Current and future trends

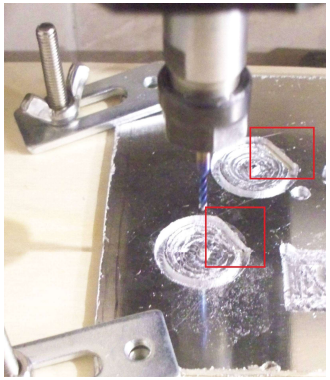
Computer Vision and Deep Learning

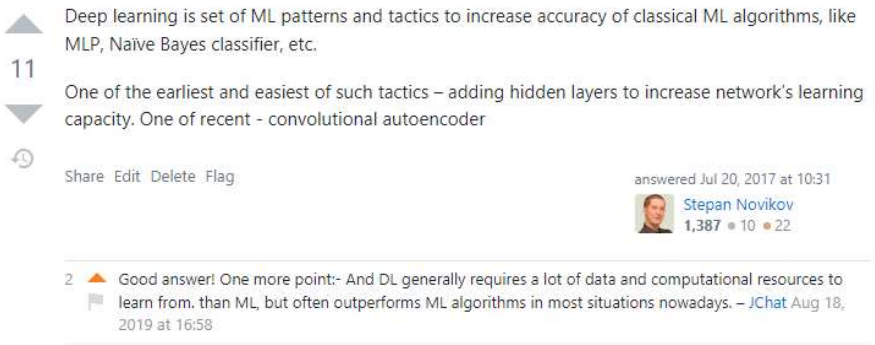
SELF INTRO

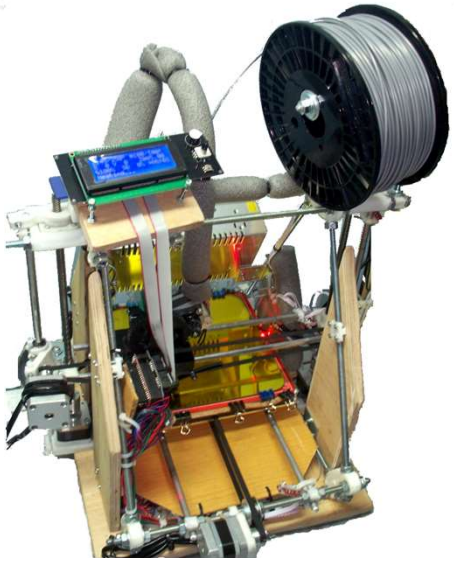
Professional experience - **15+** years of software development experience, **7+** years experience in enterprise level grade technical solutions, including cloud computing, machine learning and AI, data science and IoT

Hobby - Smart AI home systems, CNC and 3D printers

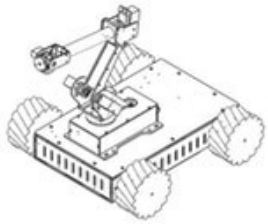
Communities - Stack Overflow, Kaggle ML competitions and many others

