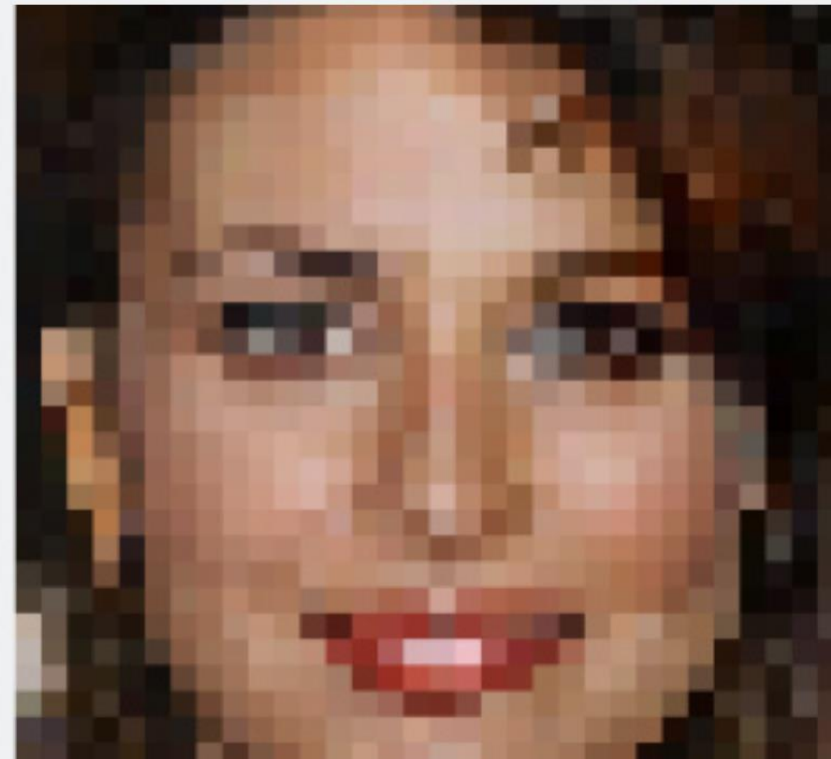
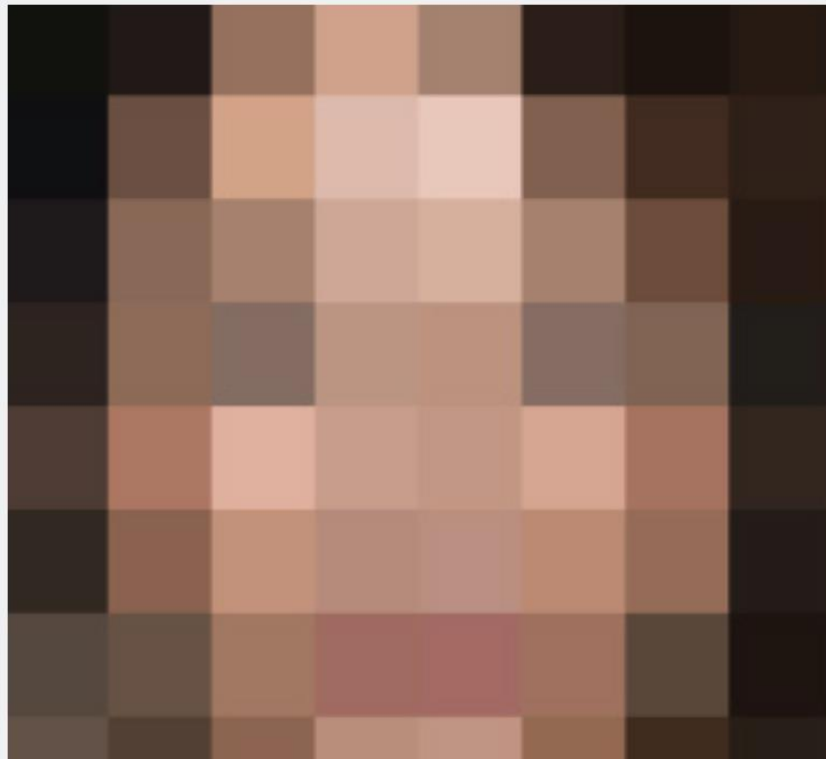
Figure copyright van der Oord et al., 2016. Reproduced with permission.

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



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

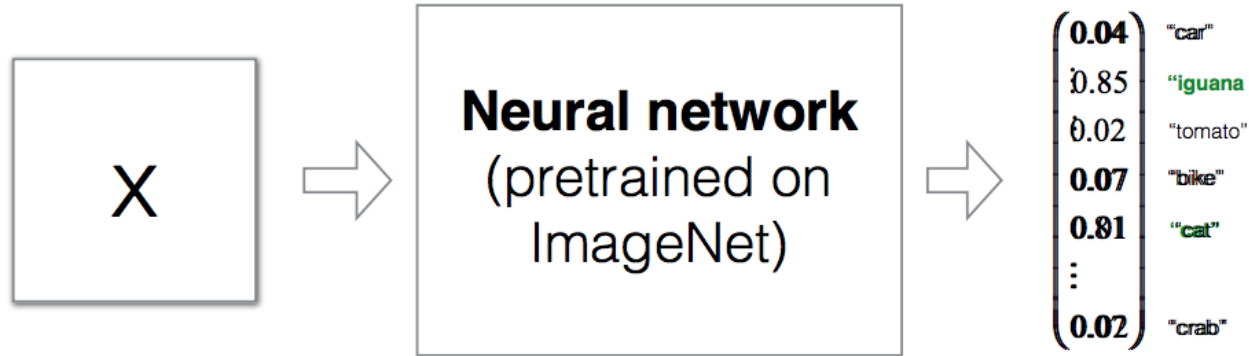
What if we give up on explicitly modeling density, and just want ability to sample?

