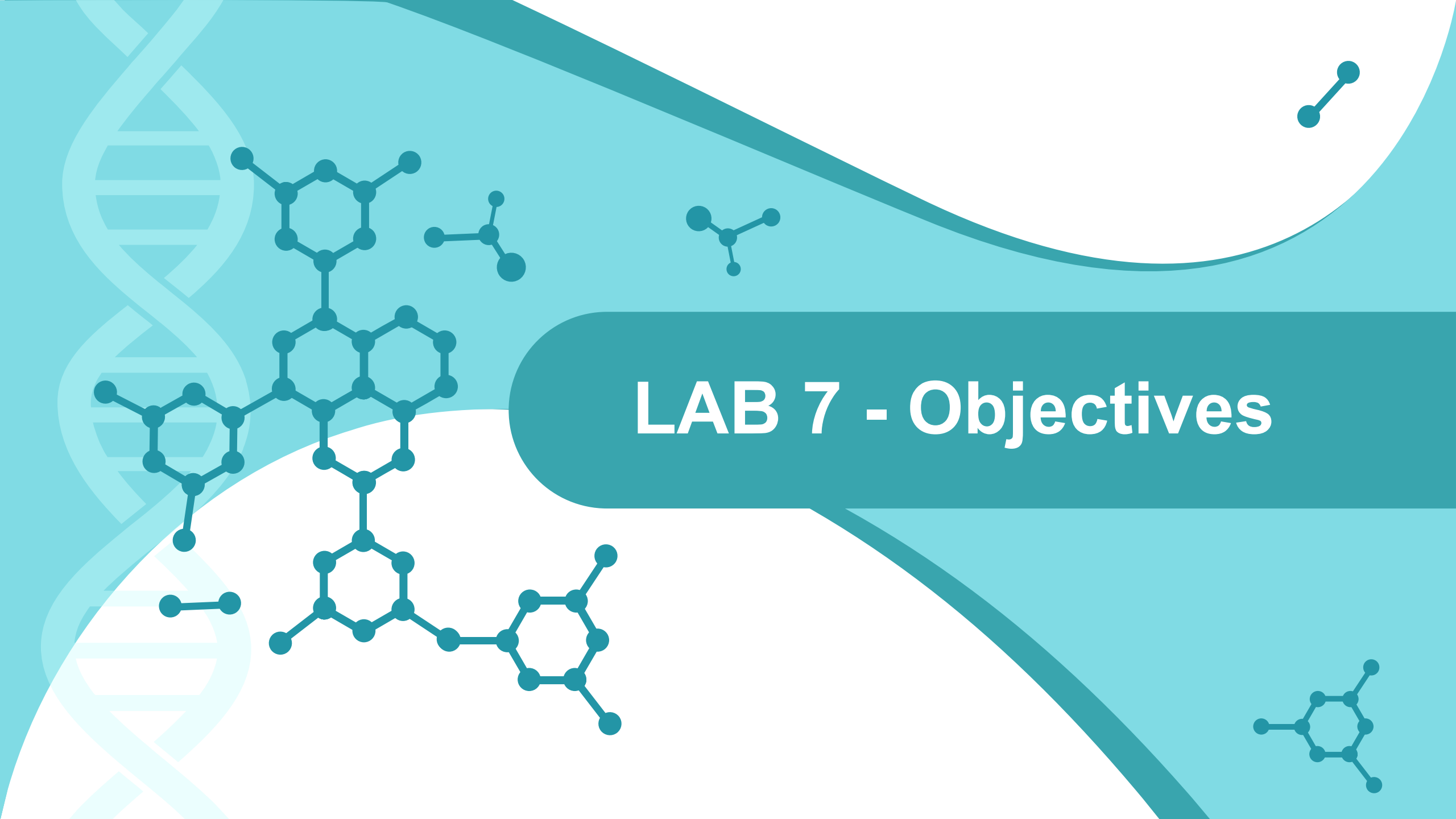
Bioinformatics LAB 7

Generative Adversarial Networks

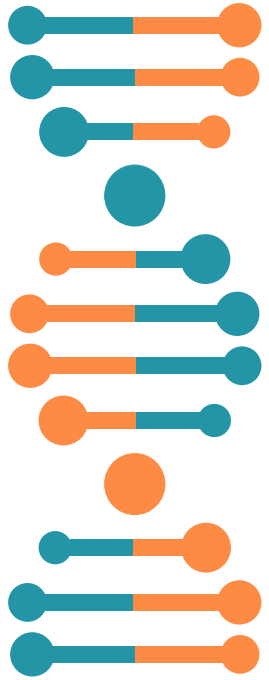
Prof.ssa Elisa Ficarra
Prof.ssa Santa Di Cataldo
Eng. Marta Lovino
Eng. Alessio Mascolini

