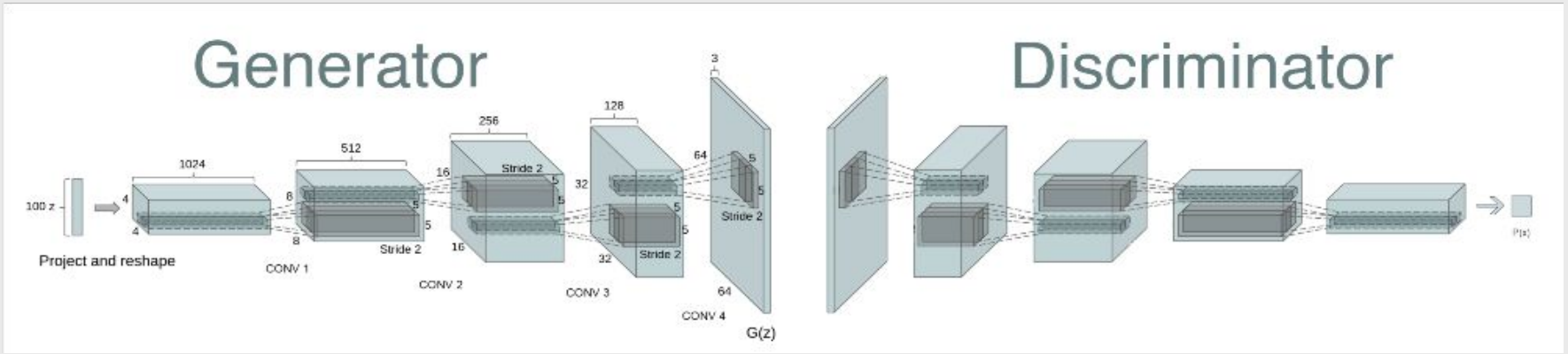
[PyData Berlin October meetup]

Deep Convolutional Network and Deep Generative Networks to Automate Industrial Manufacturing Applications



Slides availables at: https://docs.google.com/presentation/d/1pNif34jNv_RUH2caHDVVhVzJGhfW_5QL6DIuQEQDjhc/

Jonathan DEKHTIAR - @Born2Data

about:myself – Jonathan Dekhtiar

26 years old

I am French Food/Wine lover

Data Scientist Engineer

PhD Student in Deep Learning applications - UTC (FR) – EPFL (CH)

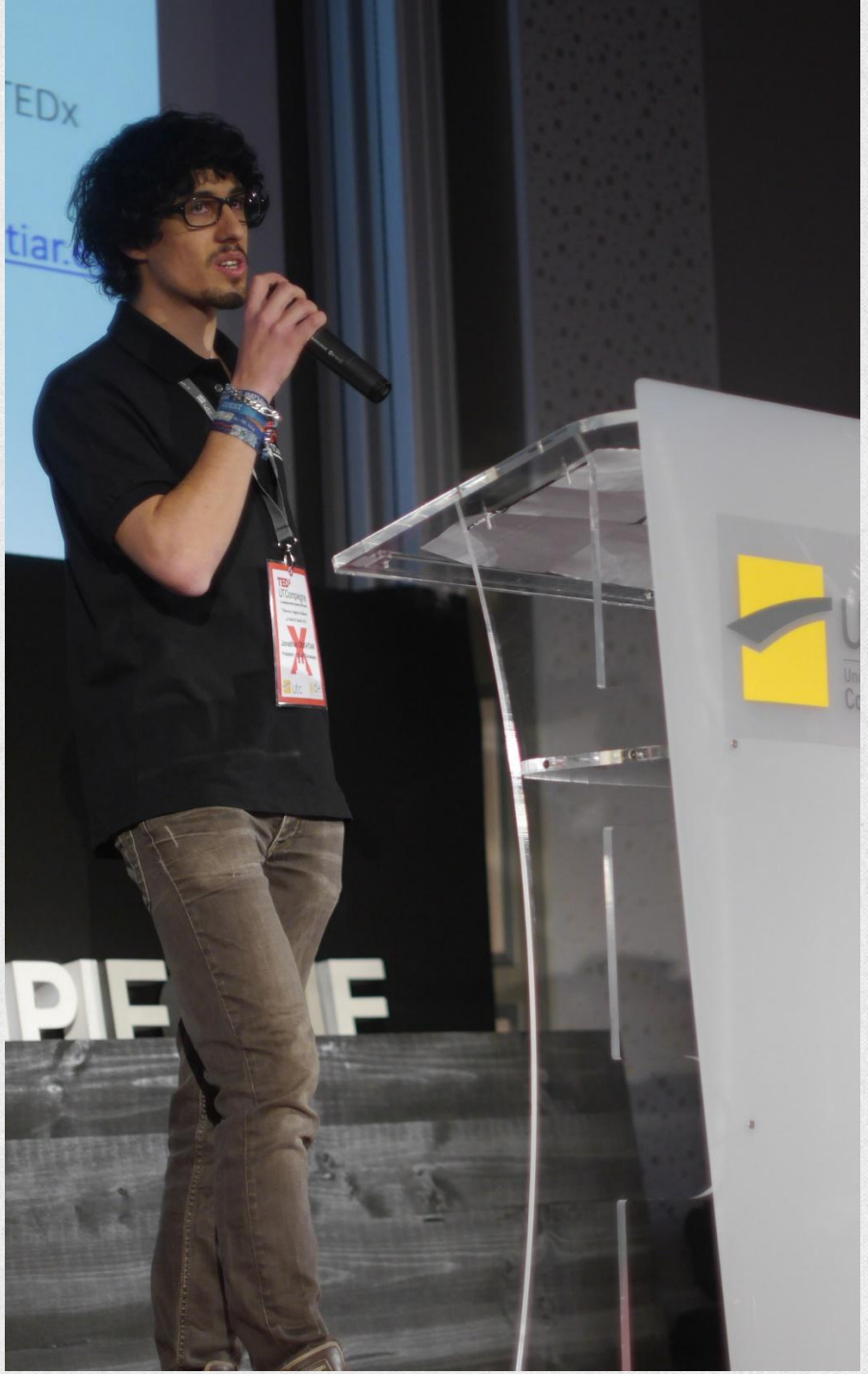
I try to automate & apply DL technologies in the manufacturing field.

Twitter: @Born2Data

Github: https://www.github.io/DEKHTIARJonathan

Email: contact@jonathandekhtiar.eu

TensorFlow™ Keras



What's on the menu today ?

1. Quick introduction: Data Science & Manufacturing Industry
2. Supervised Learning and an open dataset targeted manufacturing industry
3. Unsupervised Learning and Generative Adversarial Learning
4. Why and How could you use a GAN to detect novelty/defects ?
5. A few tips n' tricks to fine-tune a GAN

1. Data Science and Manufacturing Industry – Easy fit ?

- A few major challenges -

Data Heterogeneity and Data Complexity

** A wide variety of (proprietary) formats.

** Engineering files are way more complex than Images/Videos/Sound.

Difficulties to collect large volumes of data

** The industry have very complex data in small quantities (opposite of GAFAM)

** Collecting just a little data can cost a lot of money.

** Intellectual Property issues: let's negociate!

Reduced System Inter-Operability

System were not designed to communicate outside of closed proprietary environment.