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

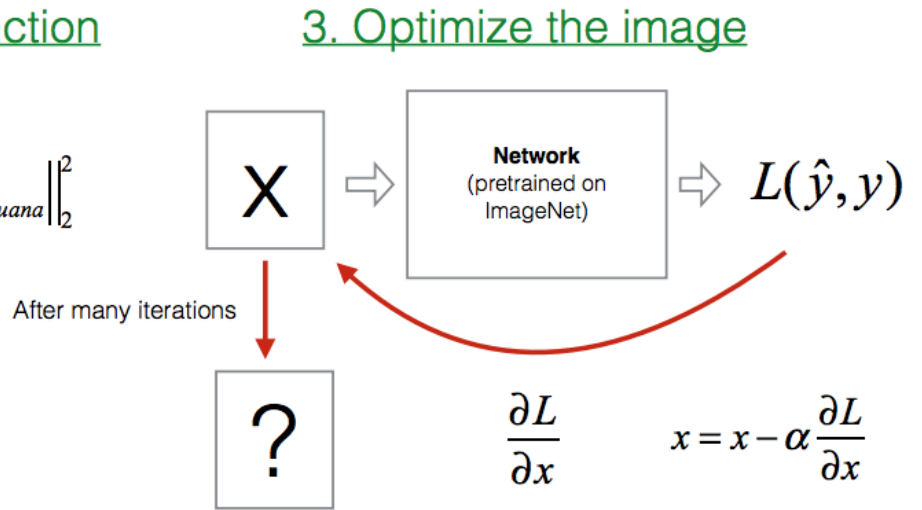
Goal: Given a network pretrained on ImageNet, find an input image that is not a iguana but will be classified as an iguana.



1. Rephrasing what we want:

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

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


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

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

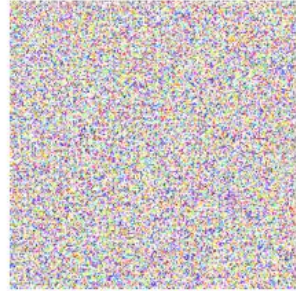
I. A. Attacking a network with adversarial examples

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

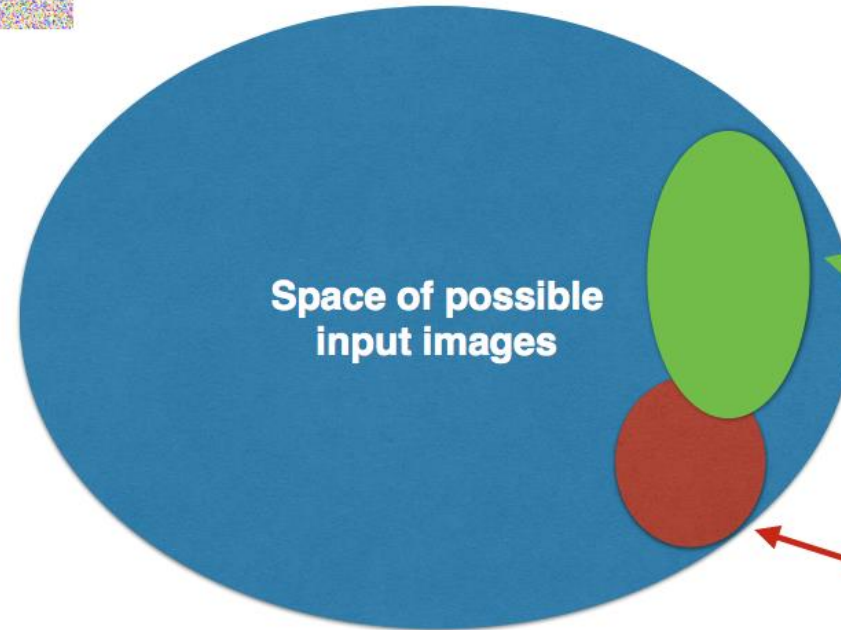


I. A. Attacking a network with adversarial examples

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



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



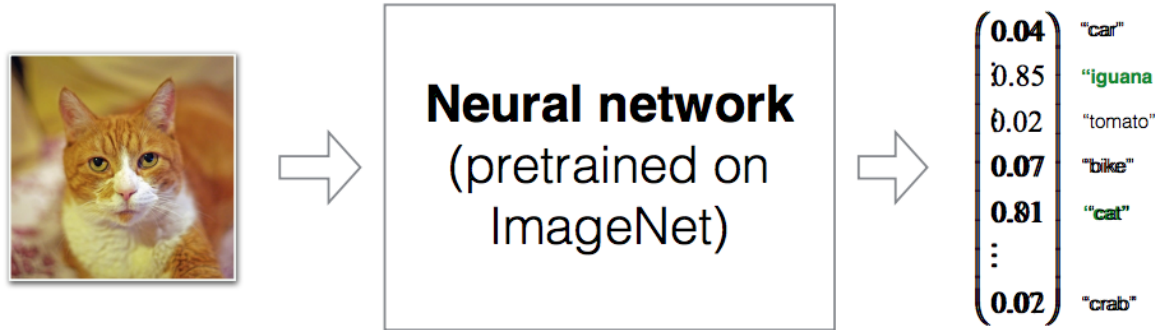
Space of possible input images

Space of images classified as iguanas

Space of real images

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:

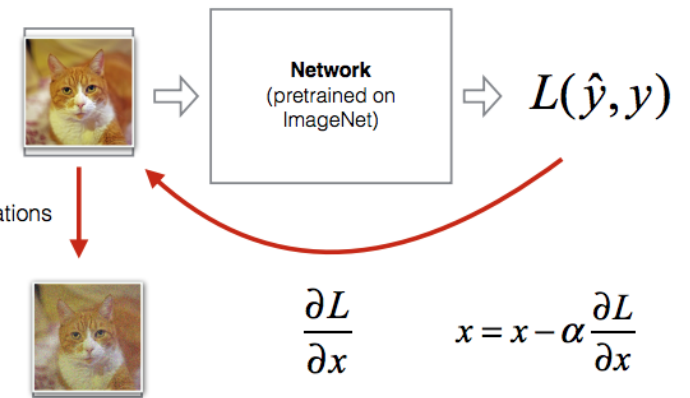
Find x such that: $\hat{y}(x) = y_{iguana} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

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

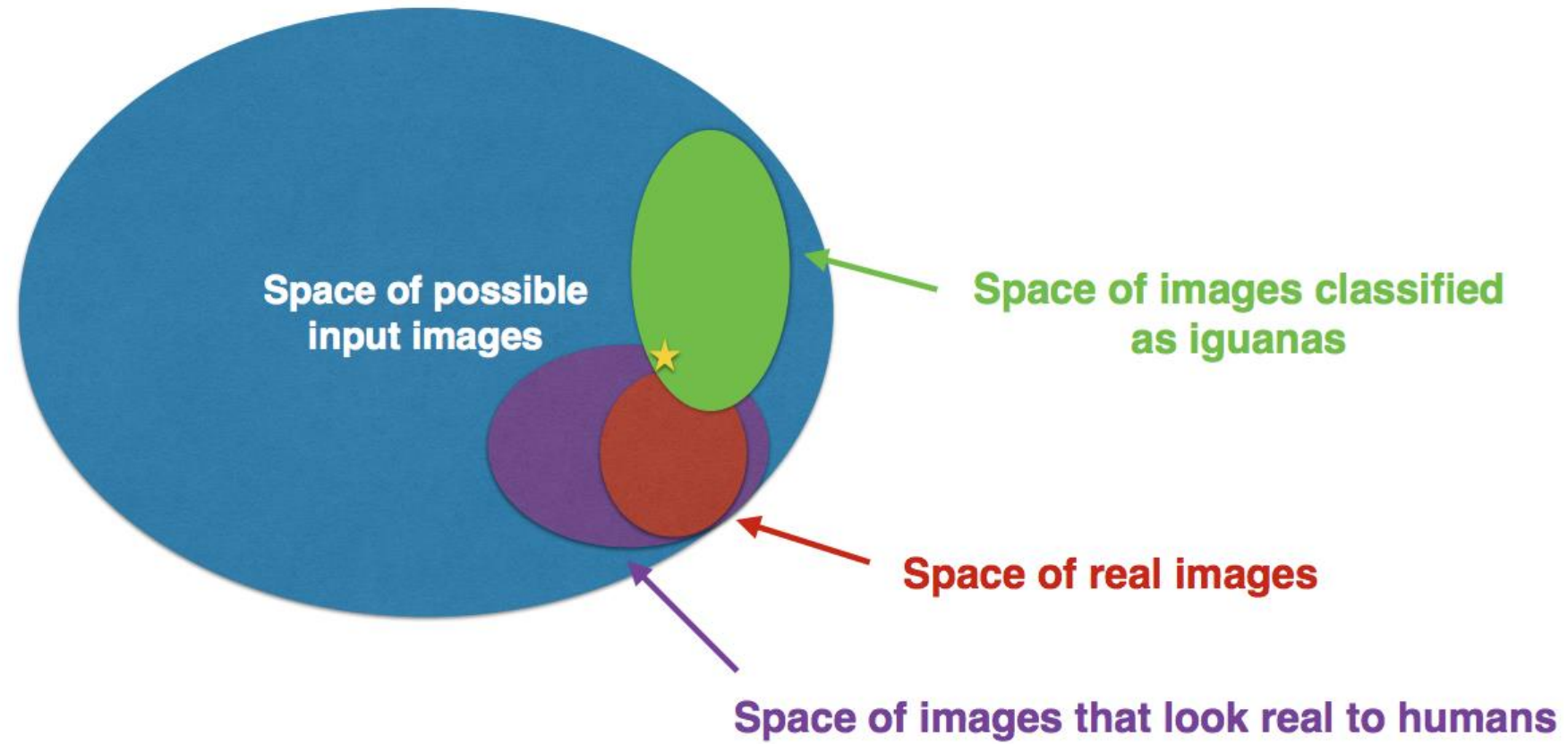


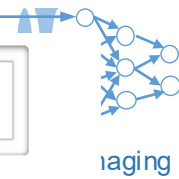
[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

I. A. Attacking a network with adversarial examples

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





Knowledge of the attacker:

- White-box
- Black-box

Solution 1

- Create a SafetyNet

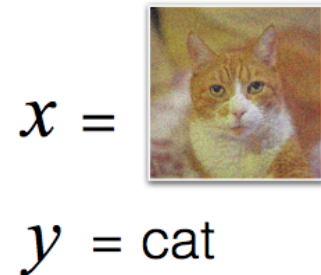
Solution 2

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



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

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

Generative adversarial networks

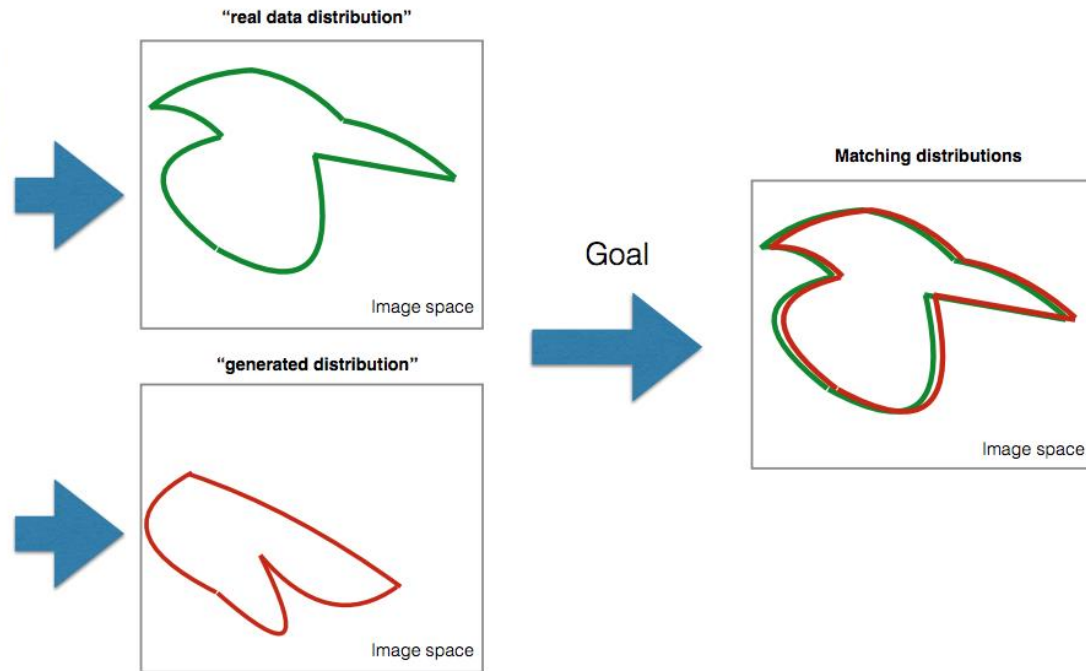
II.A - Motivation

Probability distributions:

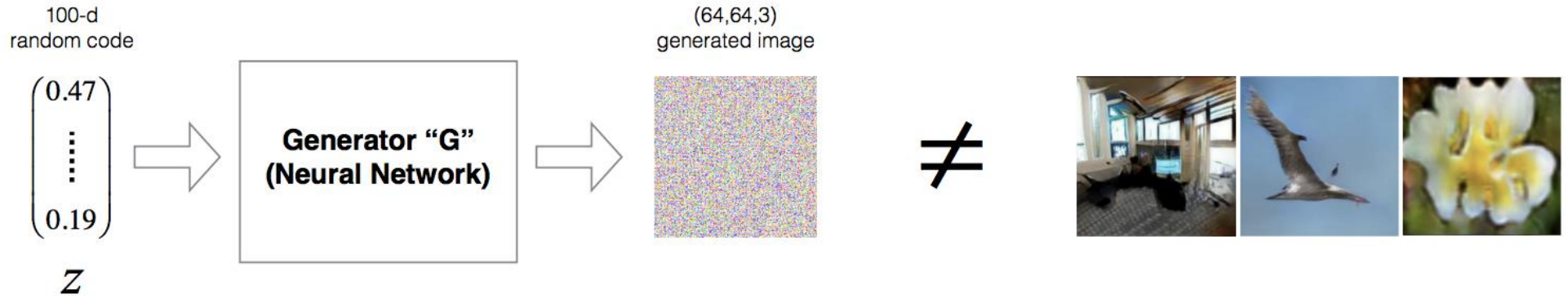
Samples from the “real data distribution”



Samples from the “generated distribution”



