

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

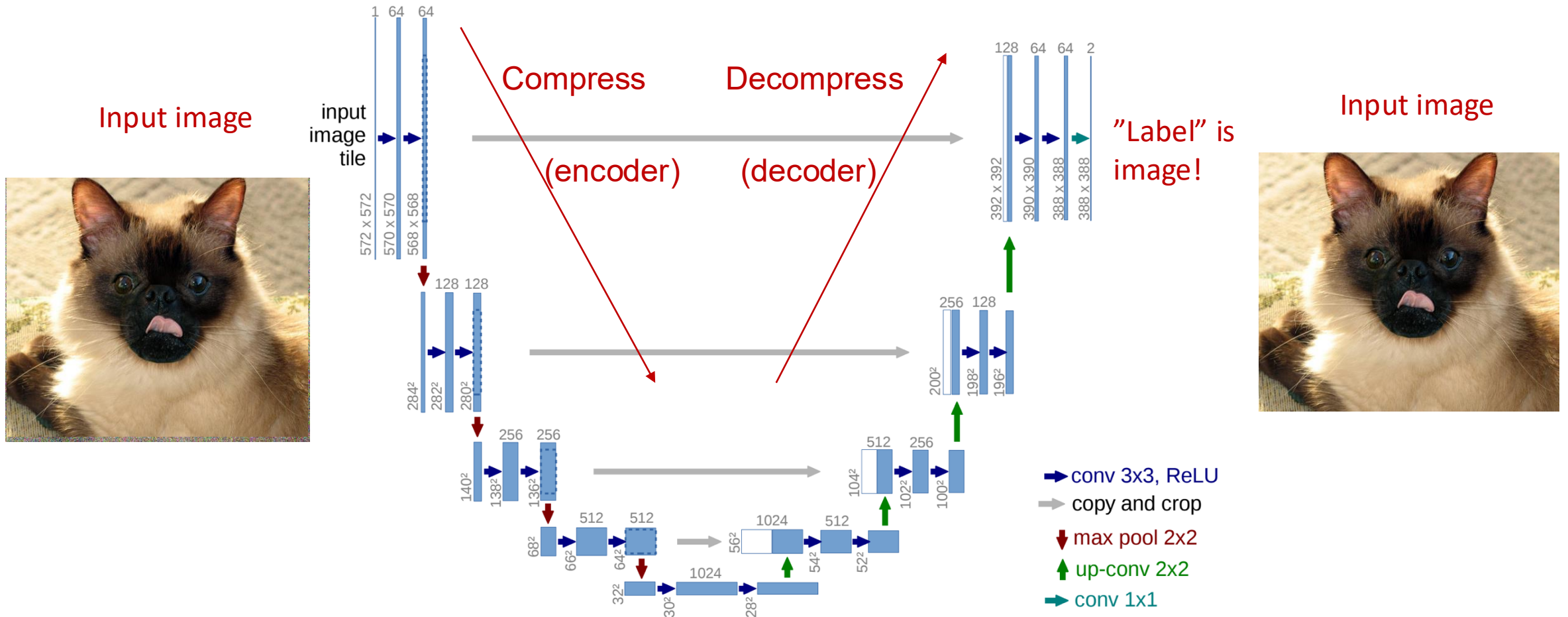
Generative models, adversarial examples and GANs

Announcements

- Homework #3 due Friday

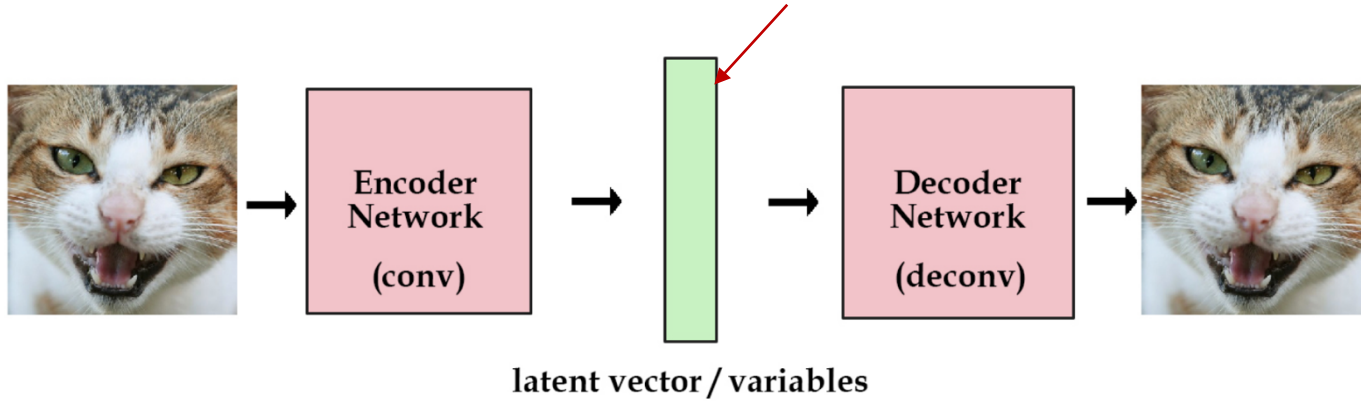
Review from last class: Autoencoders

U-Net Architecture



Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

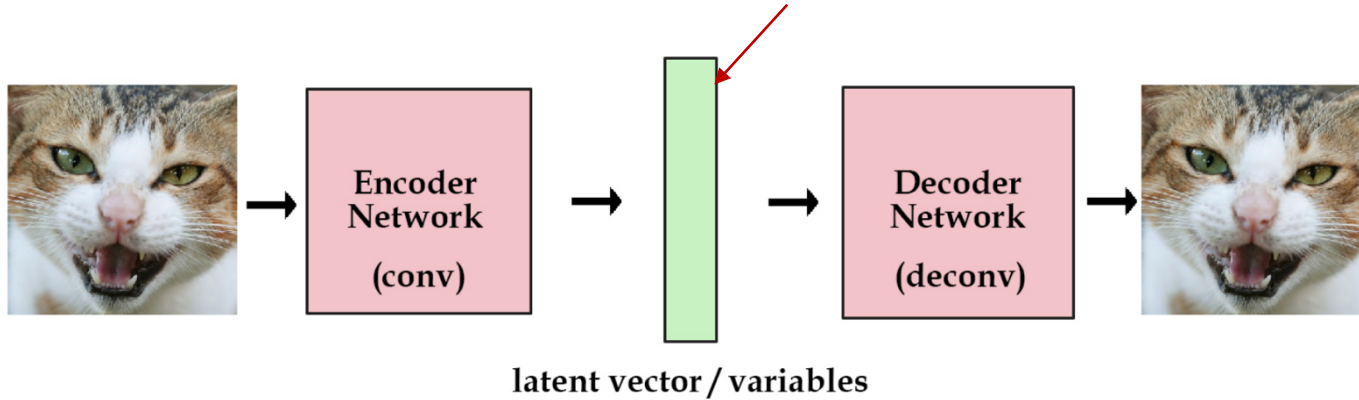


Minimize (KL) distance between latent vector and Gaussian normal

- 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

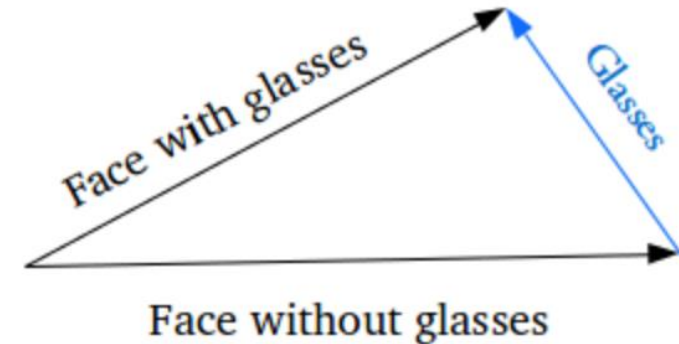


Minimize (KL) distance between latent vector and Gaussian normal

Generative Example (once trained):

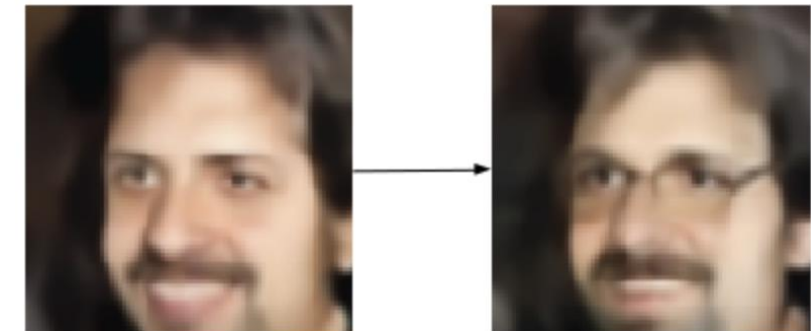
- Encode image with glasses, obtain latent vector PDF P_g
- Encode image without glasses, obtain PDF P_{ng}
- Compute $\mathbf{diff} = P_g - P_{ng}$
- Encode new image to obtain P_{new} , add in \mathbf{diff}
- Decode $P_{new} + \mathbf{diff}$ to get guy with glasses!

- 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]