Very Expensive Licenses

You need to pay for very $$$ developers licenses ($30k for one developer license in C++ with Catia)

2. Supervised Learning and opening a CV dataset

Supervised, Unsupervised or Reinforcement Learning ?

**** SL is often not possible.**

⇒ domain-experts are paid a lot of €€€

⇒ domain-experts are hardly available

**** RL approaches are very interesting**

⇒ take into account the domain-expert feedback

⇒ model fine-tuning over time

**** UL approaches present some appealing advantages**

⇒ does not require an expensive expert

⇒ cheaper

⇒ easier to develop and put into production

2. Supervised Learning and opening a CV dataset

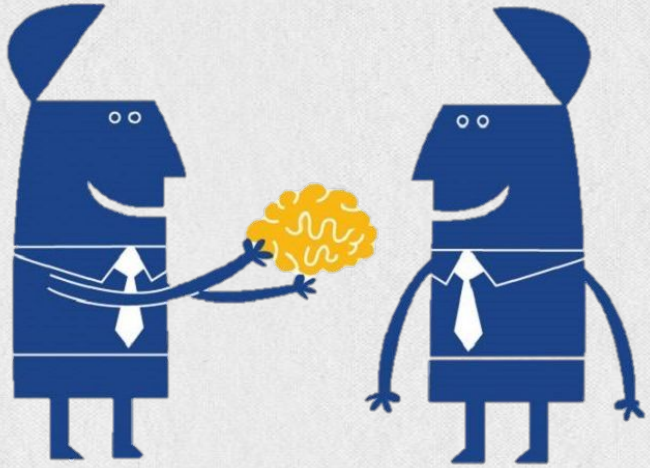
Supervised, Unsupervised or Reinforcement Learning ?

** **SL is often not possible.**

⇒ domain-experts are paid a lot of €€€

⇒ domain-experts are hardly available

Transfer Learning will be your answer to many situation



** **RL approaches are very interesting**

⇒ take into account the domain-expert feedback

⇒ model fine-tuning over time

** **UL approaches present some appealing advantages**

⇒ does not require an expensive expert

⇒ cheaper

⇒ easier to develop and put into production

2. Supervised Learning and opening a CV dataset

DMU-Net – Open Dataset for the Manufacturing Industry

- https://www.dmu-net.org -

Actuator → 37	AirFilter → 32	Battery → 46	Bearing → 128	Brake → 39
Camshaft → 46	Clutch → 61	Cooler → 30	Coupling → 43	Crankshaft → 81
CuttingTool → 41	ElectricMotor → 142	ElectronicBoard → 37	ExhaustPipe → 20	Gear → 18
GearWheel → 91	Helix → 60	Nut → 56	Piston → 87	Pulley → 109
Pump → 34	Rod → 53	RodPistonAssembly → 56	Screw → 219	ShockAbsorber → 78
SparkPlug → 87	Switch → 87	Turbo → 18	Washer → 47	WingNut → 33

+/- 2000 CAD Models in STEP-AP242, JT, STL, ThreeJS, X3DOM and Object File (OBJ+MTL)

2. Supervised Learning and opening a CV dataset

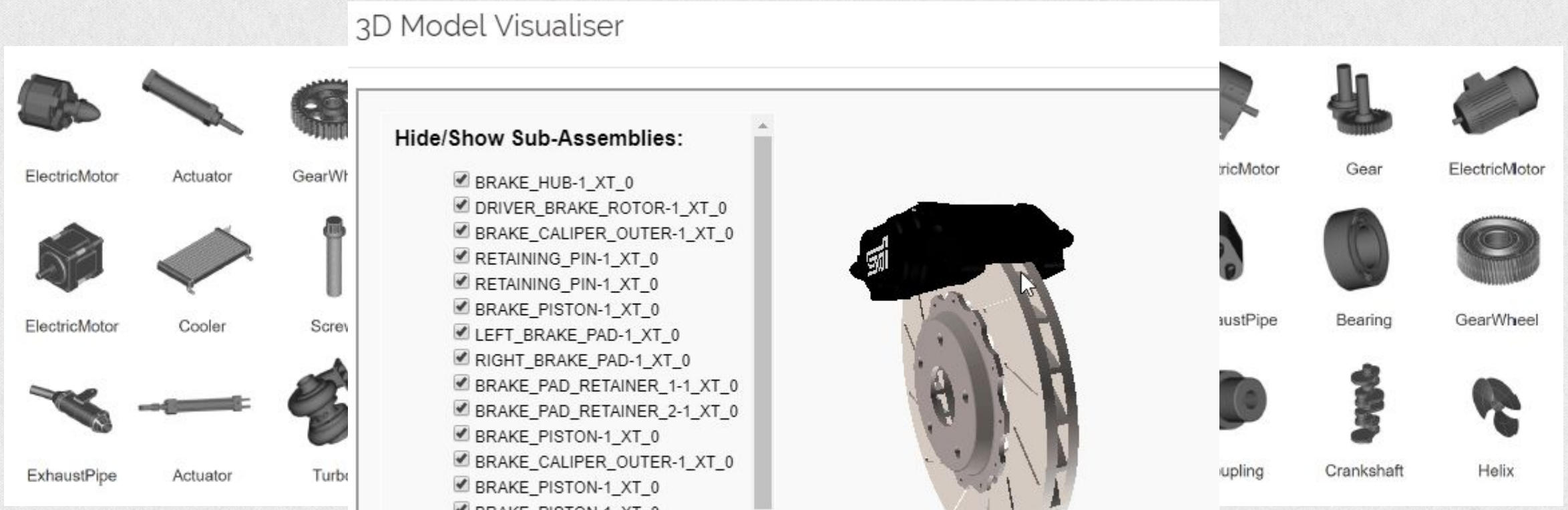
DMU-Net – Open Dataset for the Manufacturing Industry

- https://www.dmu-net.org -

2. Supervised Learning and opening a CV dataset

DMU-Net – Open Dataset for the Manufacturing Industry

- https://www.dmu-net.org -



3. Unsupervised Learning & Generative Adversarial Nets

Supervised, Unsupervised or Reinforcement Learning ?

**** SL is often not possible.**

⇒ domain-experts are paid a lot of €€€

⇒ domain-experts are hardly available

**** RL approaches are very interesting**

⇒ take into account the domain-expert feedback

⇒ model fine-tuning over time

**** UL approaches present some appealing advantages**

⇒ does not require an expensive expert

⇒ cheaper

⇒ easier to develop and put into production

Generative Adversarial Neural Networks are one of the possible directions.

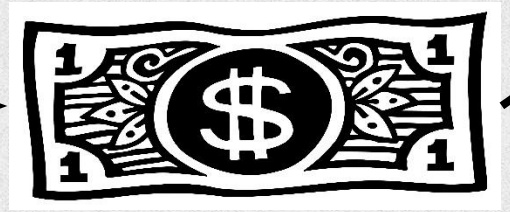
Let’s explore this possibility

3. Unsupervised Learning & Generative Adversarial Nets

Generative Adversarial Neural Networks

Real Data

Generator



Fake Data

Discriminator

P(x == real)

3. Unsupervised Learning & Generative Adversarial Nets

Minimax Game and Adversarial Training

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

The Generator G and the Discriminator D have opposite objectives:

⇒ If G wins then D looses and vice versa.

⇒ Each network tries to maximise his log likelihood.