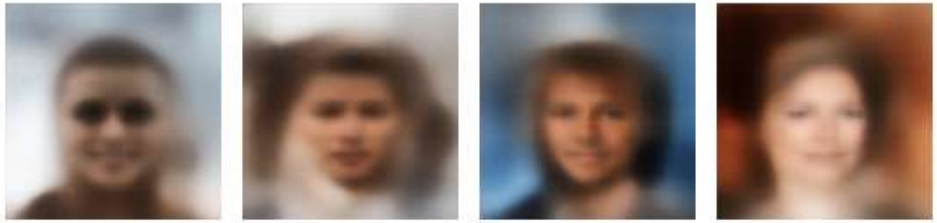


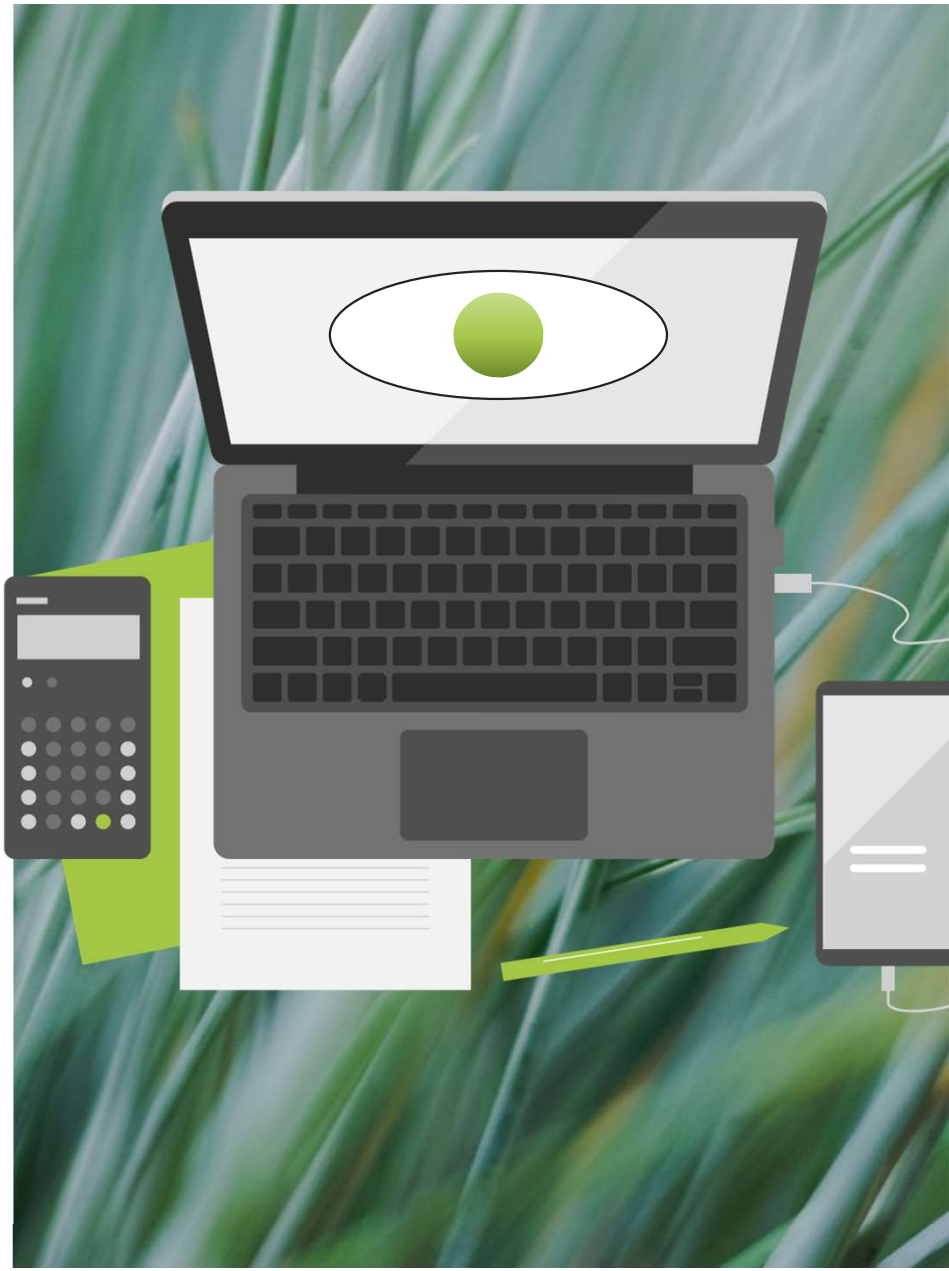


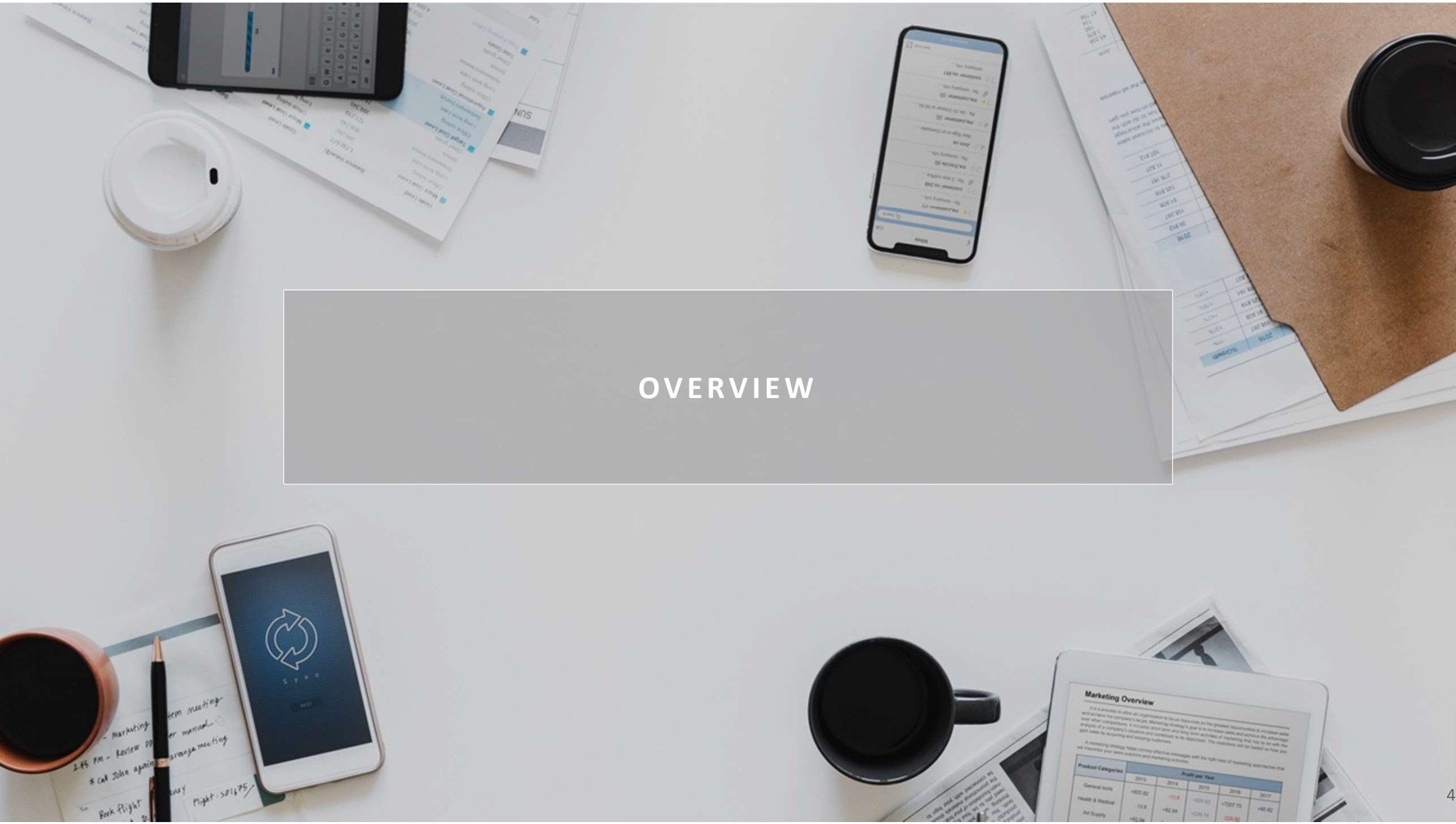


Contents

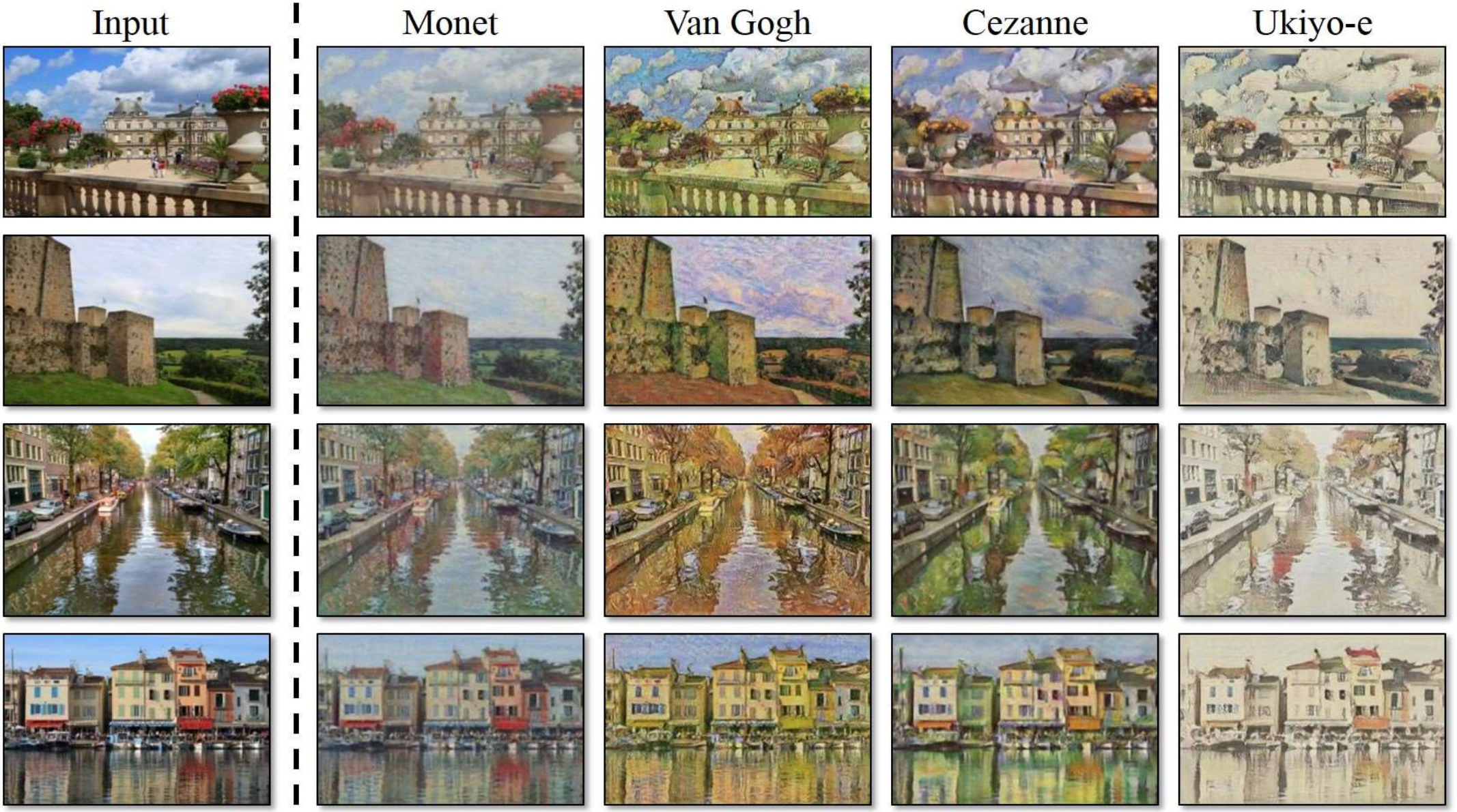






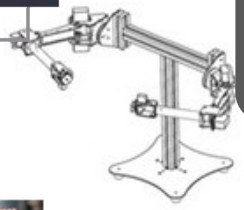


OVERVIEW

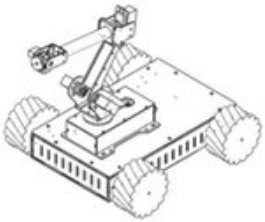


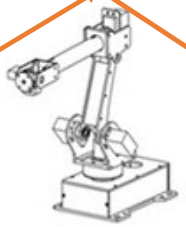