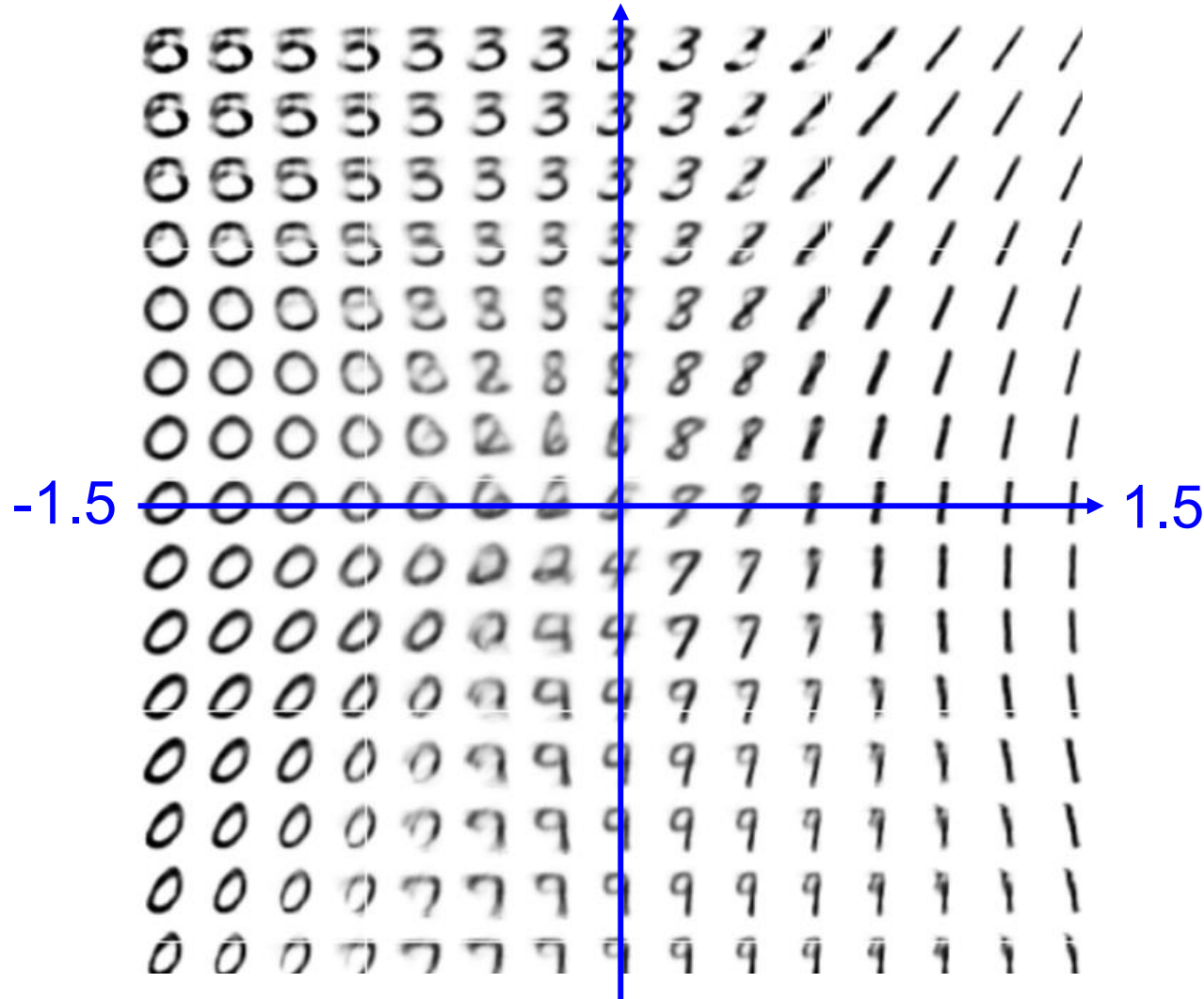
A very simple example – MNIST digits

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

On the way towards generating “fake” images of handwritten digits?

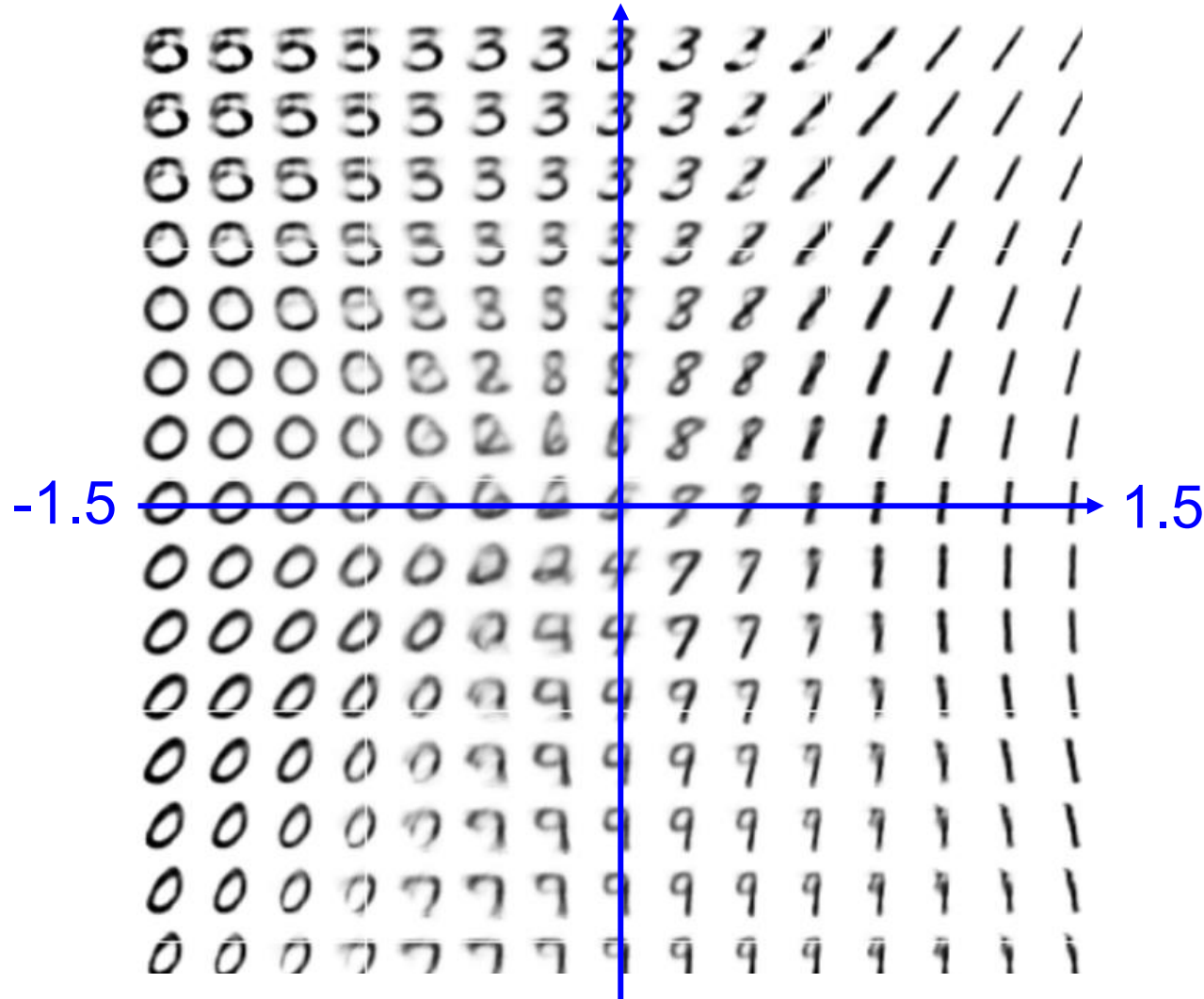


Navigating across the manifold of handwritten digits



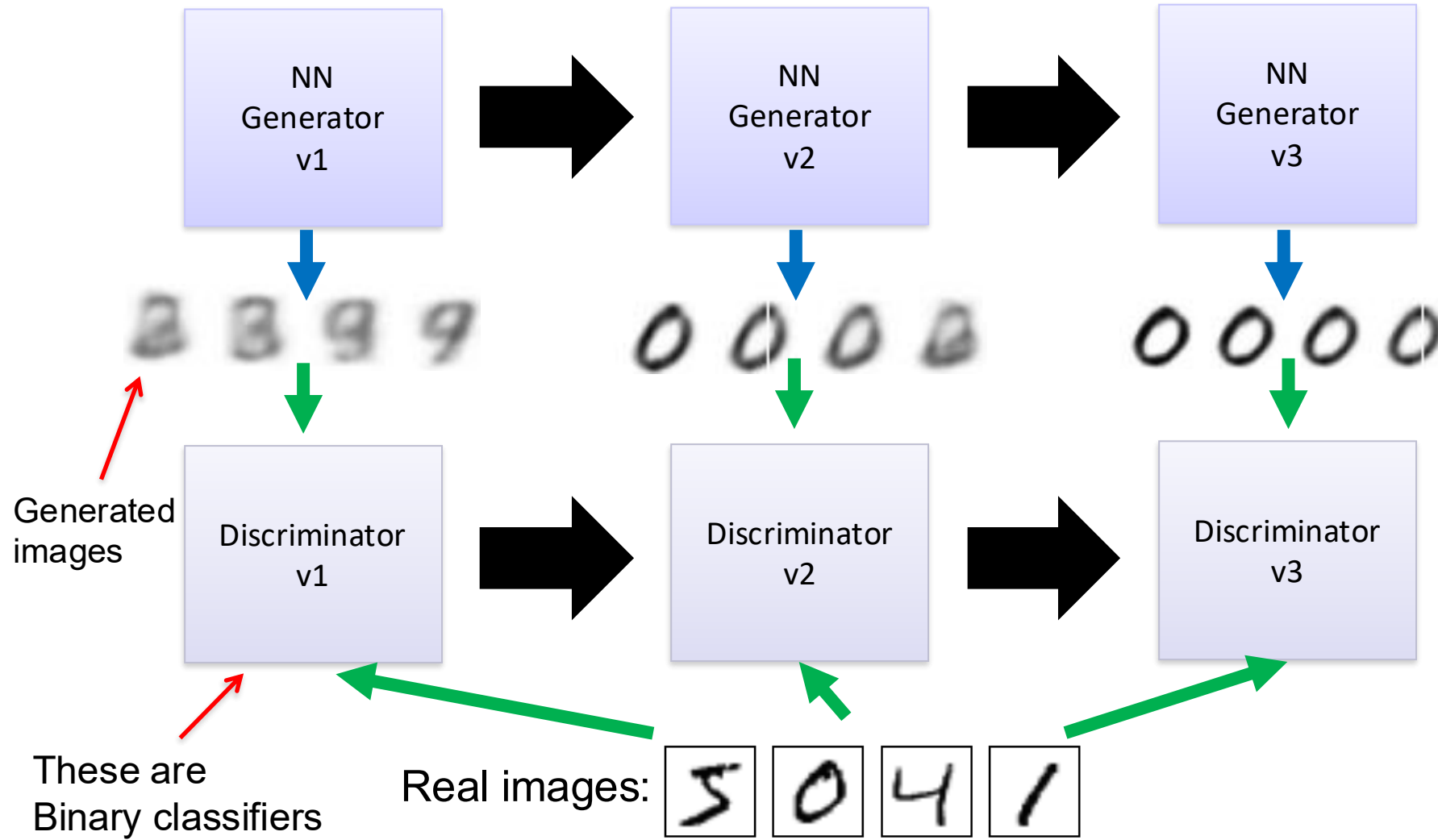
- "Selection" – starting with some vector in latent space
- "Navigation" – movement along vectors
- Selection and Navigation within the latent space produces "visually interesting" results

Navigating across the manifold of handwritten digits



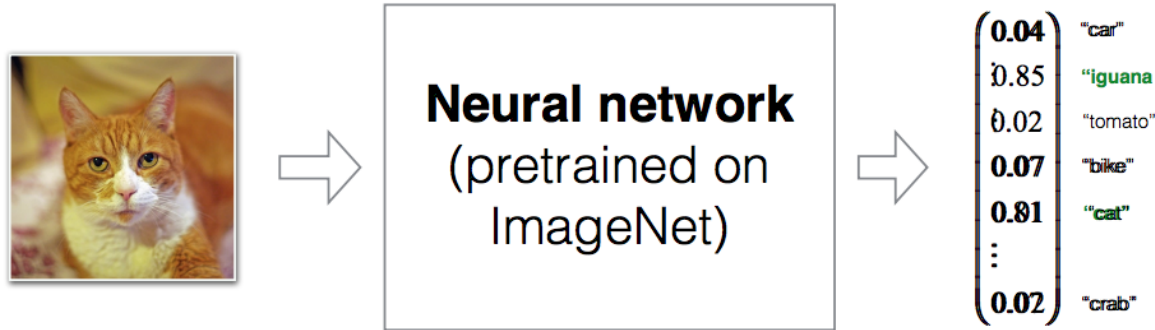
- "Selection" – starting with some vector in latent space
- "Navigation" – movement along vectors
- Selection and Navigation within the latent space produces "visually interesting" results
- If we wanted to make fake digits, **How might we ensure that they are actually correct?**