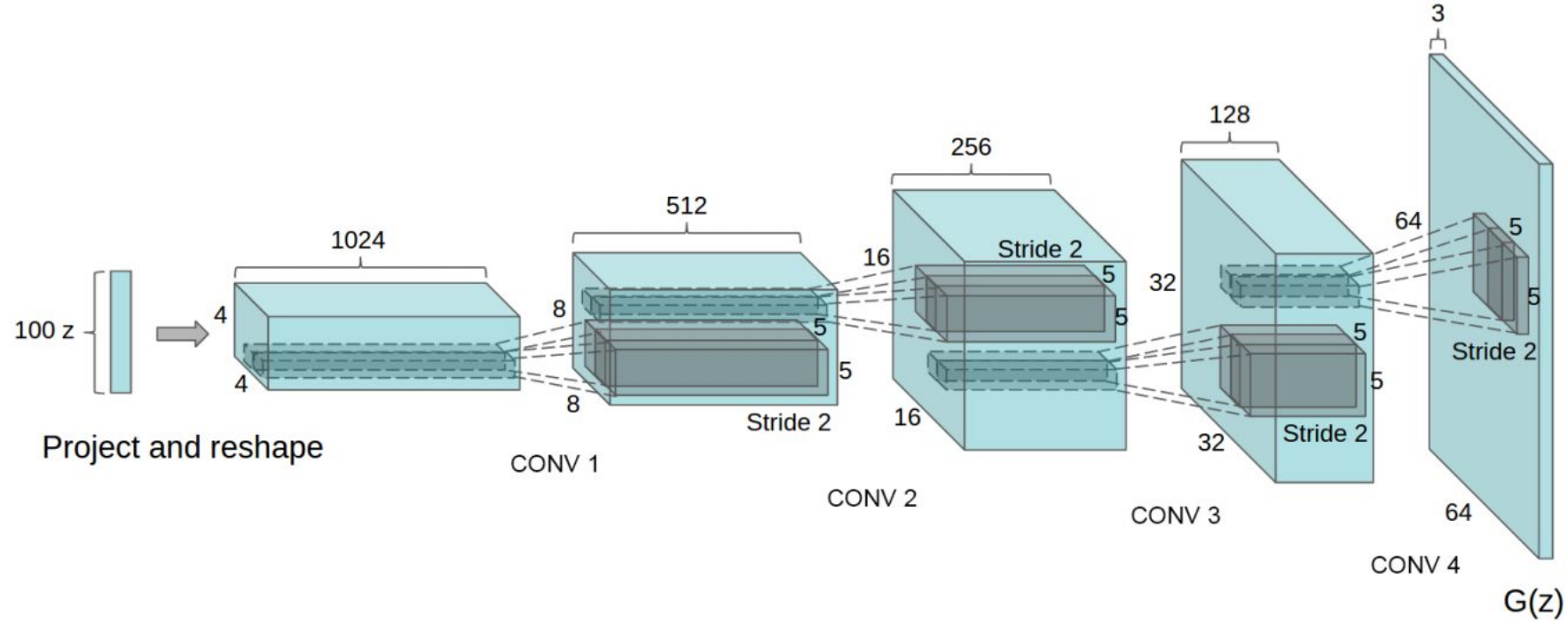
When the training converges, this point is called a ***Nash Equilibrium:***

⇒ D has 50% chance of getting its decision right *(fake or real data)*.

⇒ In other words, data produced are +/- indistinguishable from real data.

3. Unsupervised Learning & Generative Adversarial Nets

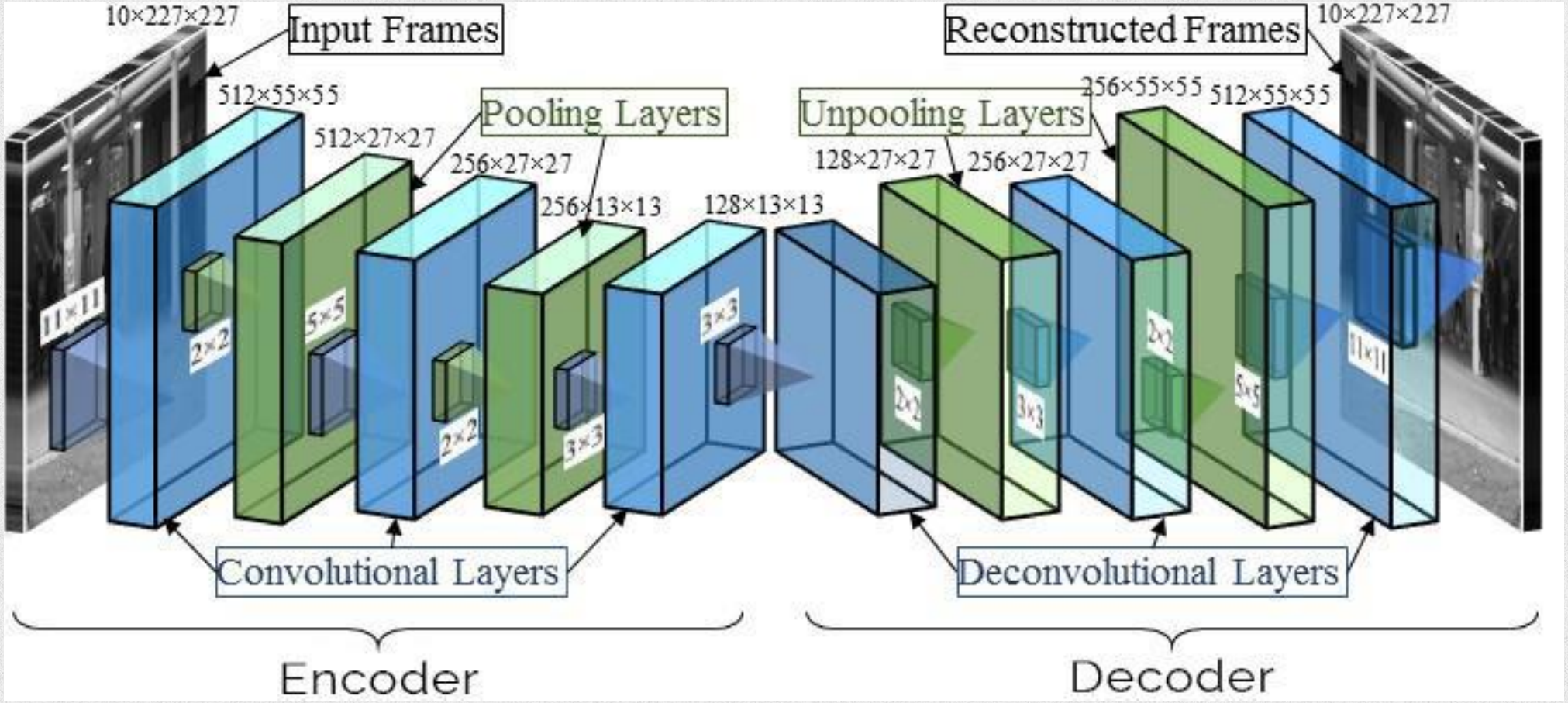
GAN and DCGAN



Generative Adversarial Networks – Goodfellow et al. 2014
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks – Radford et al. 2015