https://junyanz.github.io/CycleGAN/

ML DEFINITION



2. Choose/amend ML model

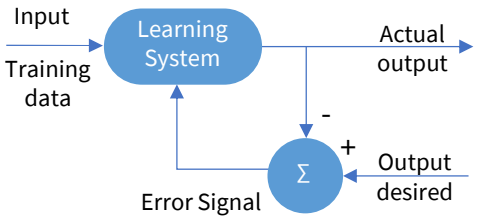




1. Get, Clean, Prepare & Normalize Data

image_id	5_o_Clock_Shadow	Arched_Eyebrows	Attractive	Bags_Under_Eyes	Bald	Bangs	Big_Lips	Big_Nose
000001.jpg	-1	1	1	-1	-1	-1	-1	-1
000002.jpg	-1	-1	-1	1	-1	-1	-1	1
000003.jpg	-1	-1	-1	-1	-1	-1	1	-1
000004.jpg	-1	-1	1	-1	-1	-1	-1	-1
000005.jpg	-1	1	1	-1	-1	-1	1	-1
000006.jpg	-1	1	1	-1	-1	-1	1	1
000007.jpg	1	-1	1	1	-1	-1	1	-1
000008.jpg	1	1	-1	1	-1	-1	1	-1
000009.jpg	-1	1	1	-1	-1	1	1	-1
000010.jpg	-1	-1	1	-1	-1	-1	-1	-1