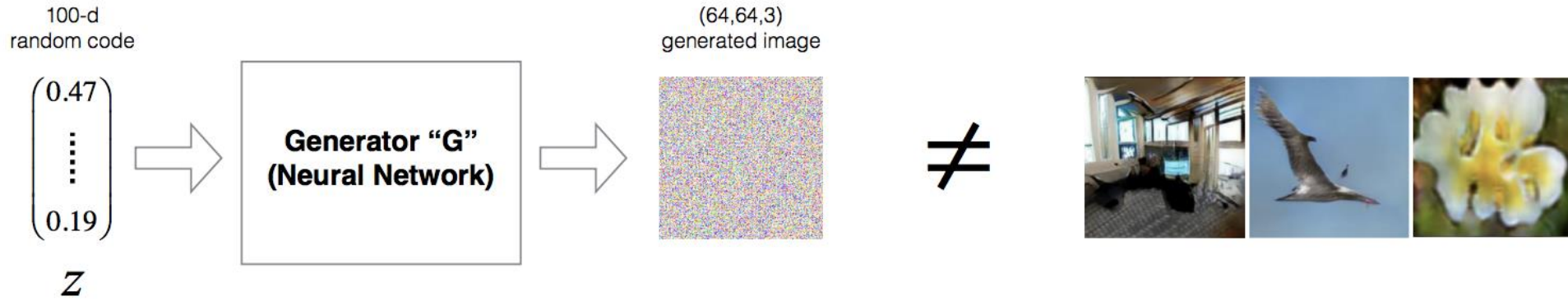
II.B - G/D Game



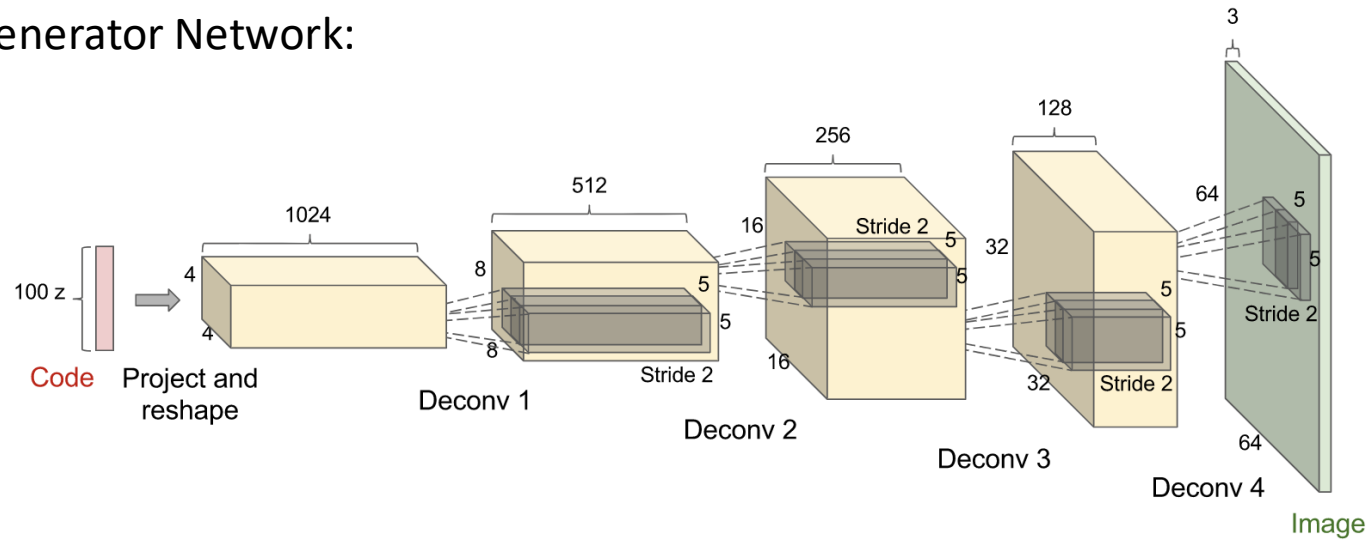
How can we train G to generate images from the true data distributions?

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

II.B - G/D Game



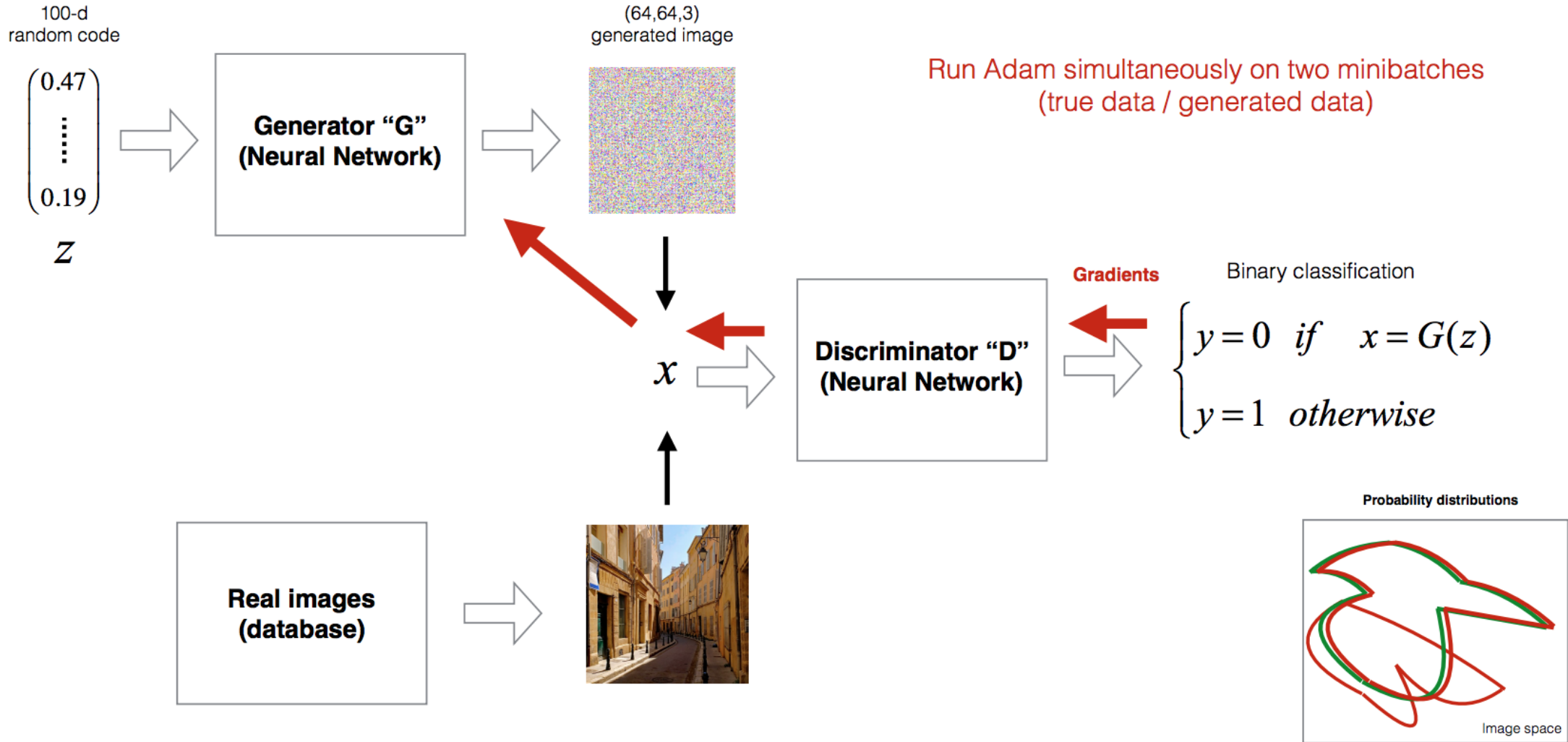
DCGAN Generator Network:



[Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiao lei 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



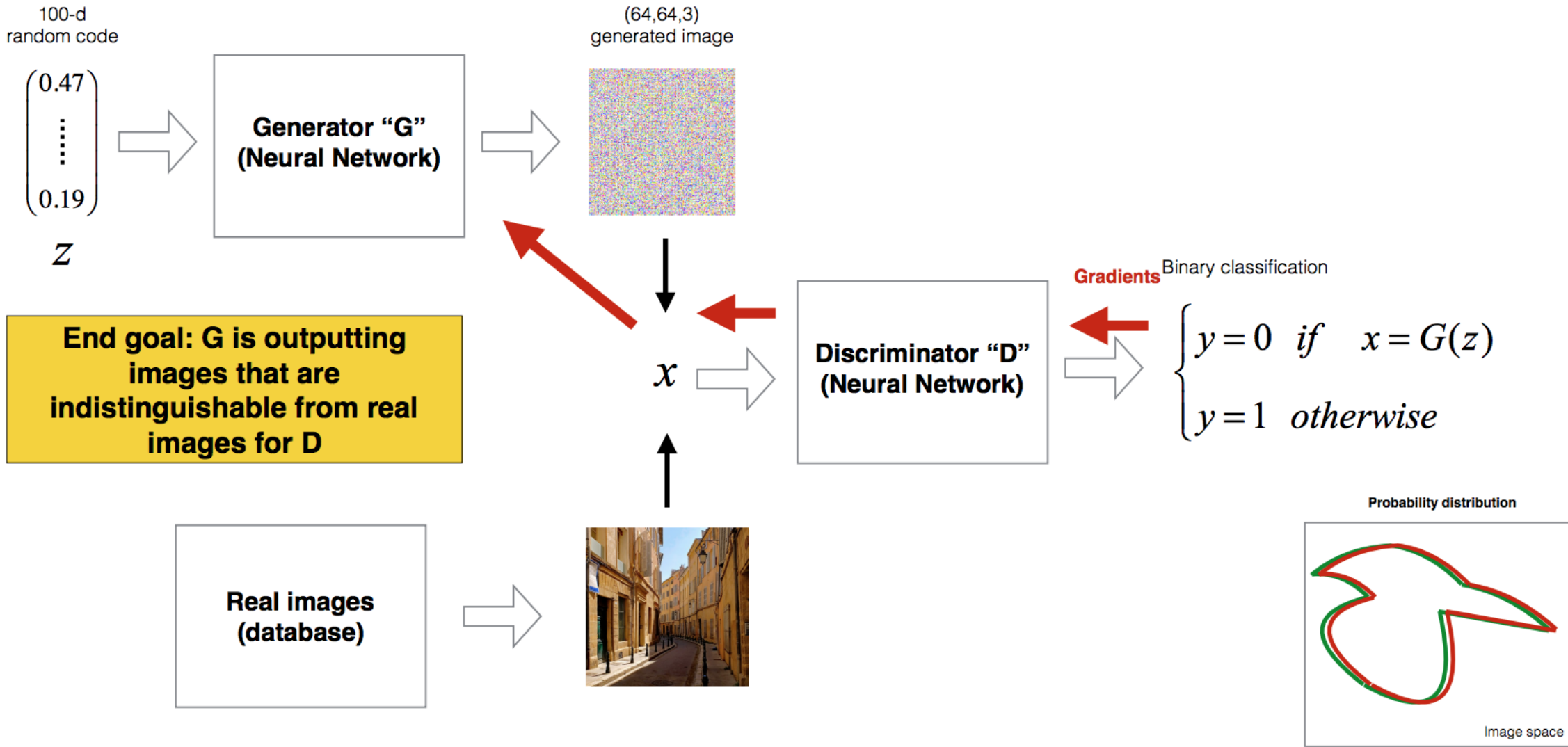
Run Adam simultaneously on two minibatches
(true data / generated data)