3. Unsupervised Learning & Generative Adversarial Nets

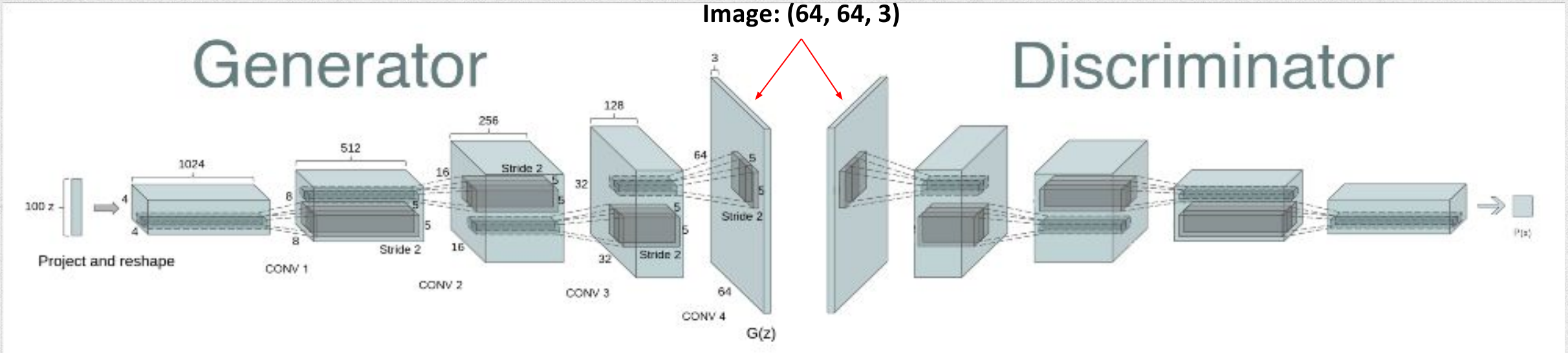
Convolution vs Strided-Convolutions – Example with an AE



Source: Learning Temporal Regularity in Video Sequences – Hasan et al. 2016

3. Unsupervised Learning & Generative Adversarial Nets

DCGAN Implementation



Network Settings		Training Settings	
Stride	= (2, 2)	Optimizer	= Adam
Kernel Size	= (5, 5)	Learning Rate	= 2e-4
Padding	= “same”	Beta1	= 0.5
Activation	= Leaky ReLU	Beta2	= 0.999
N_Filters	= [1024, 512, 256, 128]		
Latent vector shape	= (None, 100)		

3. Unsupervised Learning & Generative Adversarial Nets

DCGAN Implementation

Input

Shape: (None, 100)

Reshape + Dense
4 * 4 * 512 neurons

Shape: (None, 4, 4, 512)

Conv2DTranspose
512 filters, size=(5,5), stride=(2,2), padding='same'

Shape: (None, 8, 8, 512)

Conv2DTranspose
256 filters, size=(5,5), stride=(2,2) , padding='same'

Shape: (None, 16, 16, 256)

Conv2DTranspose
128 filters, size=(5,5), stride=(2,2) , padding='same'

Shape: (None, 32, 32, 128)

Conv2DTranspose
64 filters, size=(5,5), stride=(2,2) , padding='same'

Shape: (None, 64, 64, 64)

Conv2DTranspose
1 filters, size=(5,5), stride=(1,1) , padding='same'

Shape: (None, 64, 64, 1)

Generator

Discriminator

Input

Shape: (None, 64, 64, 1)

Conv2D
64 filters, size=(5,5), stride=(2,2), padding='same'

Shape: (None, 32, 32, 64)

Conv2D
128 filters, size=(5,5), stride=(2,2) , padding='same'

Shape: (None, 16, 16, 128)

Conv2D
256 filters, size=(5,5), stride=(2,2) , padding='same'

Shape: (None, 8, 8, 256)

Conv2D
512 filters, size=(5,5), stride=(2,2) , padding='same'

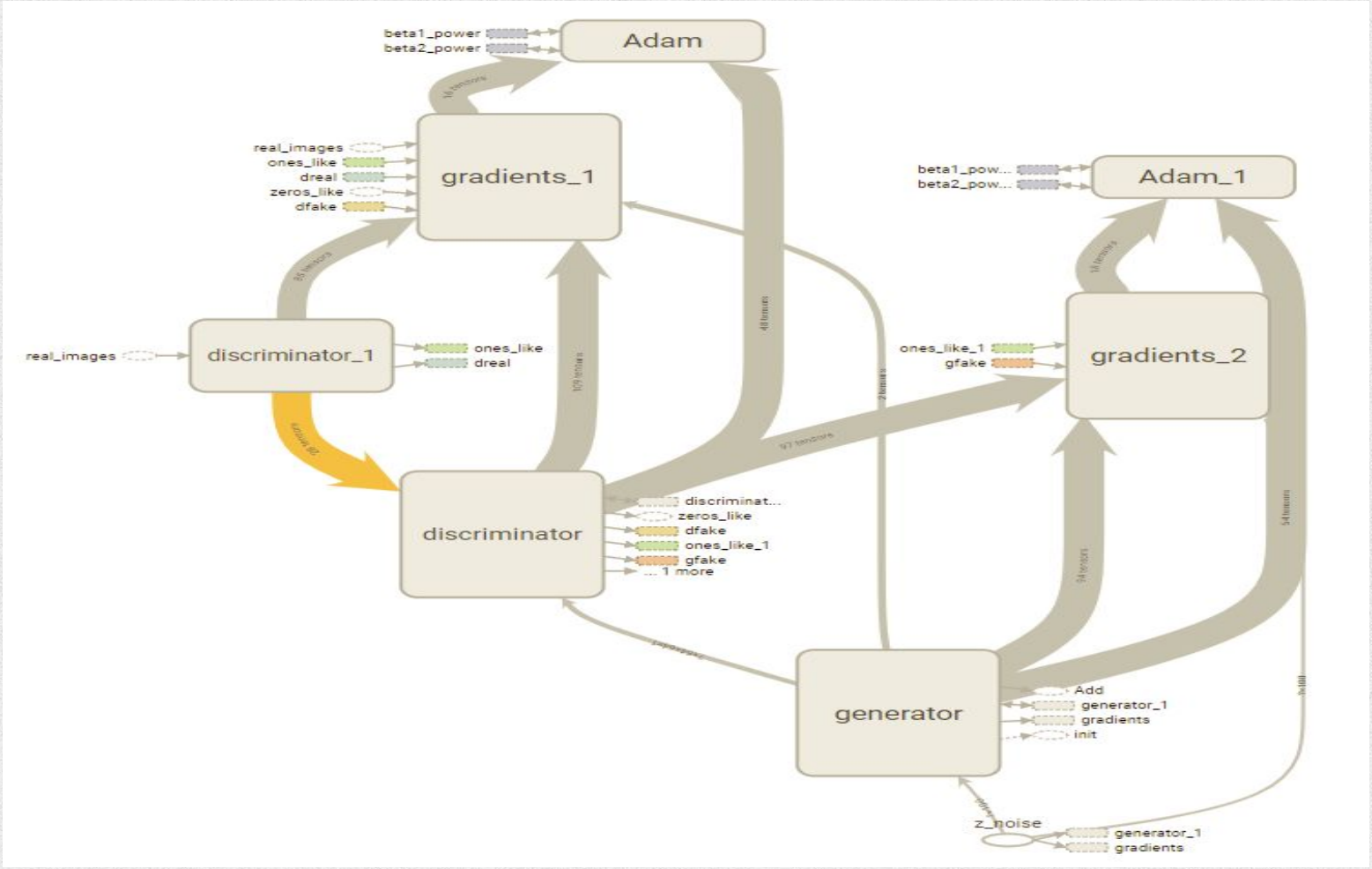
Shape: (None, 4, 4, 512)