One approach: sequentially check if “generated” images match ground-truth with a classifier



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:

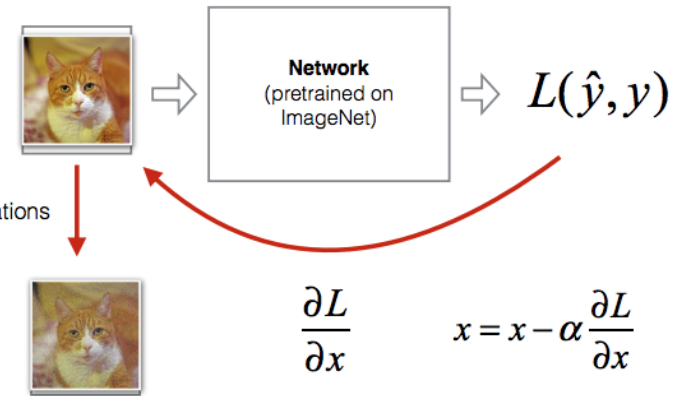
Find x such that: $\hat{y}(x) = y_{iguana} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

And: $x = x_{cat}$

2. Defining the loss function

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

3. Optimize the image

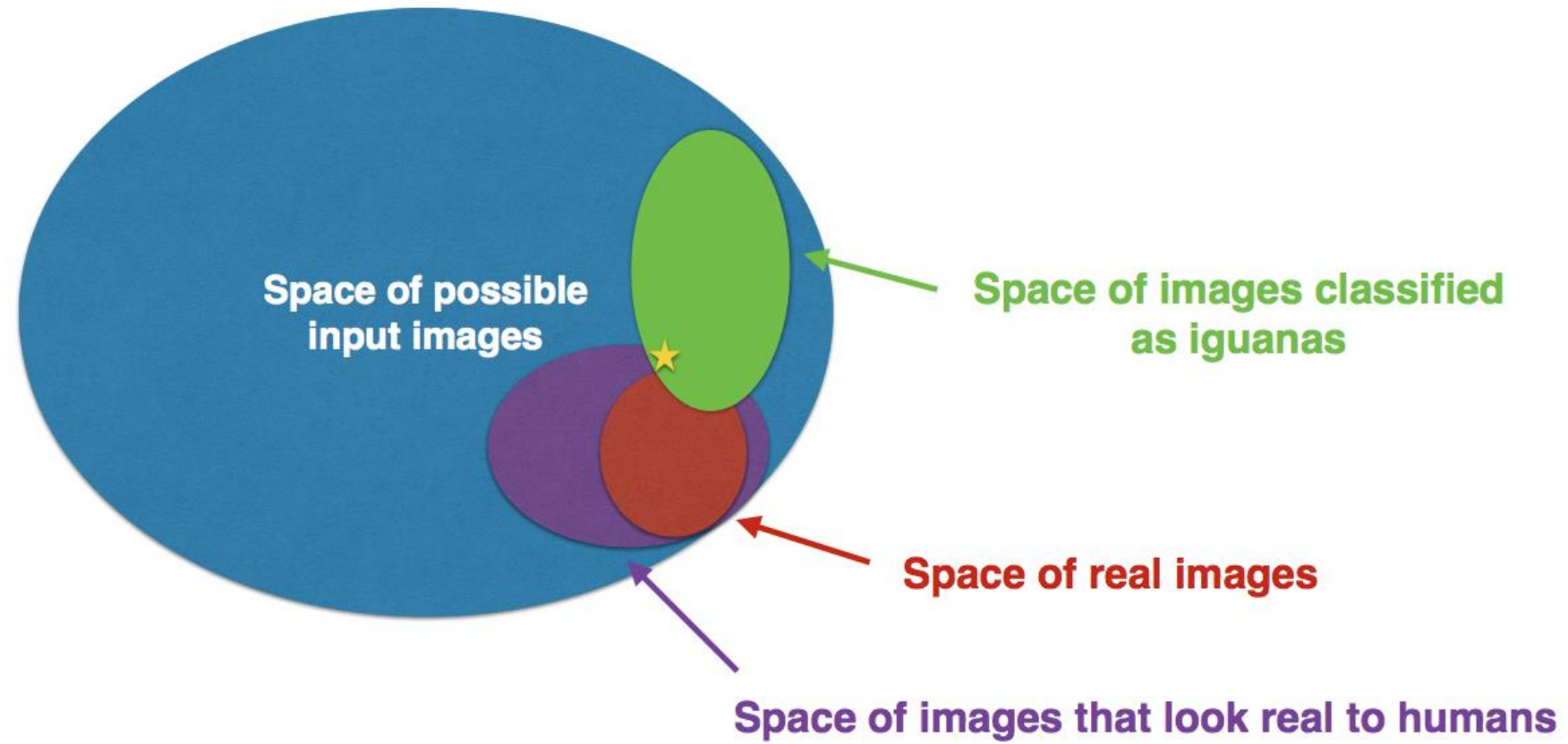


[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

I. A. Attacking a network with adversarial examples

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



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:



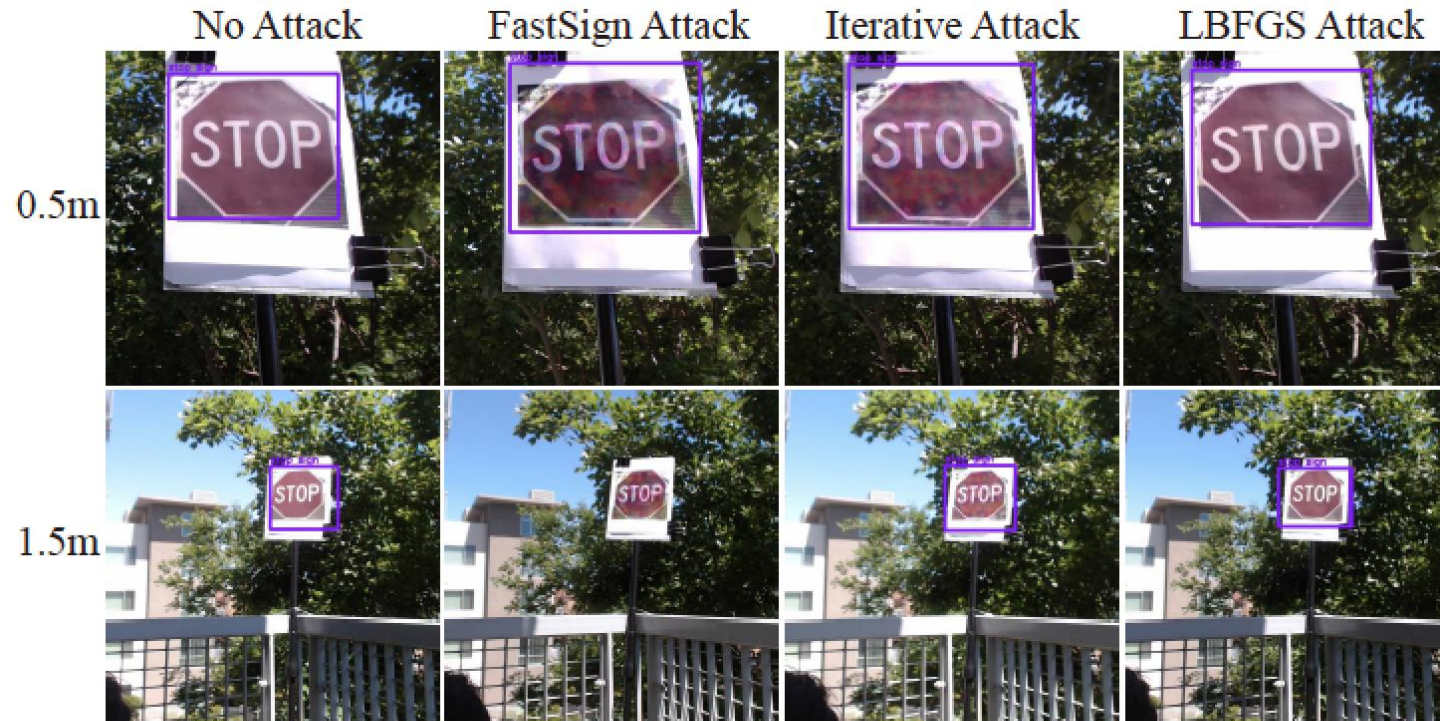
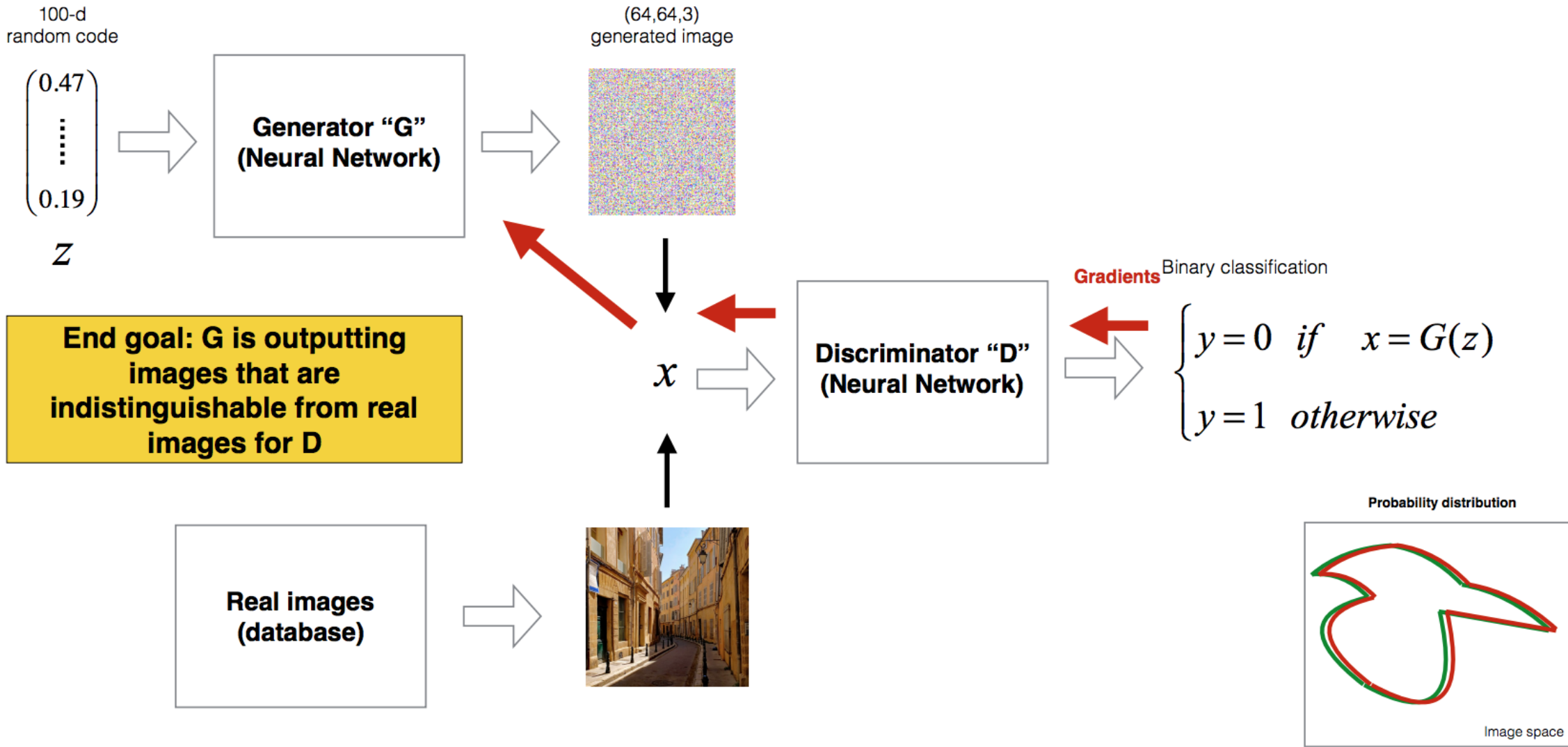