Gradients

Binary classification

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

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

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

- The cost of the generator

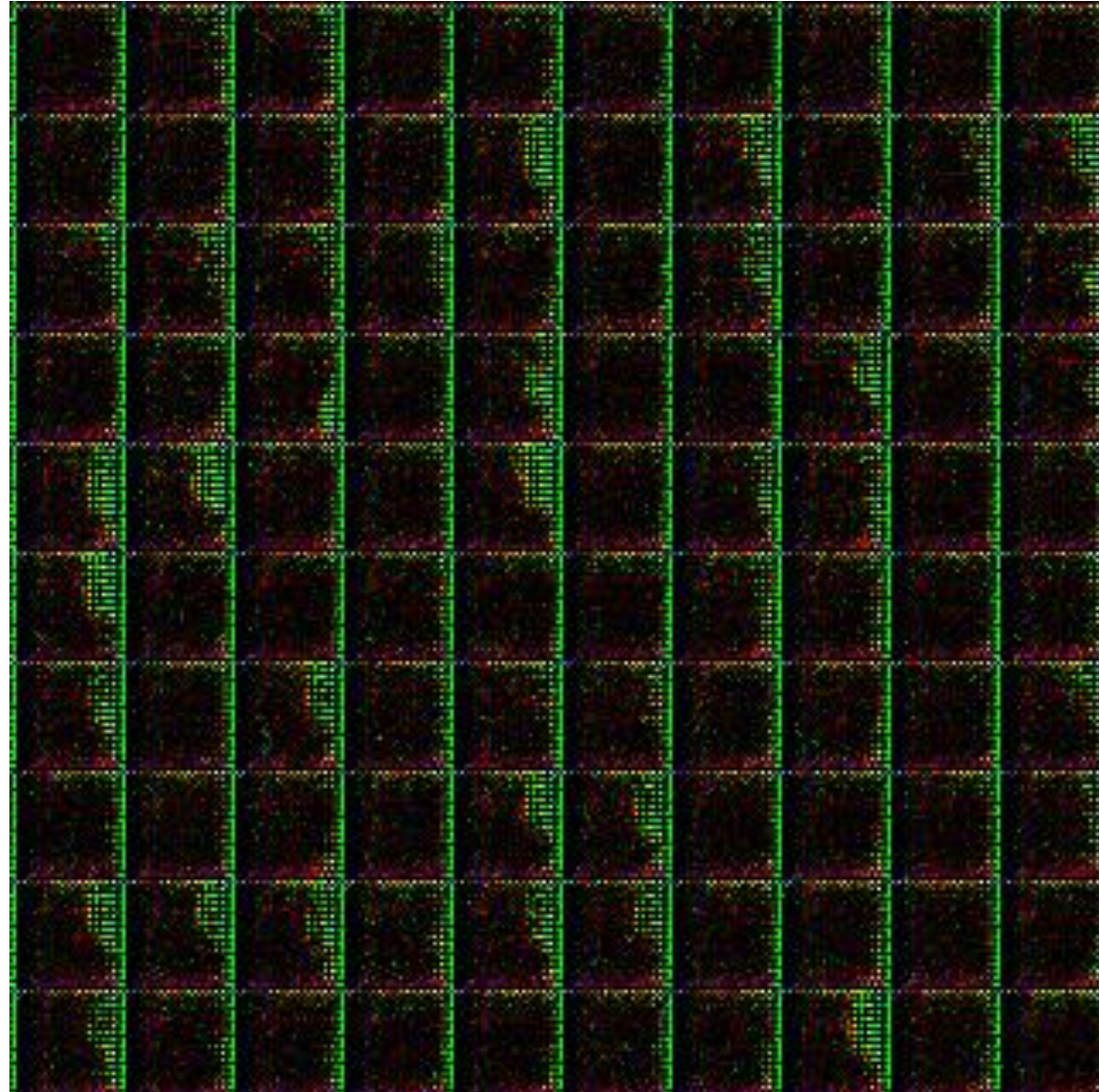
$$J^{(G)} = -J^{(D)} = \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log(1 - D(G(z^{(i)})))$$

"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"

Typical # of weights:
100 Million

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



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(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)