Flatten + Dense
1 neuron

Shape: (None, 1)

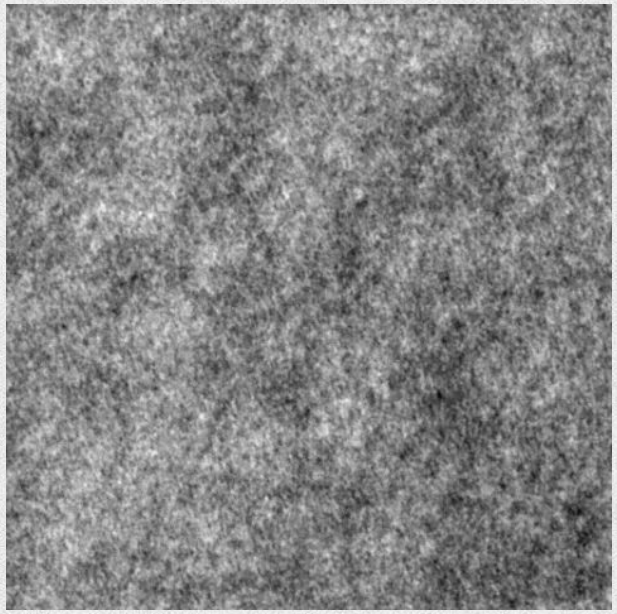
3. Unsupervised Learning & Generative Adversarial Nets

TensorBoard view of DCGAN

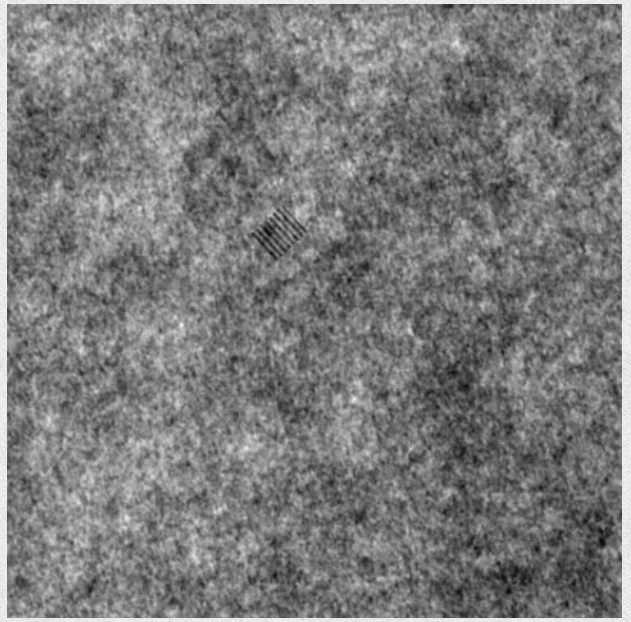


4. How did I use GANs to detect novelty/defects ?

Dataset used for this experiment



Healthy Machined Surface

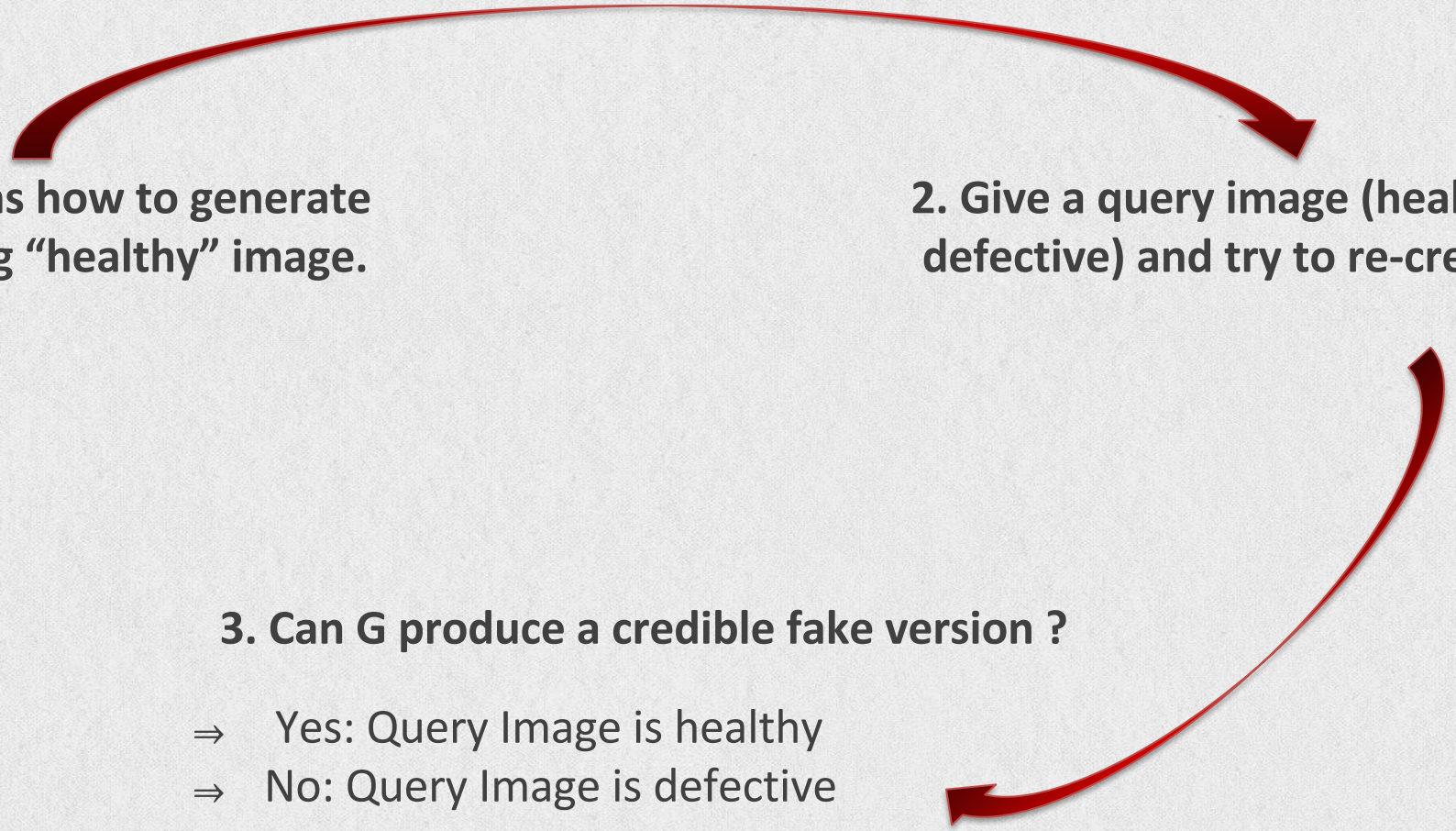


Defective Machined Surface

Toy Dataset Provided by:
German Association for Pattern Recognition http://resources.mpi-inf.mpg.de/conferences/dagm/2007/