Figure 3: This figure shows experiment setup, and we use the printed stop signs to simulate real stop signs with natural background. These are examples for successful 0.5 meters and 1.5 meters detection: both original images and adversarial examples are detected in both distances. It demonstrates that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

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

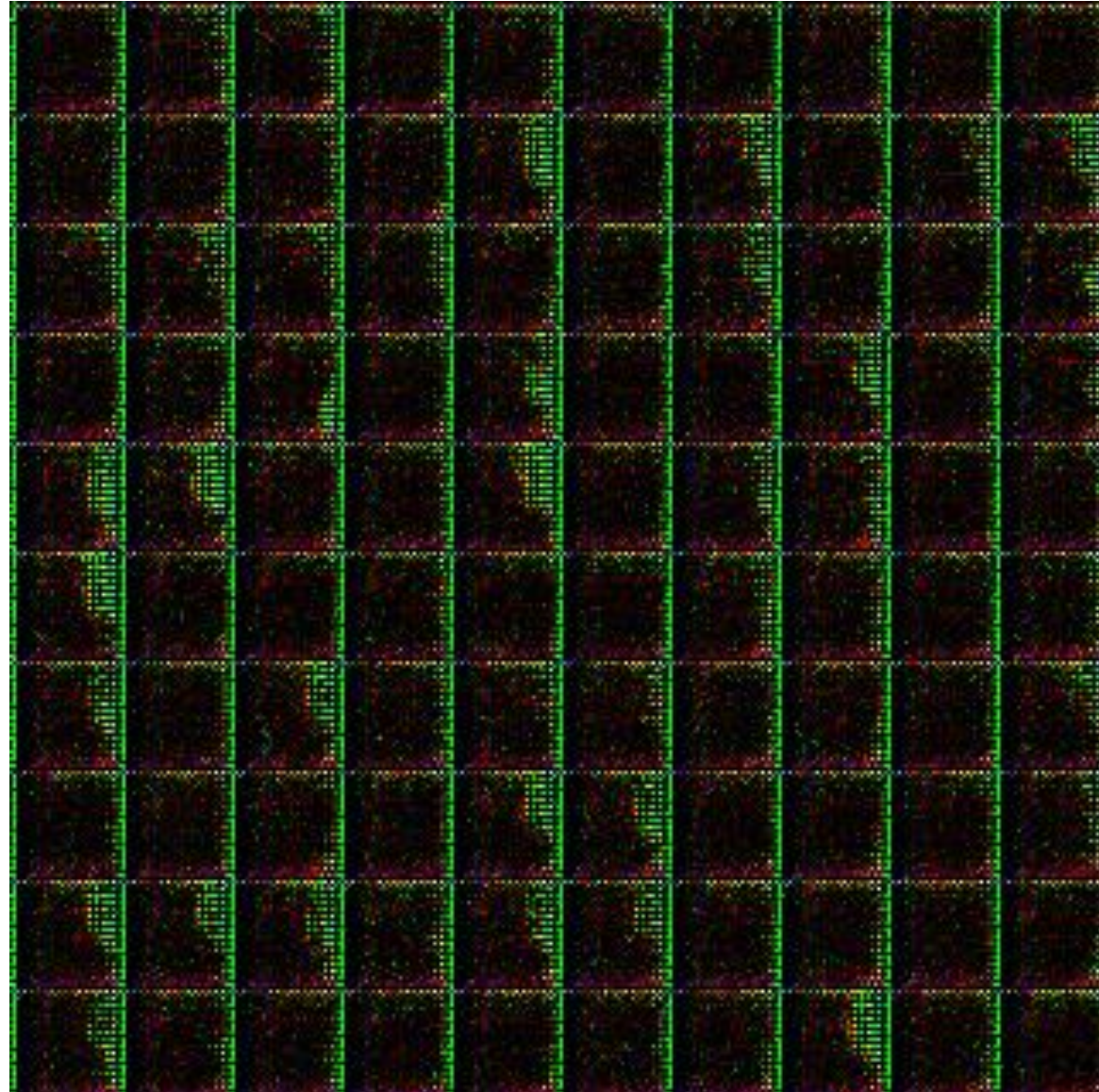
$$J^{(G)} = -J^{(D)} = \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} \log(1 - D(G(z^{(i)})))$$

"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"

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri

- Typical # of weights: 100 Million
- Typical training dataset: 20GB (e.g., Imagenet)
- Lots of code and resources available to get started
- Increasing integration with large language models



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()

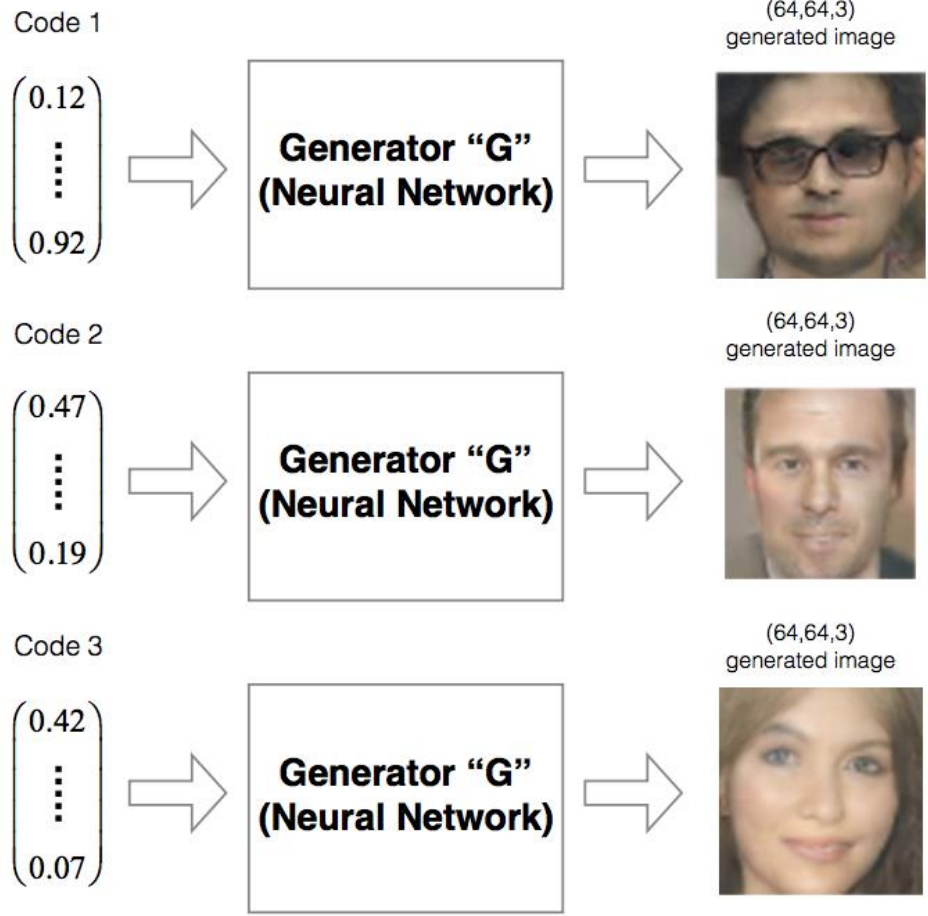
    model.add(Flatten(input_shape=self.img_shape))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation='sigmoid'))
    model.summary()