Politecnico di Torino
DAUIN
Dept. of Control and Computer Engineering



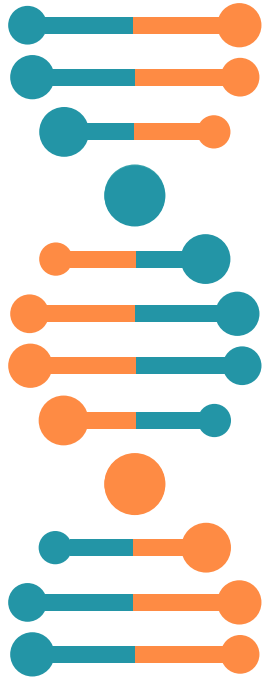
LAB 7 - Objectives

Objectives



- Similarity between images
- Adversarial Training
- Generative Adversarial Networks
- Upsampling in decoder layers
- Evaluating GANs
- Adversarial Training in Tensorflow 2.3

Similarity between images



Despite it being easy for humans, calculating the similarity between images is not an easy task for algorithms

MSE is strongly affected by factors that should not be influential e.g. same image shifted 1px in any direction will have higher loss than a full gray image

Distance in latent space is a good technique, but it requires a suitable latent space and a paired dataset for training

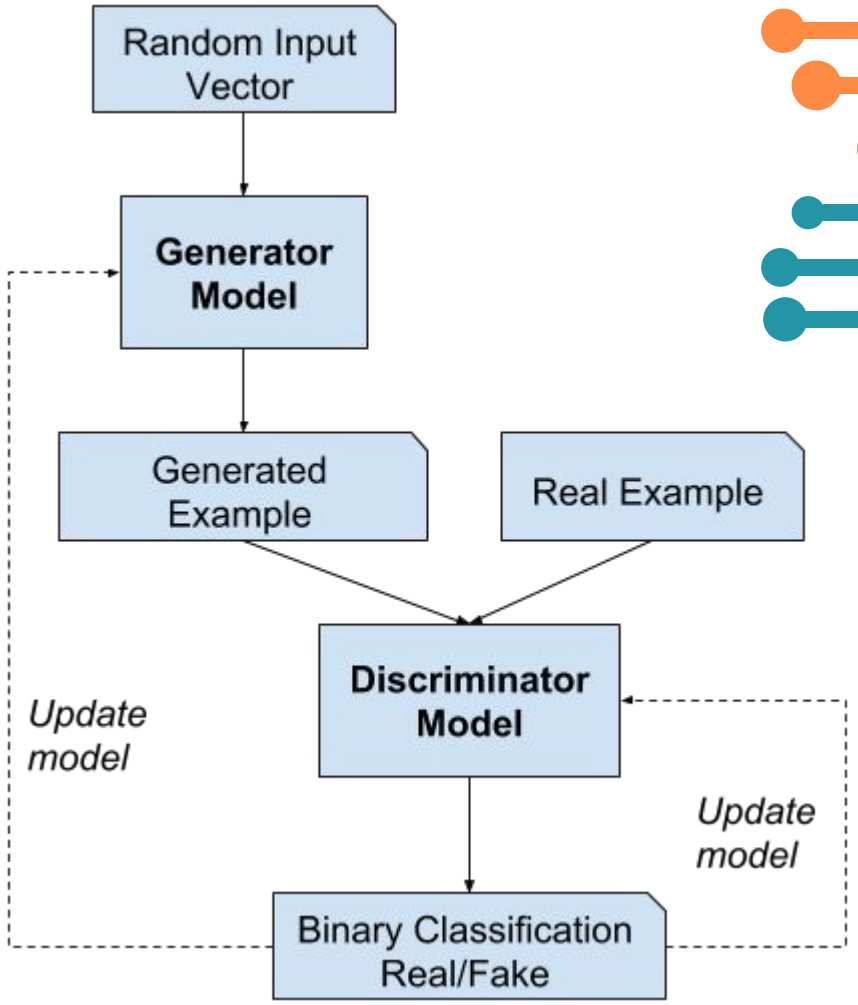
Adversarial Training

How can we know how similar the images made by our generator are to the original data?