4. How did I use GANs to detect novelty/defects ?

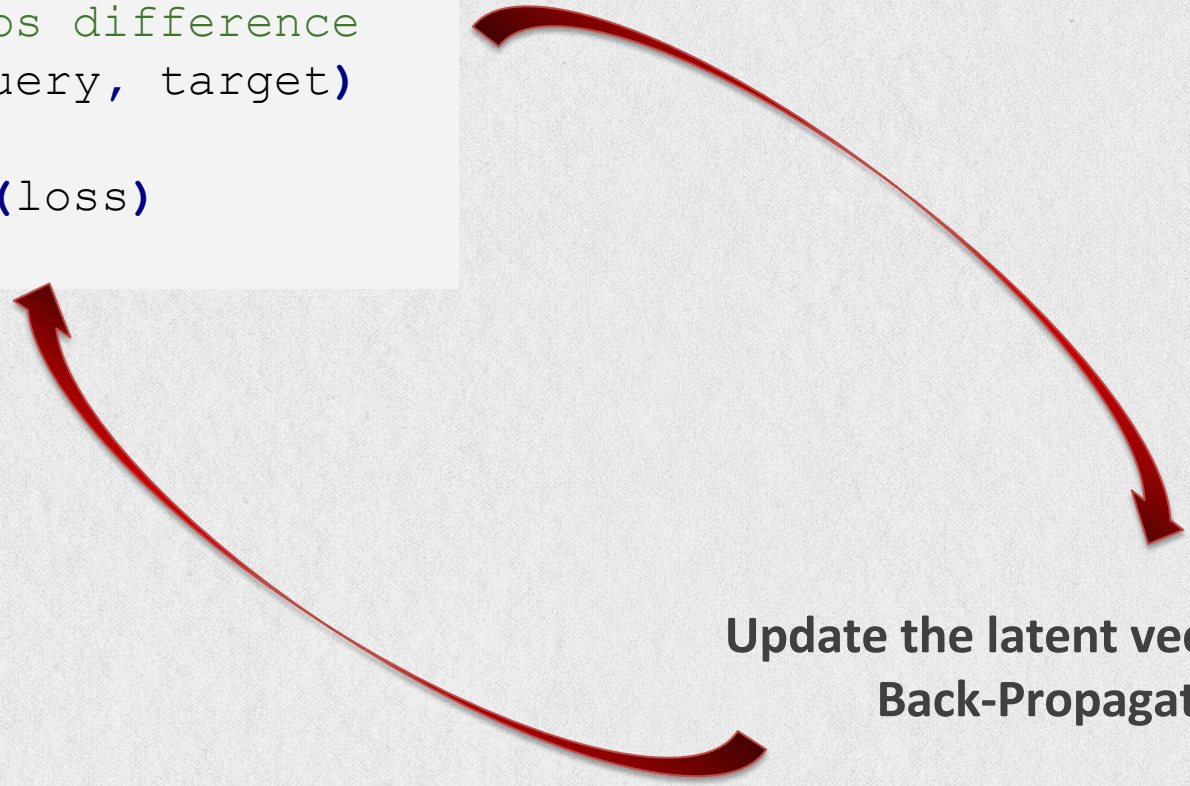
How GANs can help detecting defects / novelty / unusual situations



4. How did I use GANs to detect novelty/defects ?

How can I control my GAN to produce the closest image as possible ?

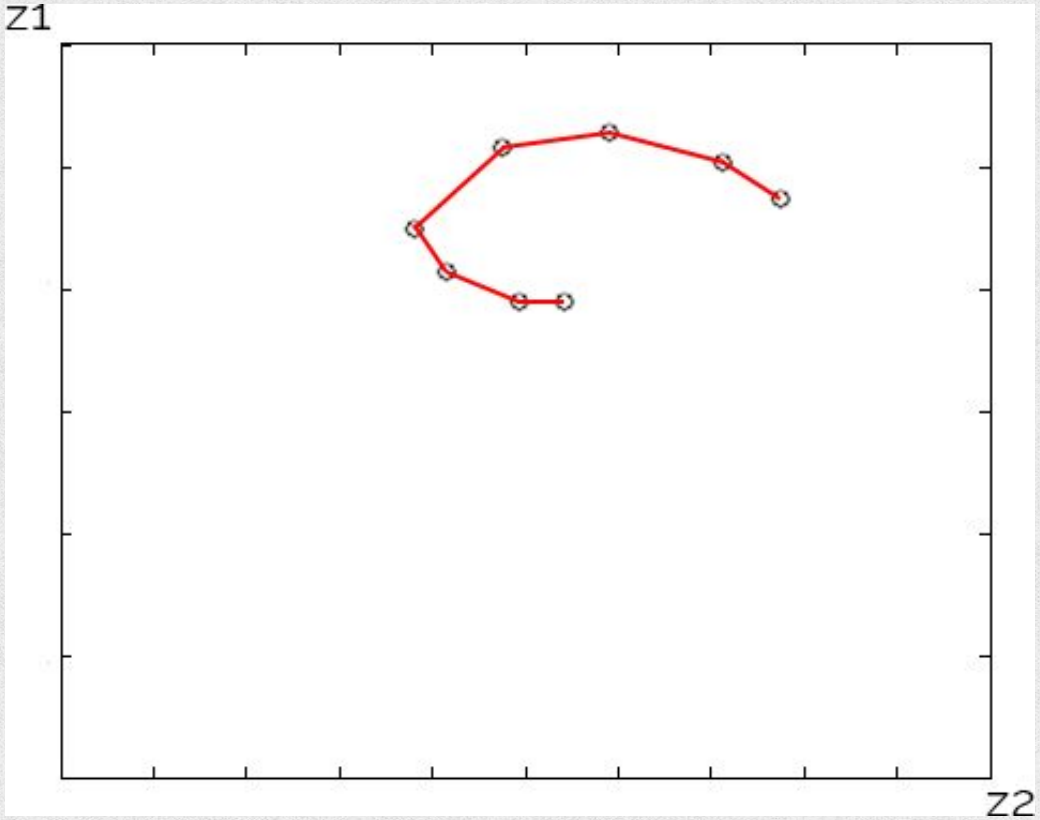
```
def reconstruction_loss(query, target):
    # Sum over pix2pix abs difference
    loss = tf.subtract(query, target)
    loss = tf.abs(loss)
    loss = tf.reduce_sum(loss)
    return loss
```