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

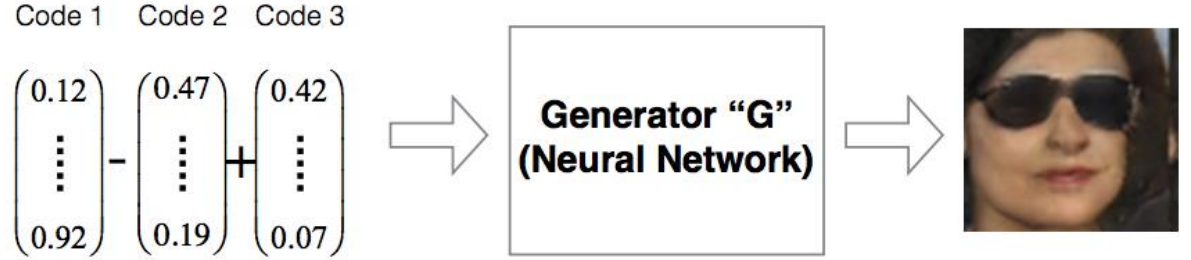
    return Model(img, validity)
```

II.E - Nice results

Operation on codes



Note: this is like the previously mentioned selection and navigation process!



Man with glasses - man + woman = woman with glasses

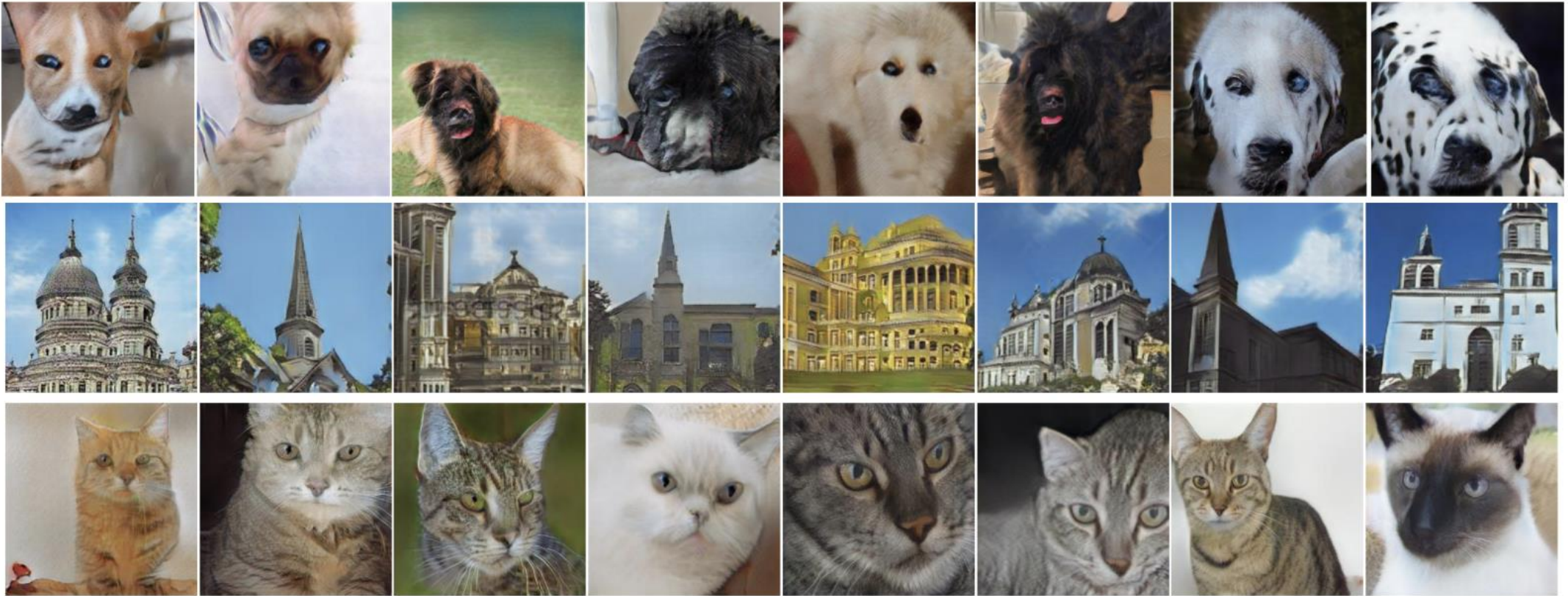
[Radford et al. (2015): UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS]

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



In 2024...

https://www.youtube.com/watch?v=HK6y8DAPN_0



In 2024...

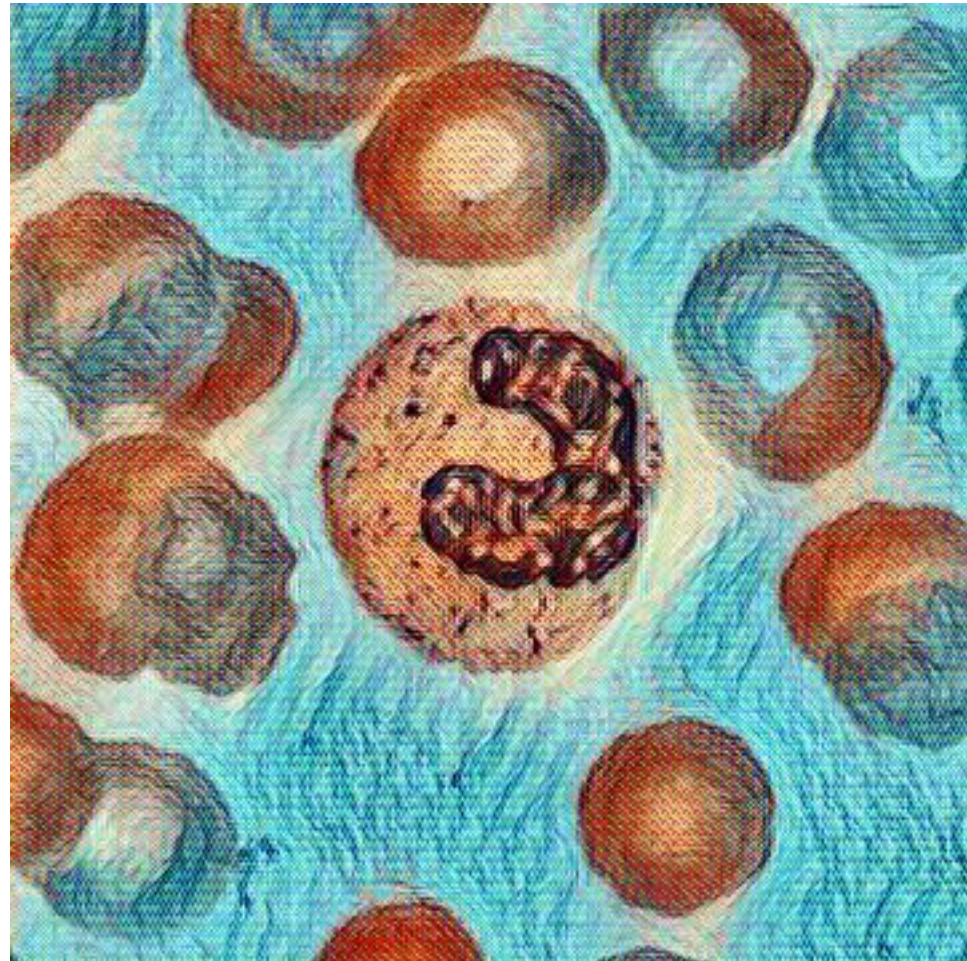
https://www.youtube.com/watch?v=HK6y8DAPN_0

In 2026: ***What Sora's End Says About OpenAI's Strategy***

The artificial intelligence start-up has pulled the plug on the video-making tool as it rethinks spending ahead of a possible blockbuster I.P.O.