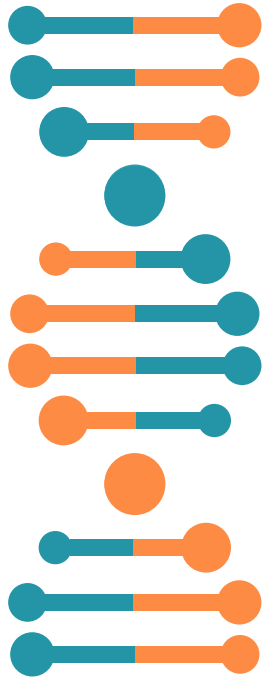
Generative Adversarial Networks infer this metric using an heuristic, which is how difficult a second neural network is to deceive.

This approach works without paired data, which makes acquiring datasets significantly easier.

GANs represent the state of the art in image generation and image translation.



Generative Adversarial Networks



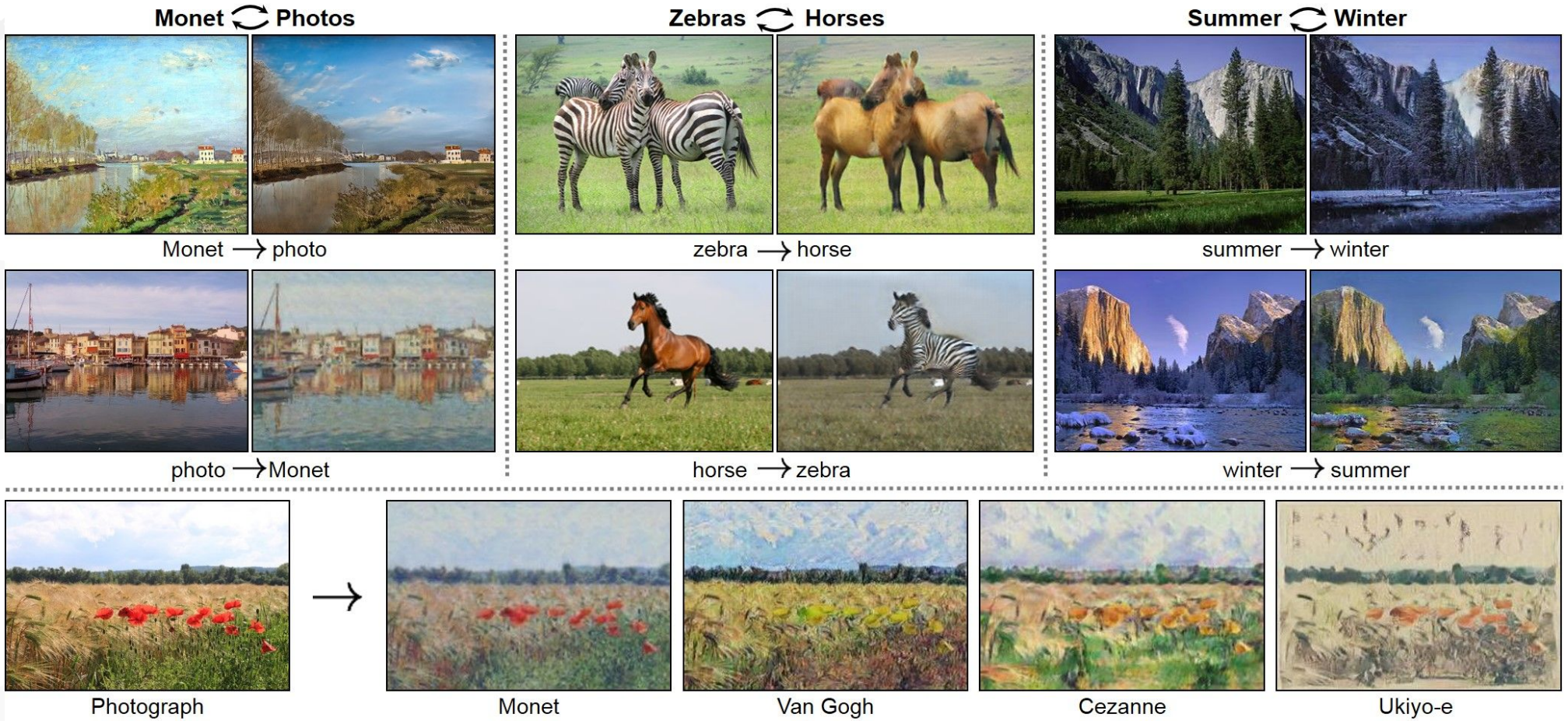
Strengths:

- The discriminator can tell the generator how it managed to discover that the image is fake, allowing it to avoid making the same mistake again
- Can learn to translate images (e.g. Apples to oranges, horses to zebras) from unpaired data
- Extremely high quality results, often indistinguishable to the human eye in the best models

Weaknesses:

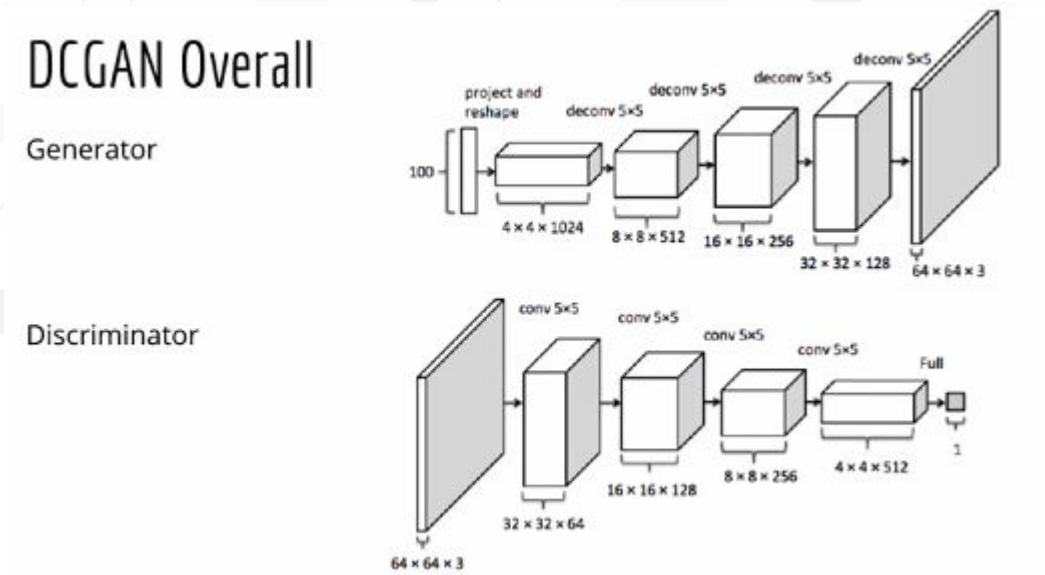
- Unstable training due to the loss function constantly shifting
- Requires lots of resources and computation time
- Mode collapse

Generative Adversarial Networks



DCGAN

Simplest type of GAN



Upsampling in decoder layers

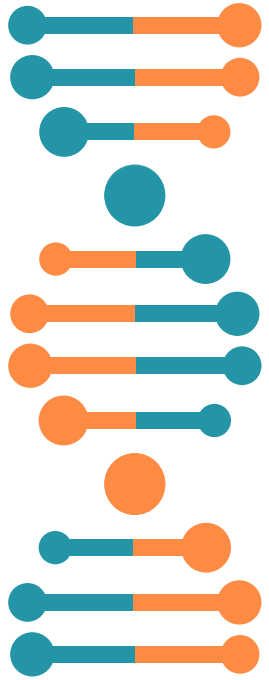
ML approach: Transposed Convolution
Requires tuning to avoid artifacts

Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

All layers use resize-convolution.
No artifacts.

Newer architectures tend to use traditional upsampling (Nearest neighbour, bilinear)

Evaluating GANs



Discriminator and Generator losses fluctuate depending on who's having the upper hand.
This is normal, provided that one network does not overpower the other.