Update the latent vector Z with Back-Propagation

4. How did I use GANs to detect novelty/defects ?

Back-Propagate over Z during … Evaluation



**** Perform Back-Propagation during evaluation:**

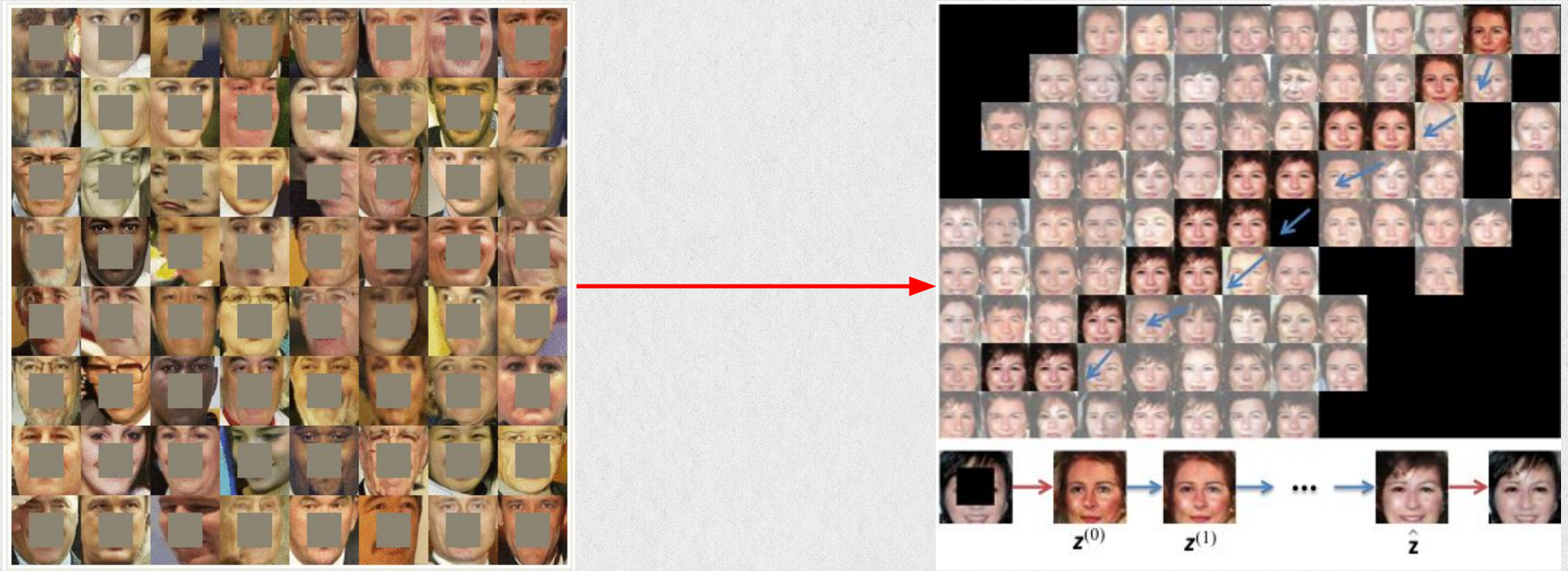
⇒ Try to generate the closest looking image by back-propagating through the input layer: Z.

⇒ Depending on the final loss value:
*** Low Value:* Healthy
*** High Value:* Defective

Inspired By: Semantic Image Inpainting with Deep Generative Models – Yeh et al. 2016

4. How did I use GANs to detect novelty/defects ?

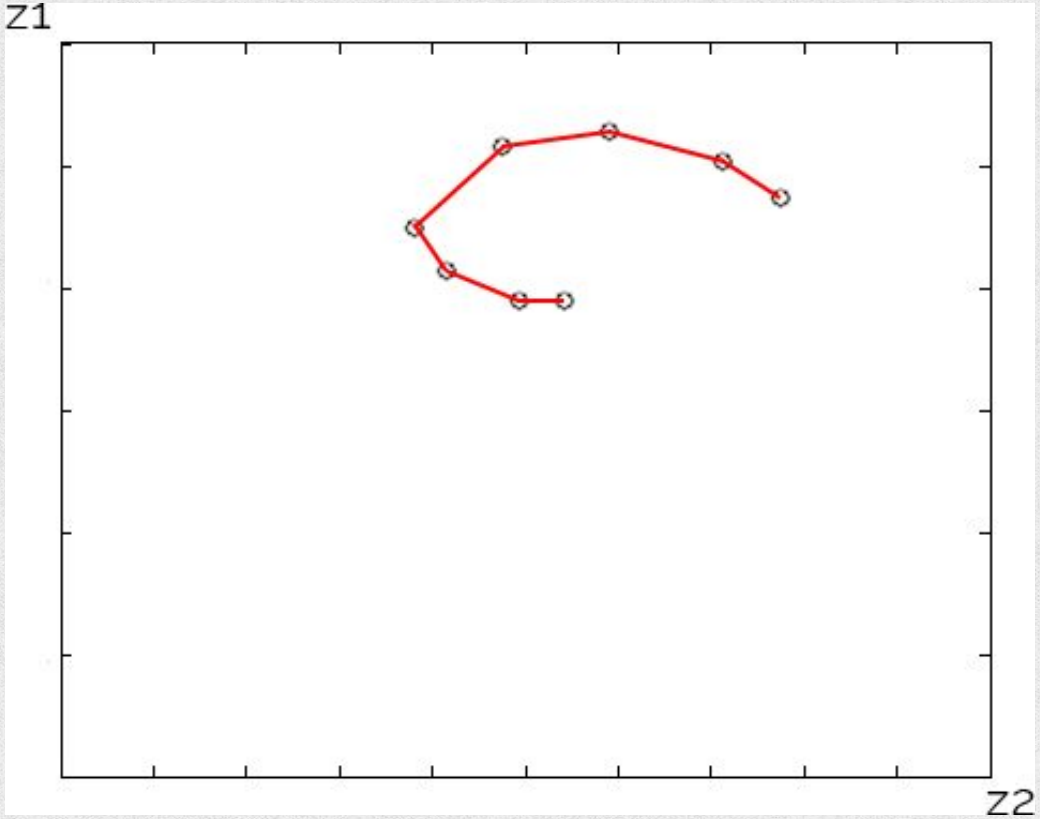
Back-Propagate over Z during … Evaluation



Inspired By: Semantic Image Inpainting with Deep Generative Models – Yeh et al. 2016

4. How did I use GANs to detect novelty/defects ?

Back-Propagate over Z during … Evaluation



** **Back-Propagation Settings:**

⇒ **Steps:** 500
⇒ **Learning Rate:** 1e-4
⇒ **Optimizer:** Momentum
⇒ **Gradient Momentum:** 0.8

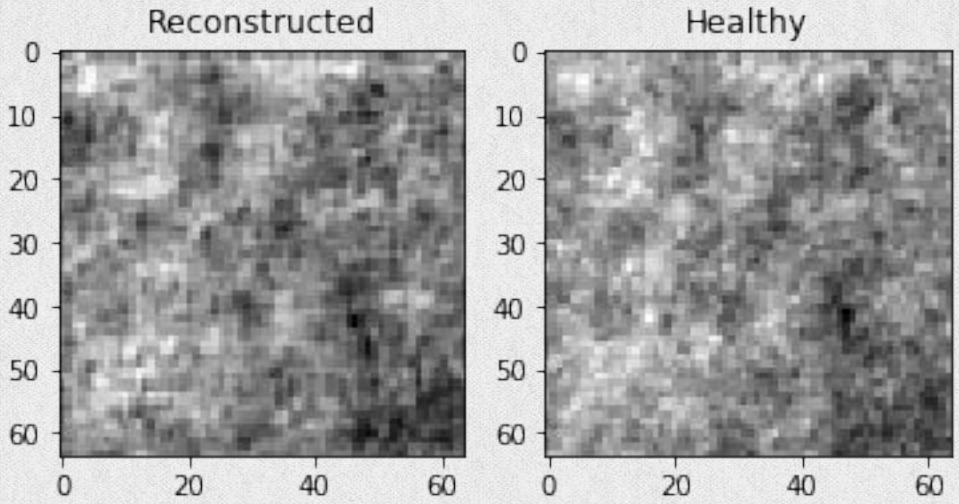
⇒ **Execution Time:** 1.5 sec per sample on GTX Titan X

Inspired By: Semantic Image Inpainting with Deep Generative Models – Yeh et al. 2016