Applications in biomedical/scientific imaging?

- Generating labeled examples when annotation is difficult/impossible
- Transferring images of one type to another (“style transfer”) for human viewing
- Increasing dataset sizes (augmentation, but better)



Domain Transfer and Virtual “Staining”/Fluorescence

CycleGANs

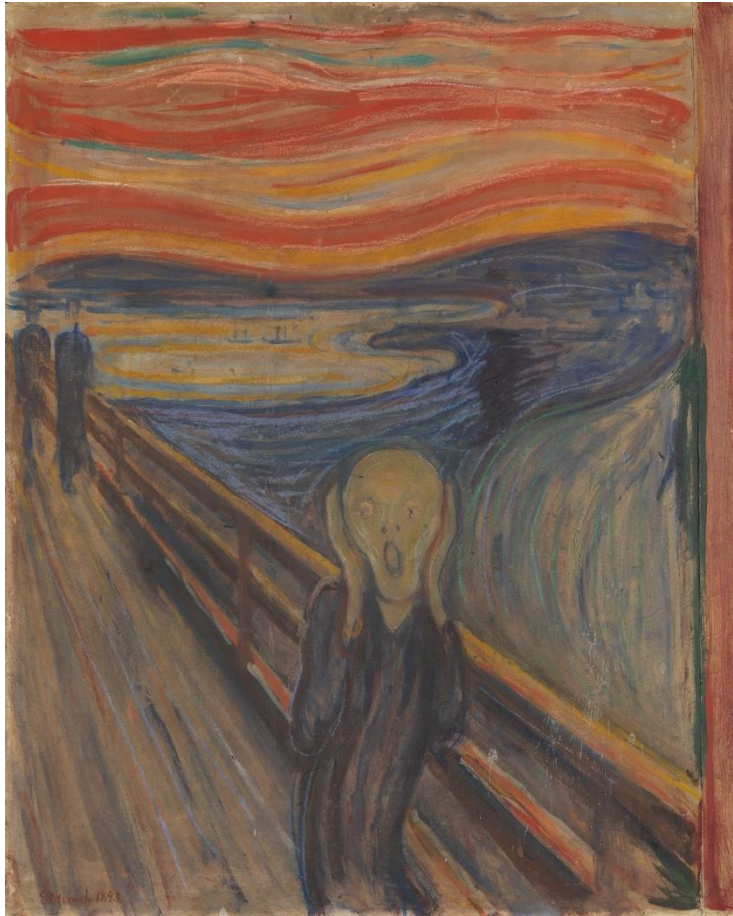


Picture 1

What is content in this picture?

What is the style?

CycleGANs



Picture 2

What is content in this picture?

What is the style?

CycleGANs

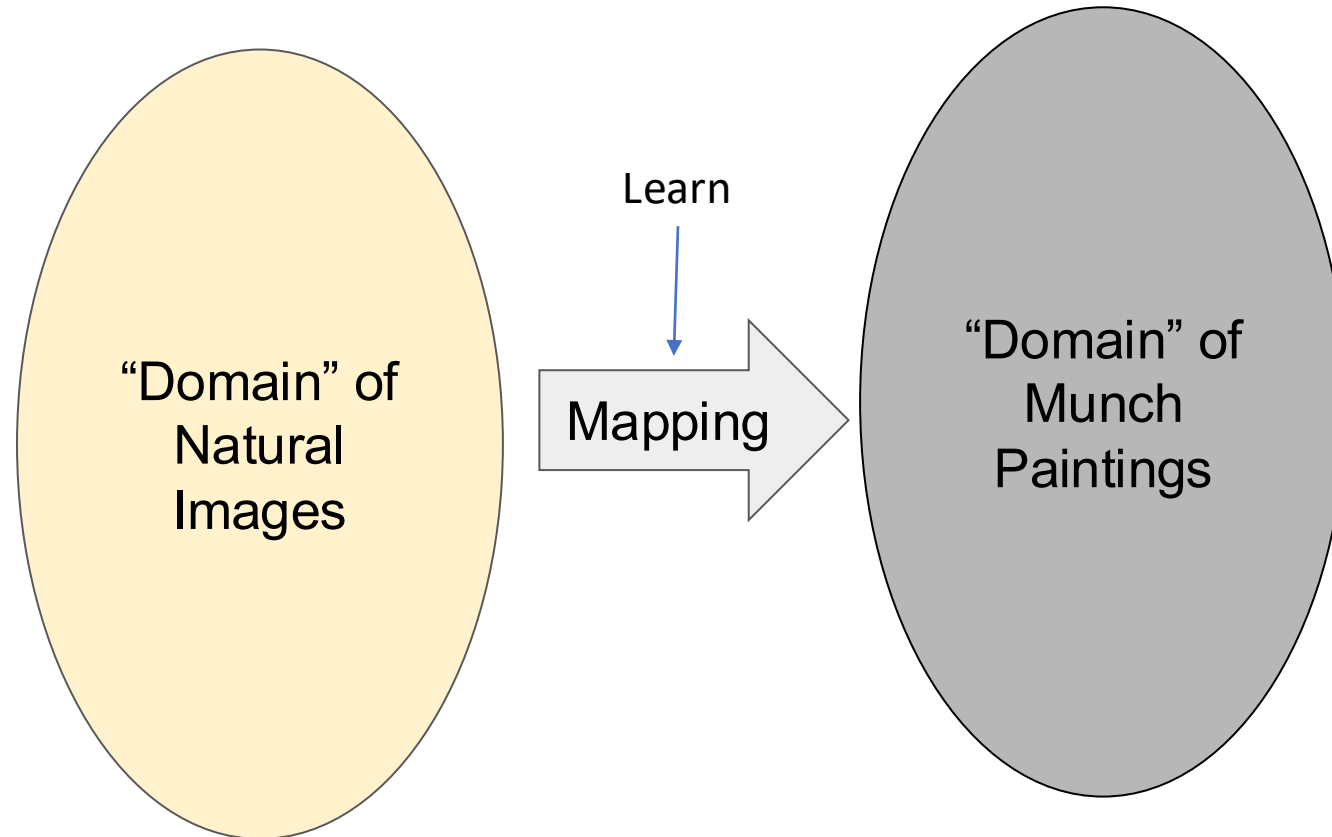
What would it look like if the landscape was painted like The Scream?

Or, what would the content from Picture 1 look like in the style of Picture 2?

We do not have paired data!

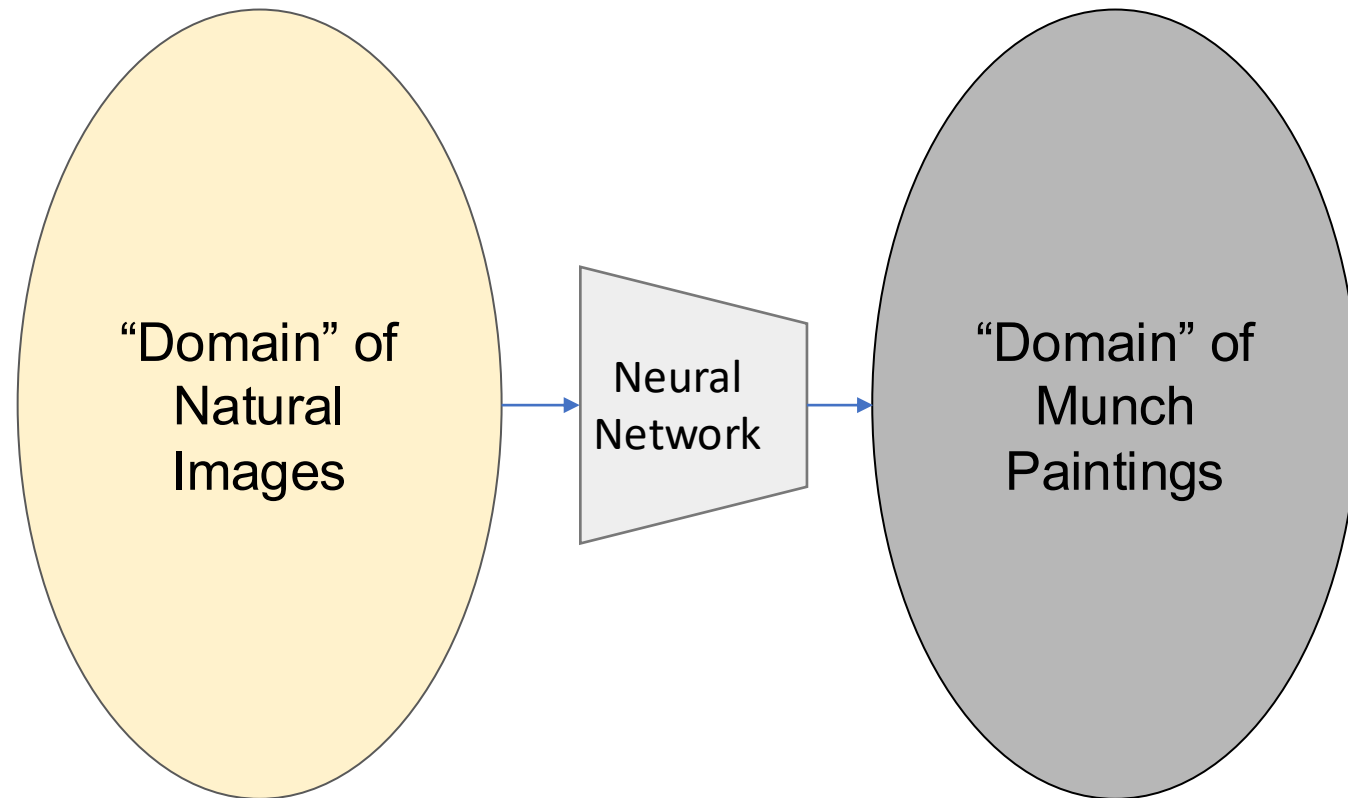


CycleGANs – how?



CycleGANs – how?

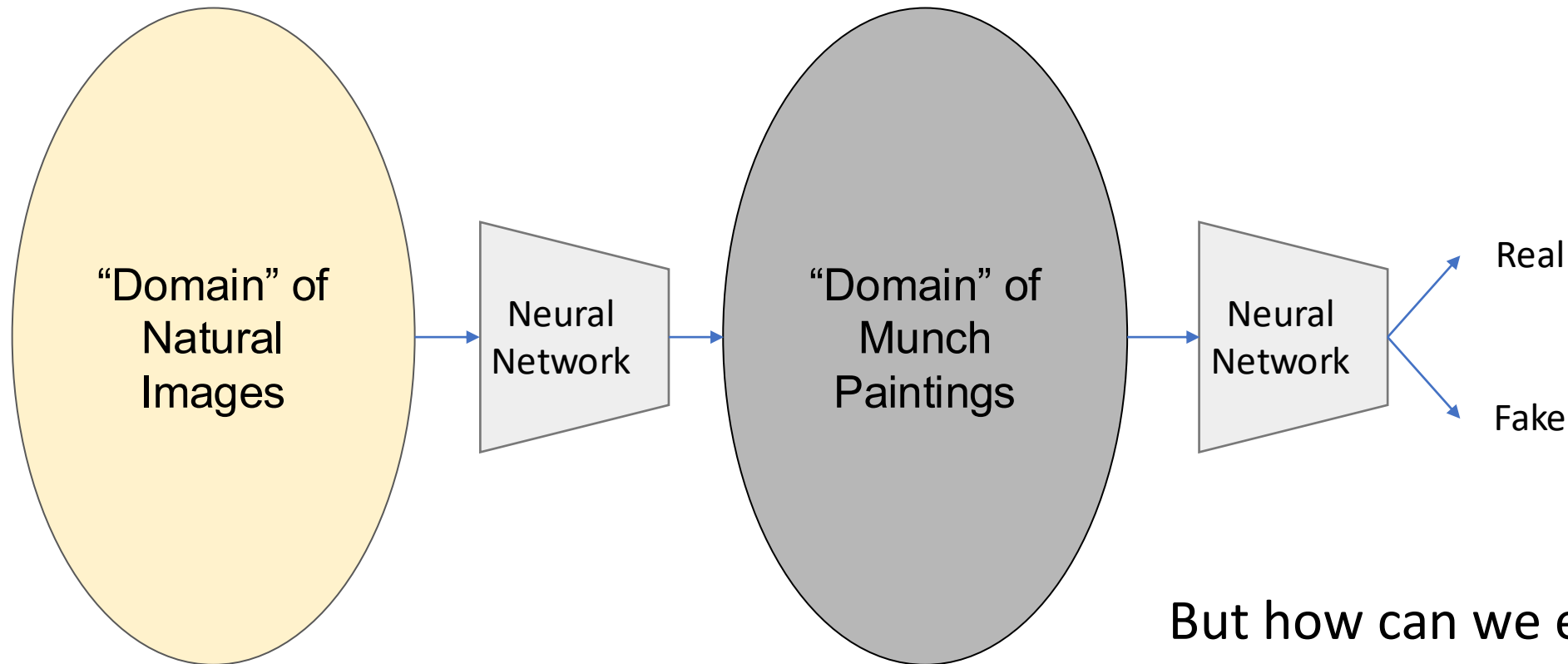
We can learn a mapping with supervised learning...