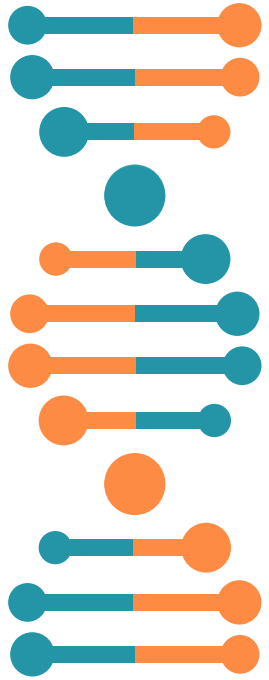
This loss cannot be used to measure the quality of the result

It's not trivial to obtain quantitative data on how good a GAN performs
Most common metric: Frechet Inception Distance

FID is the Frechet distance between the feature vectors of the real and generated images using a pretrained Inception v3 classifier

FID cannot be used as a loss, for the same reasons accuracy can't be used as a loss for a classifier

How to make one in tensorflow



tf.keras.Model needs __init__ and compile
We can overload train_step to define custom training behaviour

tensorflow gives us the GradientTape to do automatic differentiation

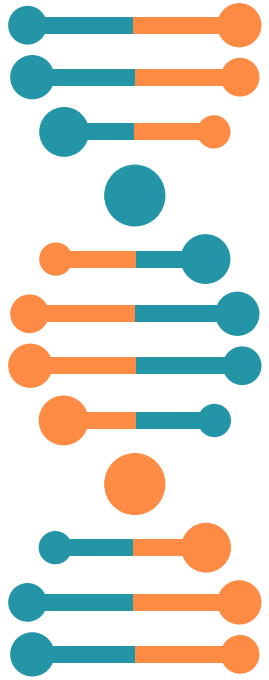
Can calculate gradients for any tf.Variable, provided operations are differentiable

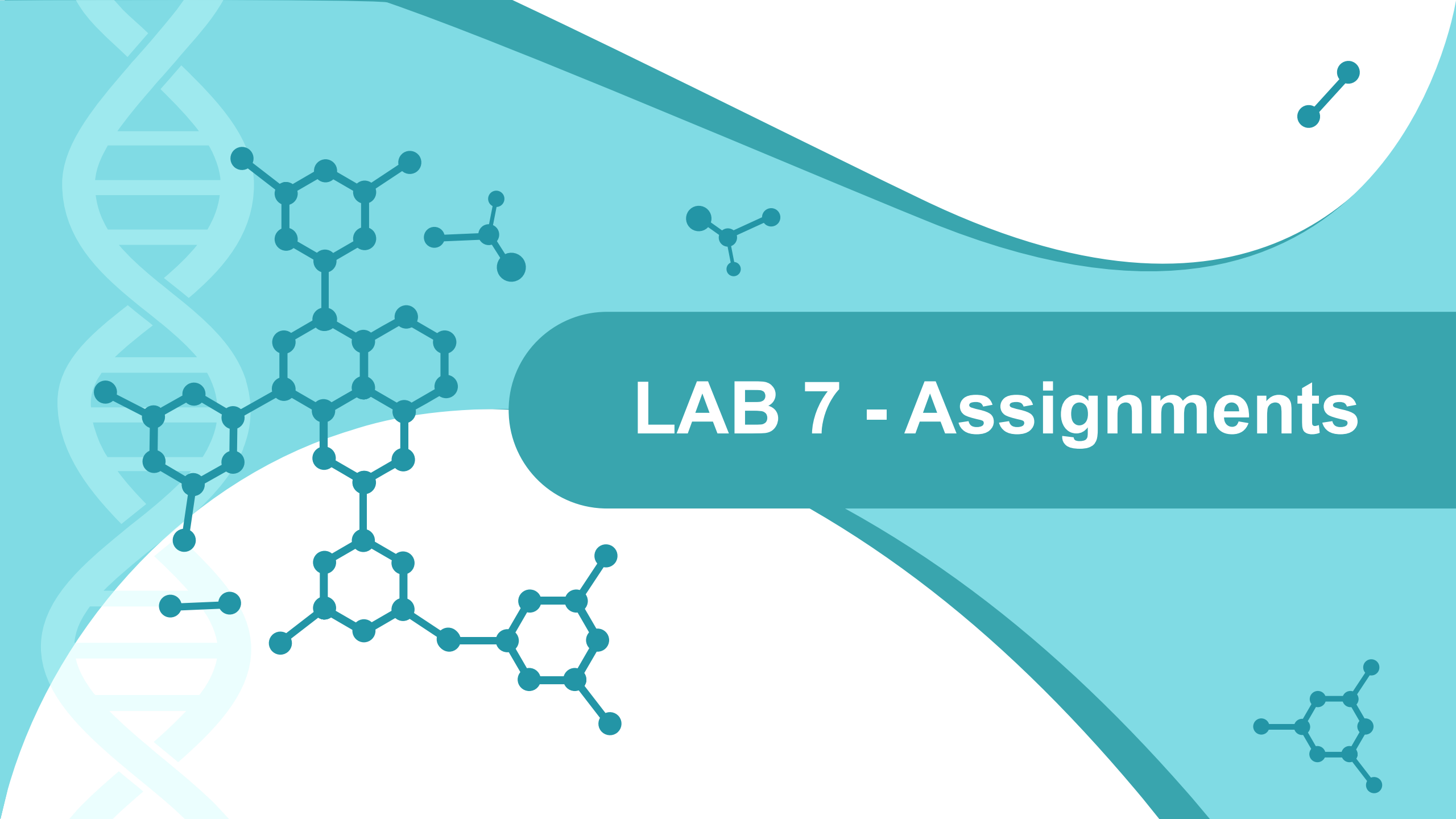
```
with tf.GradientTape() as tape:
    y_pred = self(x)
    loss = self.loss(y, y_pred)

gradients = tape.gradient(loss, self.trainable_variables)
self.optimizer.apply_gradients(zip(gradients, self.trainable_variables))
```