4. How did I use GANs to detect novelty/defects ?

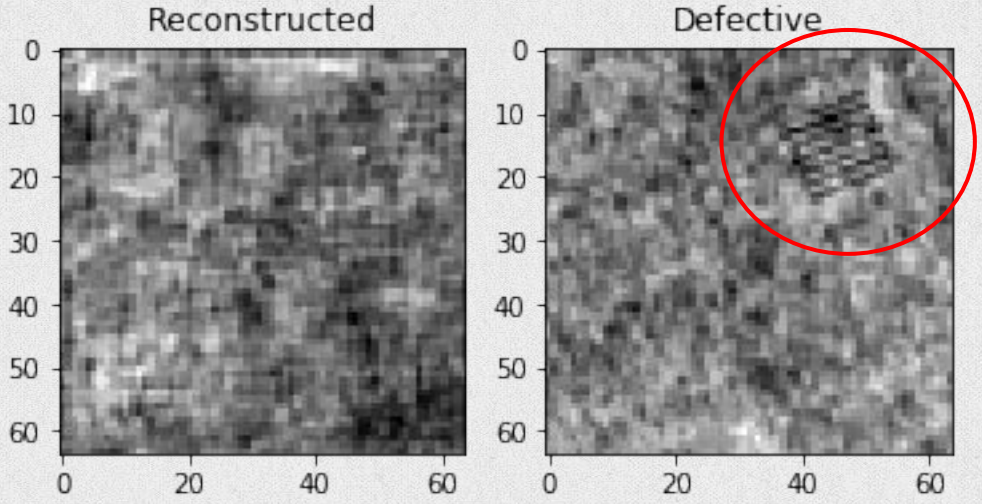
And Johnny … Does your trick work ?

Healthy Case



Loss: 361.672

Defective Case



Loss: 1002.677

2.77x more

5. Tips n Tricks to effectively train a GAN

5. Tips n Tricks to effectively train a GAN

1. Fractionally Strided Convolution vs Resize + Convolution

```
def deconv2Dlayer(self, x, nb_filters):
    x = Conv2DTranspose(
        filters     = nb_filters,
        kernel_size = (5, 5),
        strides     = (2, 2),
        padding     = 'same'
    )(x)

    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization(momentum=0.8)(x)

    return x
```

5. Tips n Tricks to effectively train a GAN

1. Fractionally Strided Convolution vs Resize + Convolution

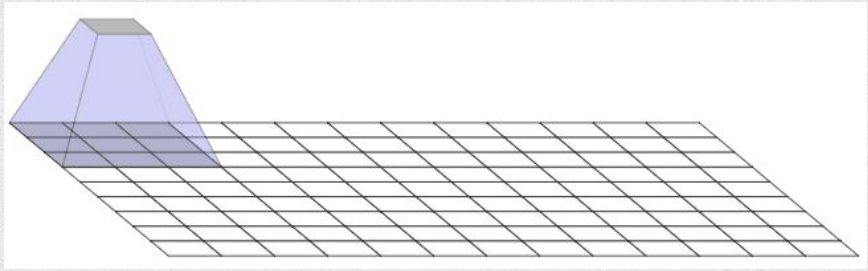
```
def deconv2Dlayer(self, x, nb_filters):
    x = Conv2DTranspose(
        filters     = nb_filters,
        kernel_size = (5, 5),
        strides     = (2, 2),
        padding     = 'same'
    )(x)

    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization(momentum=0.8)(x)

    return x
```

Fractionally Strided Convolution creates artifacts.

** Inherent to the deconvolution operation
** No “setting“ can prevent this



Source: https://distill.pub/2016/deconv-checkerboard/

5. Tips n Tricks to effectively train a GAN

1. Fractionally Strided Convolution vs Resize + Convolution

```
def deconv2Dlayer(self, x, nb_filters):
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(
        filters     = nb_filters,
        kernel_size = (5, 5),
        strides     = (1, 1),
        padding     = 'same'
    )(x)

    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization(momentum=0.8)(x)

    return x
```

5. Tips n Tricks to effectively train a GAN

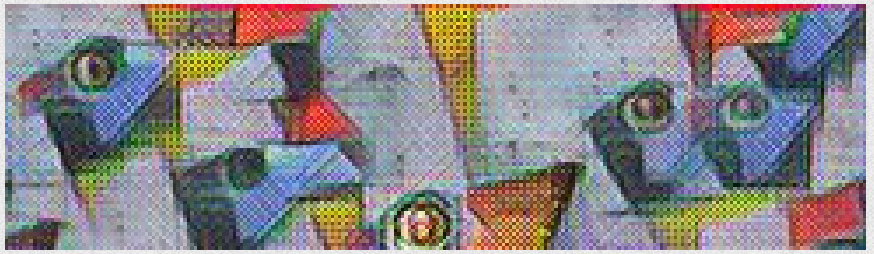
1. Fractionally Strided Convolution vs Resize + Convolution

```
def deconv2Dlayer(self, x, nb_filters):
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(
        filters     = nb_filters,
        kernel_size = (5, 5),
        strides     = (1, 1),
        padding     = 'same'
    )(x)

    x = LeakyReLU(alpha=0.2)(x)
    x = BatchNormalization(momentum=0.8)(x)

    return x
```

With Fractionally Strided Convolutions



With Resize + Convolutions

Source: https://distill.pub/2016/deconv-checkerboard/

5. Tips n Tricks to effectively train a GAN

2. Randomly Sampling Z from a non-uniform latent space

The Generator G creates very similar-looking images for close positions in the latent-space:

⇒ For certain objectives, we want that a small change in Z lead to a large variation

⇒ How To: Train G with Z vectors sampled from a non uniform latent-space

⇒ Sample from a *Truncated Gaussian Distribution* for instance

```
from scipy.stats import truncnorm

def get_truncated_normal(mean=0, std=0.6, low=-1., high=1.):
    # Constrain the sampled data to be in [-1., 1.], centered on 0 with a std of 0.6 (hyperparameter)
    return truncnorm((low - mean) / std, (high - mean) / std, loc=mean, scale=std)

latent_generator = get_truncated_normal()

z = latent_generator.rvs( size=(64, 100) )  # generate a batch of 64 z vectors (64, 100)
```

5. Tips n Tricks to effectively train a GAN

3. Avoid to kill your gradients (sparse) => ReLU & MaxPool

GAN training leads to an instable equilibrium, avoid everything that makes it less stable:

⇒ **ReLU** kills the gradients:

⇒ **MaxPooling**: the gradient from the next layer is passed back to only the neuron which achieved the max. All other neurons get zero gradient.

Better using LeakyReLU => for G and D.

⇒ **Breaking news:** Now implemented in TensorFlow 1.4.0-rc0: **tf.nn.leaky_relu**

5. Tips n Tricks to effectively train a GAN

4. Try everything you can do to make the discriminator's job harder

Forging some data is far harder than discriminating legit from fake data => D trains faster.

⇒ Train G twice each time you train D.

⇒ Train D only if G accuracy is higher than a threshold (0.3 ? 0.25 ? => HyperParameter)

⇒ Use a less efficient optimizer for D (SGD) than for G (Adam).

⇒ Noisy Labels for D: Flip labels from time to time for D: Fake is Legit and Legit is fake

⇒ Soft Labels for D: Do not use "{0 ,1}" labels but "{0.1, 0.9}"

⇒ Add noise for D and decay it over time (Dropout, GaussianNoise, etc.)

THANK YOU

Jonathan DEKHTIAR - @Born2Data
contact@jonathandekhtiar.eu

"Let's see if I can back-propagate from the questions to the answers."