But that needs a lot of paired data!

CycleGANs – how?

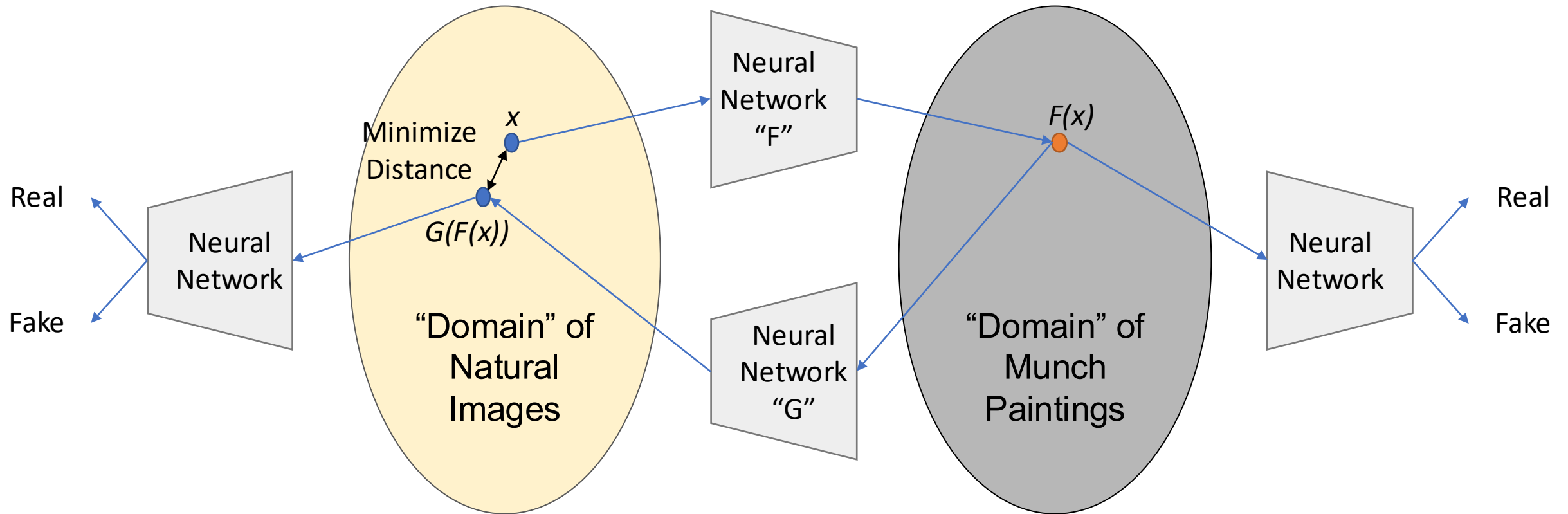
We could use a GAN instead:



But how can we ensure that the content is the same?

CycleGAN

Use a second GAN!

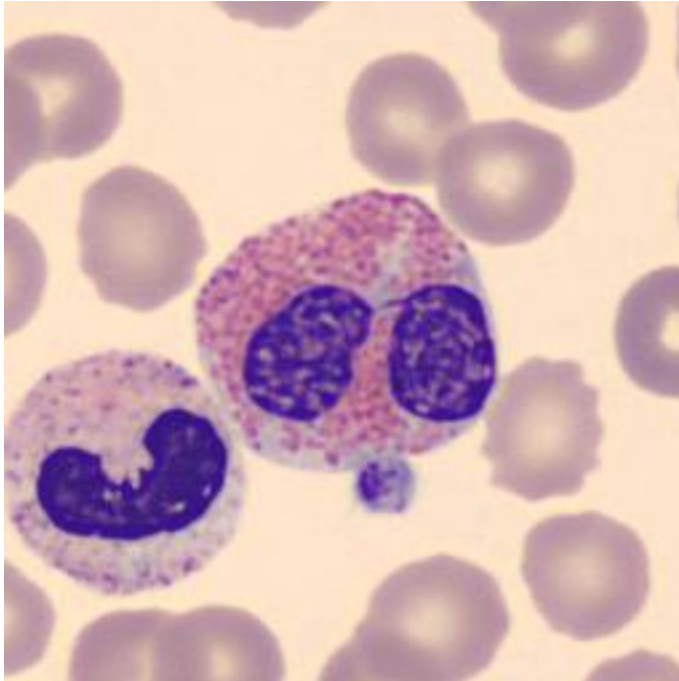


CycleGAN

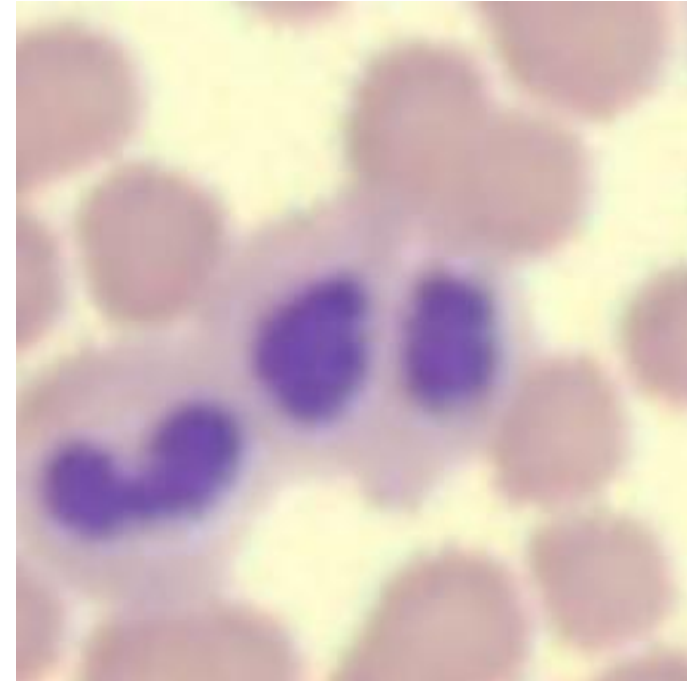
- Learn a forward mapping F and reverse mapping G
- Make sure that $G(F(x))$ is very close to x – **Imposes content preservation**
- Use discriminator with $F(x)$ and known style image – **Imposes Style similarity**

Where is this useful? “Lossy learning” – let’s make use of annotated images online!

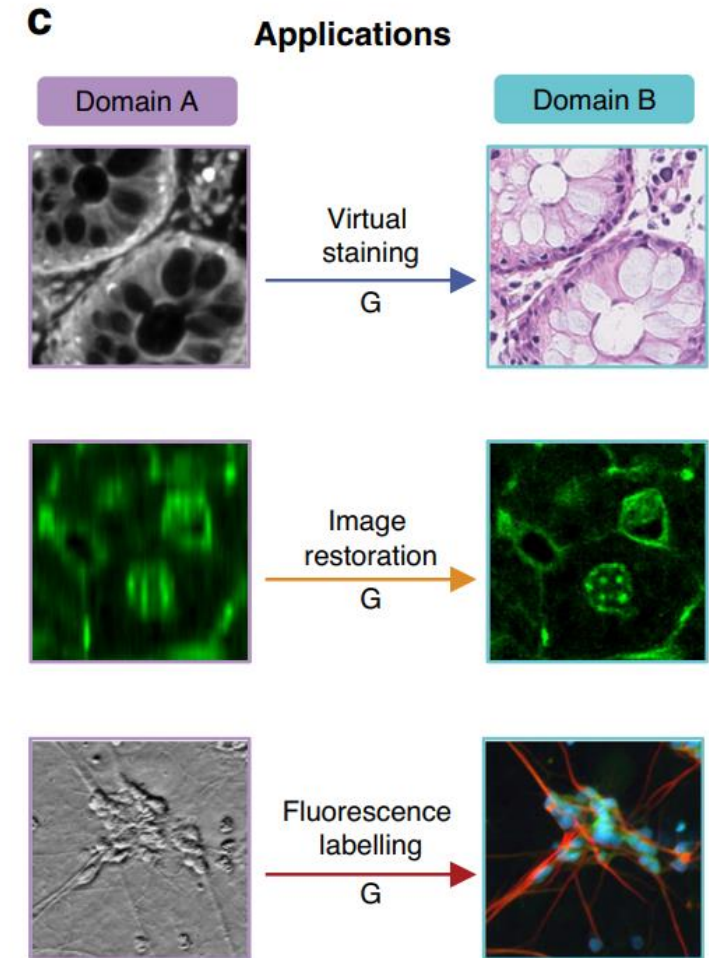
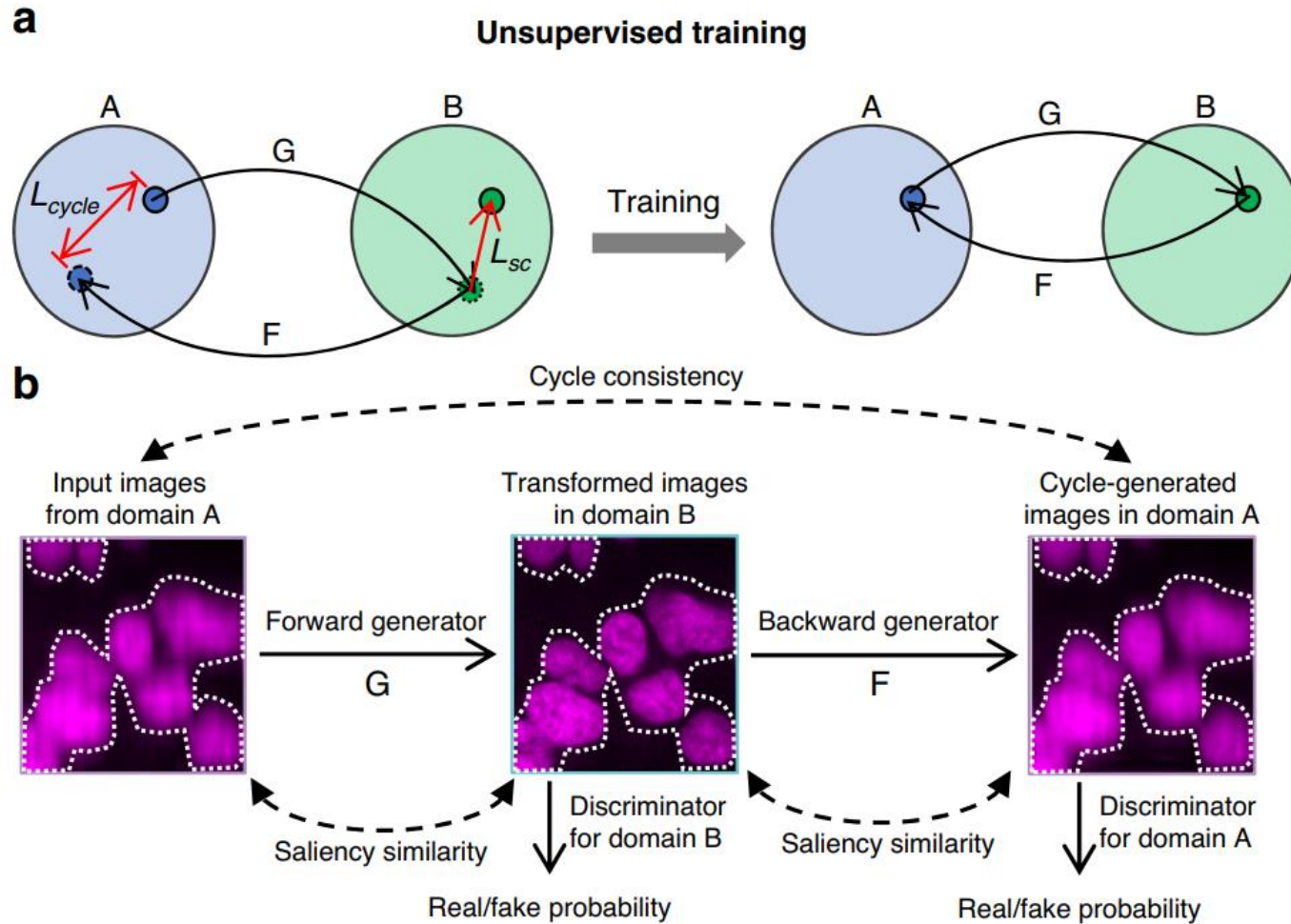
Easy to annotate



What my data looks like....

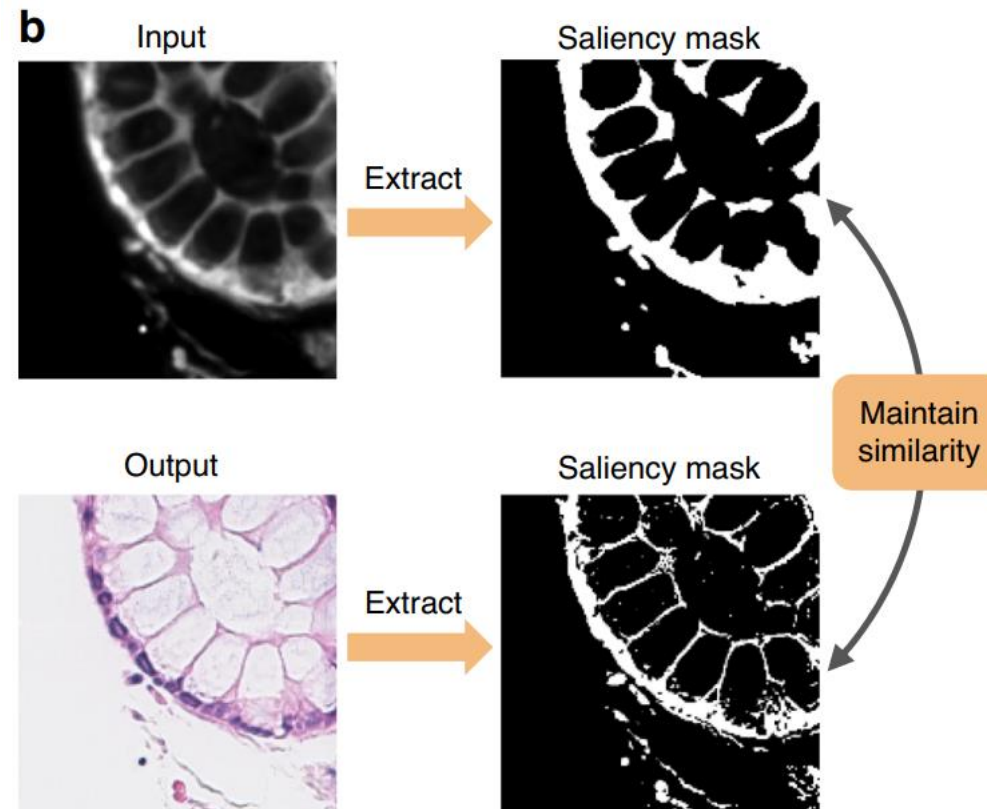


Application in Biomedical Context

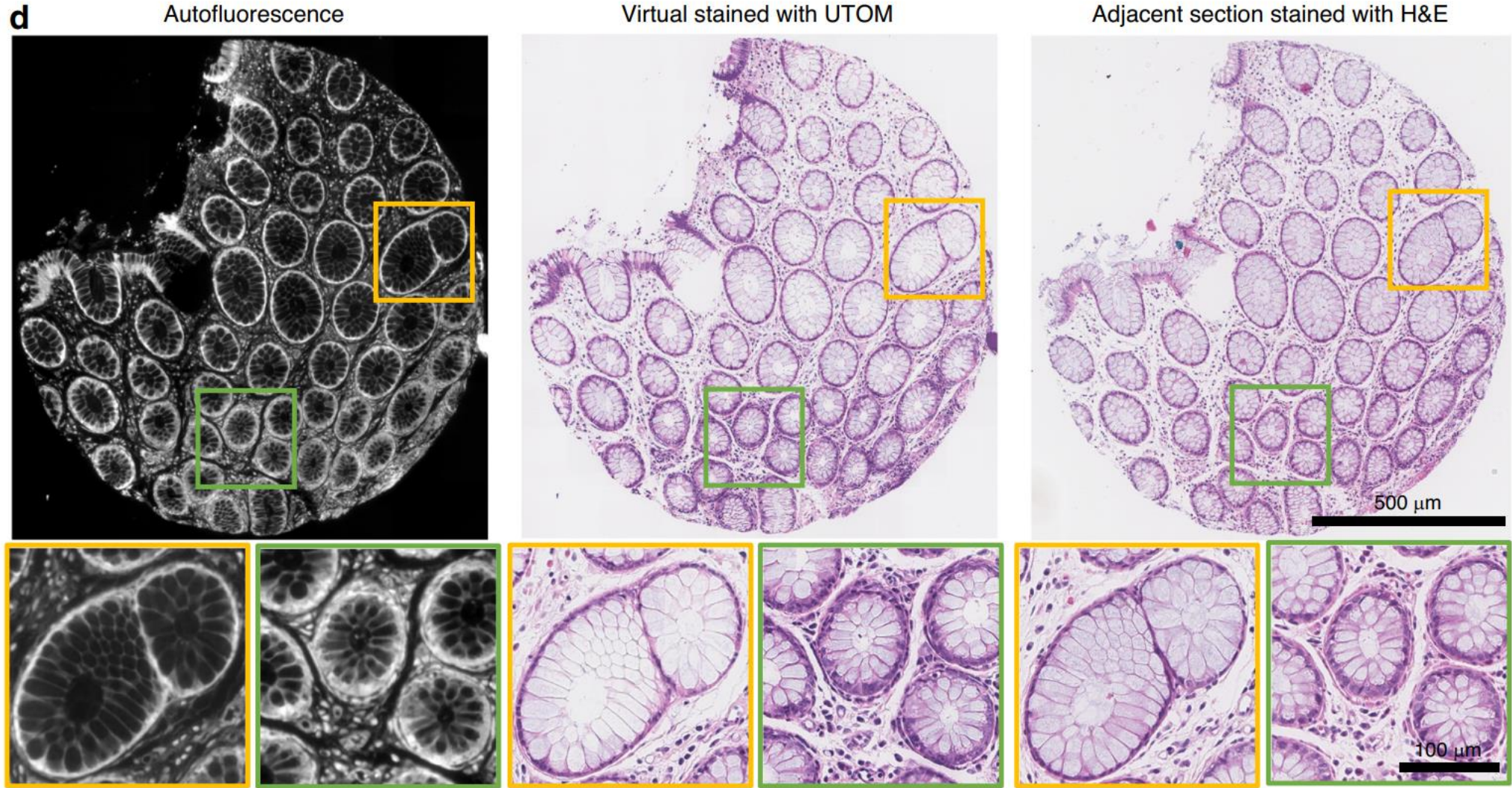