How to make one in tensorflow

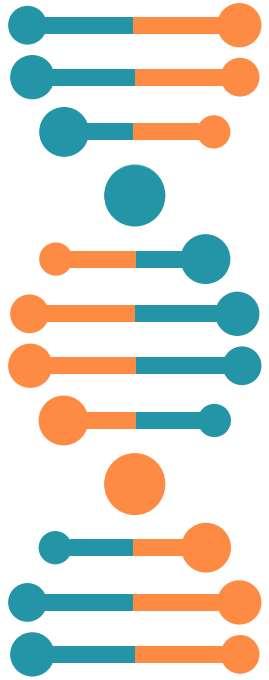
```
class MyModel(tf.keras.Model):
    def __init__(self,...):
        super(MyModel,self).__init__()
        #Constructor code
    def compile(self,...):
        super(MyModel,self).compile()
        #Code to run on compile
    def train_step(self,data):
        x,y = data
        #Code for training
        return {‘Metric name’: metricVariable}
```





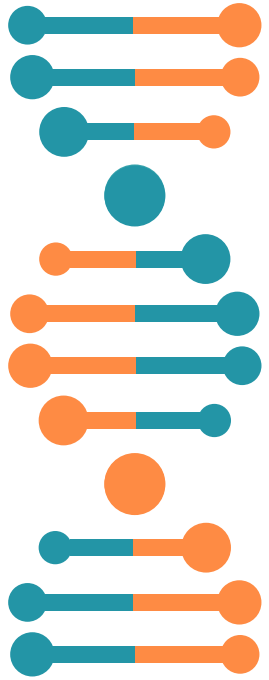
LAB 7 - Assignments

Assignment 1: Linear regression using keras custom training and automatic differentiation



- Generate a dataset consisting of 10 points along y=x+2 + some gaussian noise
- Create a tf.keras.Model using subclassing, overload the train_step function to implement linear regression using tensorflow automatic differentiation
- Predict m and q
- Minimise the mse between your model predictions and the data

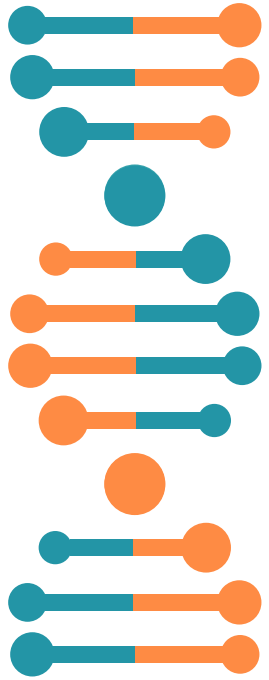
Assignment 2: Fashion MNIST hallucination using DCGAN



- Download Fashion MNIST using tf.keras.datasets
- Create a generator model using Convolutional and Upsample2D layers
- Create a discriminator model using a simple convnet
- Create a GAN model by subclassing, taking the generator and discriminator as inputs in the constructor
- Overload the GAN model train_step to implement adversarial training
- Visualize the generated fashion MNIST images

LAB7 – Take home message

- MSE is appropriate for the latent space, not the image space
- A neural network can be a loss function
- GANs are not magic, they require a lot of computational power and are unstable, especially at higher resolutions
- Discriminator and Generator loss are only meaningful to evaluate the training process, not as quality metrics



Questions?

Remember:
no question is
stupid