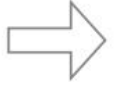
    return Model(img, validity)
```

II.E - Nice results

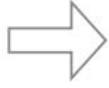
Operation on codes

Code 1

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

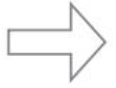


Generator "G"
(Neural Network)

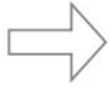


Code 2

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

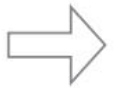


Generator "G"
(Neural Network)

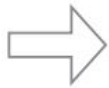


Code 3

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

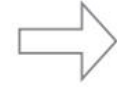


Generator "G"
(Neural Network)

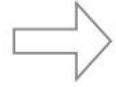


Code 1 Code 2 Code 3

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



Generator "G"
(Neural Network)



Man with glasses - man + woman = woman with glasses

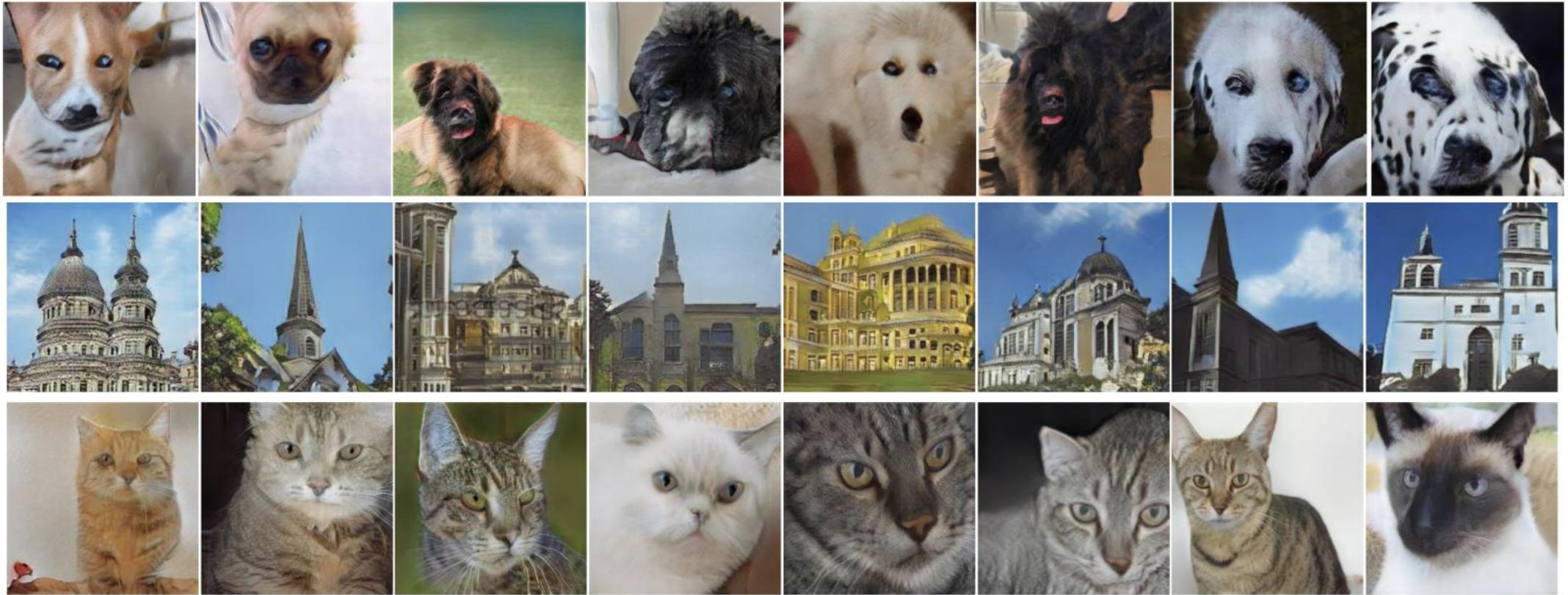
[Radford et al. (2015): UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

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

4 different adversarial examples for object detector:



4 different adversarial examples for object classifier:



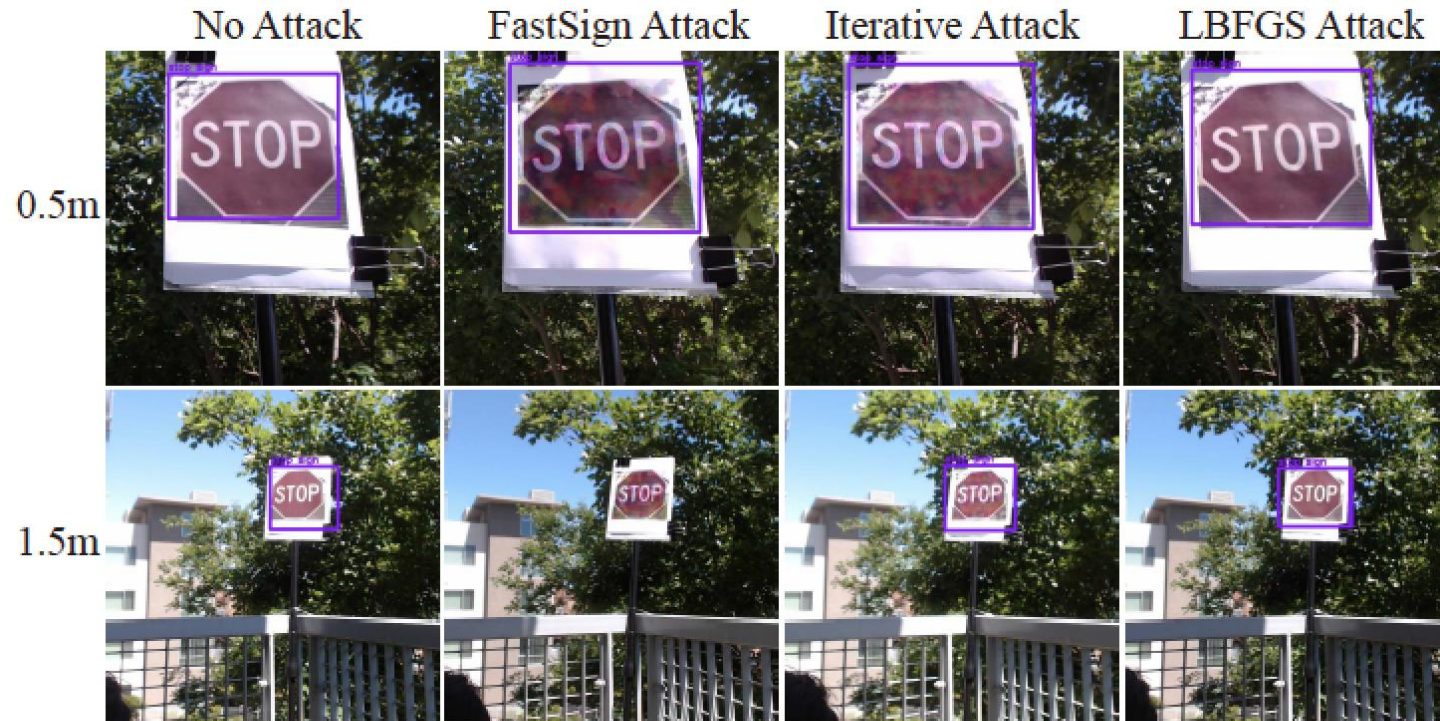


Figure 3: This figure shows experiment setup, and we use the printed stop signs to simulate real stop signs with natural background. These are examples for successful 0.5 meters and 1.5 meters detection: both original images and adversarial examples are detected in both distances. It demonstrates that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

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

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