Additional Step

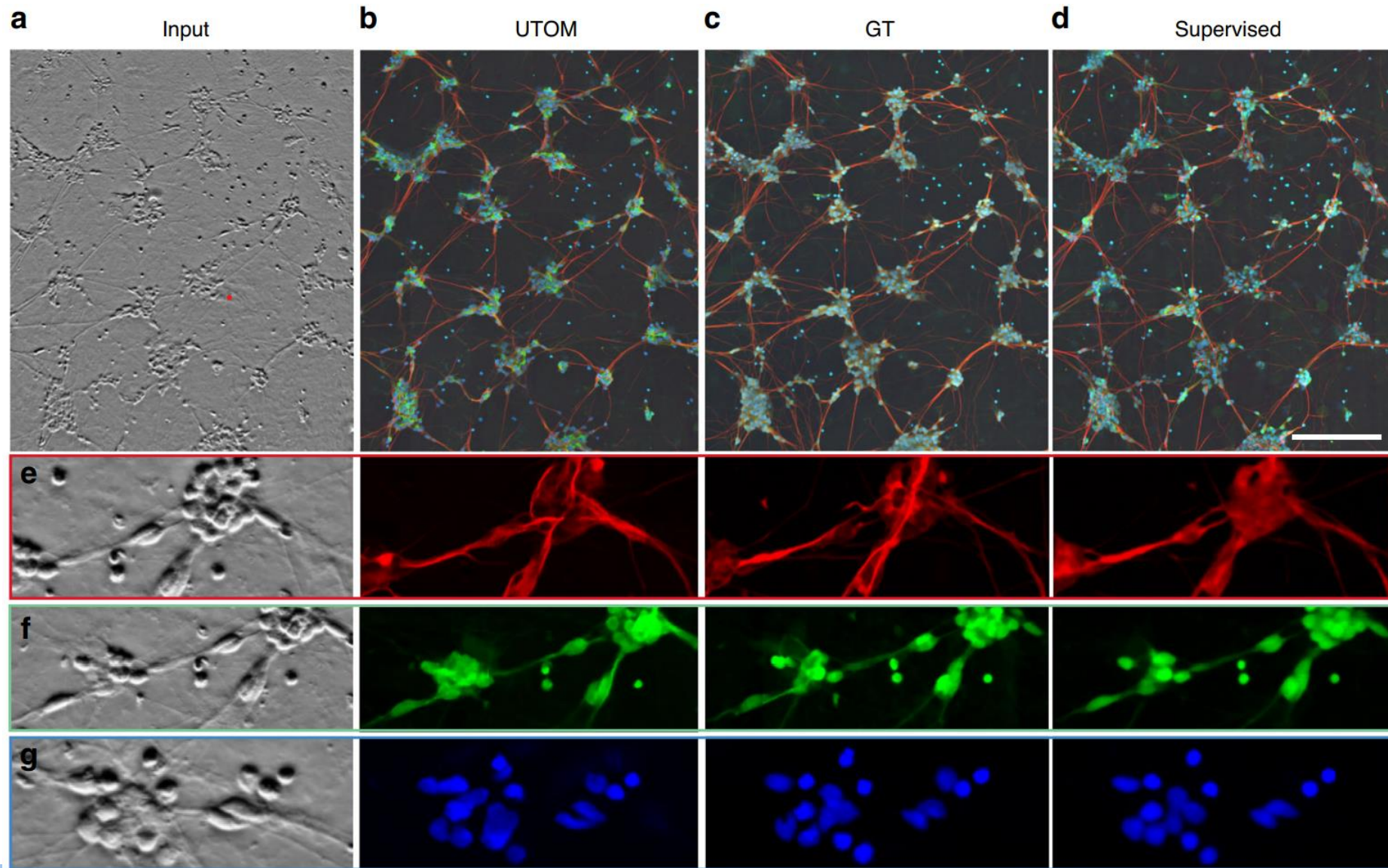
- In order to strongly impose content preservation, minimize saliency on input and generated images



Results – Autofluorescence to H&E



Results – Virtual Fluorescence



CycleGANs and Domain Adaptation

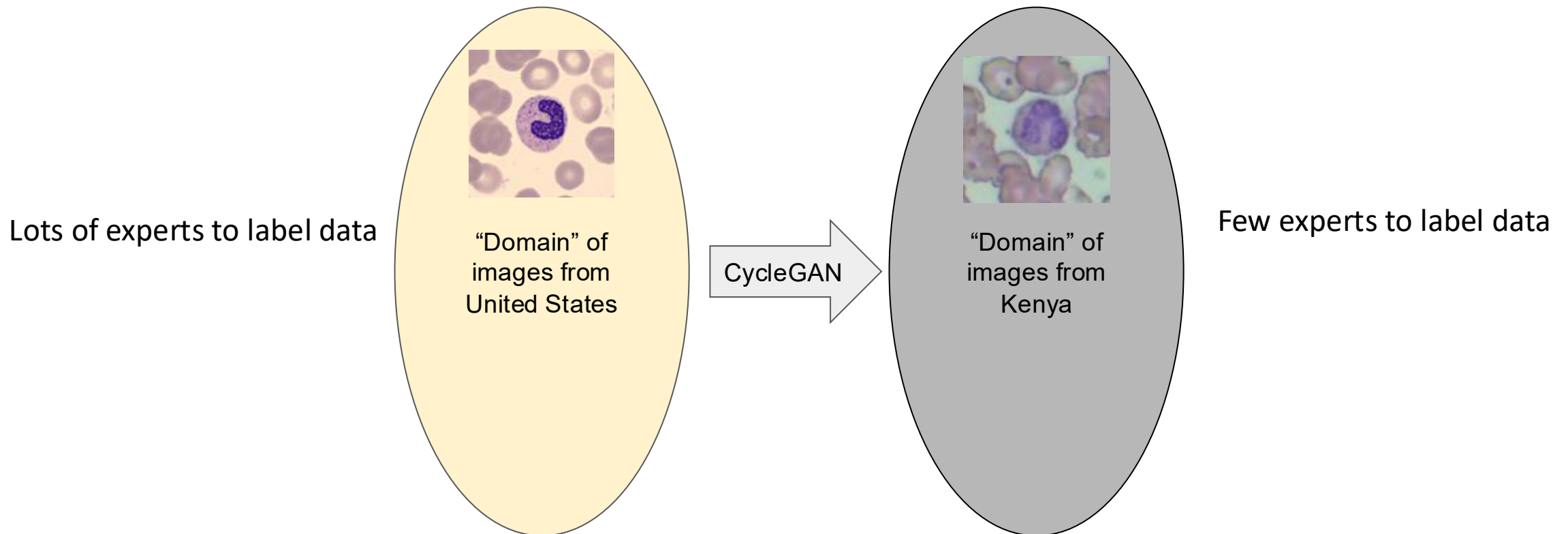
Machine Learning needs a ton of labeled data that is expensive to obtain

Machine Learning only works if it used on data that is similar to training data

Existing labeled data cannot be used if we develop a new imaging system

CycleGANs and Domain Adaptation

We could solve this problem by using CycleGANs to make the available data look like the data we want to learn



CycleGANs and Domain Adaptation

Using different source domains for the learning process of an algorithm, which is then applied to a different but related target domain is called **domain adaptation**.