3. Train Model



4. Test Outcomes

5. Improve

Convolutional Kernel

DL DEFINITION

2. Choose/amend ML model

1. More layers

2. Convolutional Networks

3. Autoencoders

SoftMax

1. Get, Clean, Prepare & Normalize Data

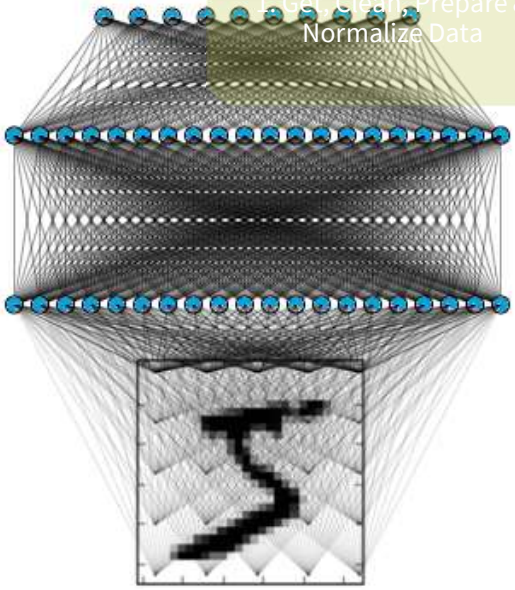
3. Train Model

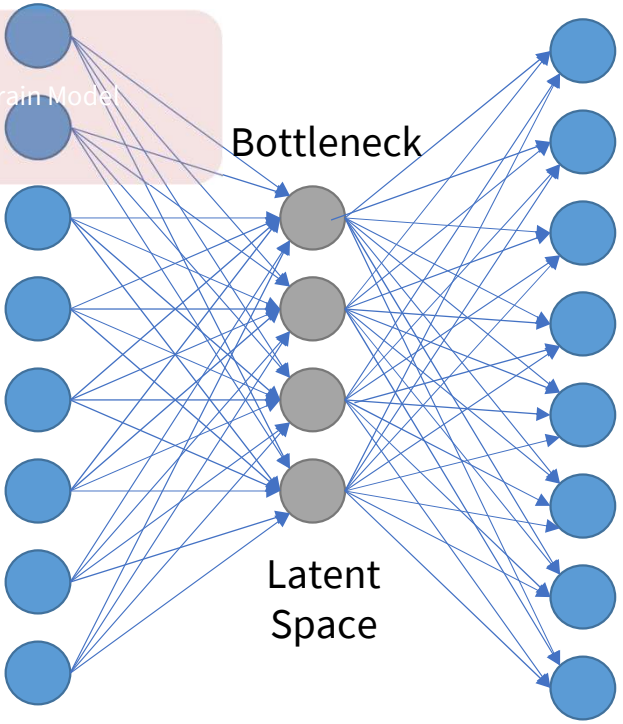
4. Generative adversarial networks

5. Ensembles

5. Improve

4. Test Outcomes

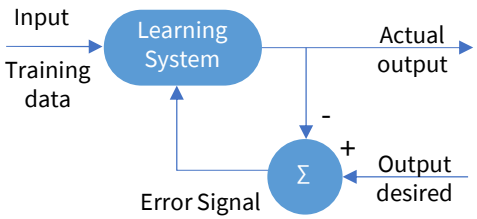




ML TYPES

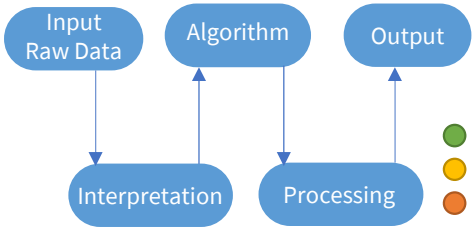
SUPERVISED

Based on training
X,Y -> predict Y* for input I*



UNSUPERVISED

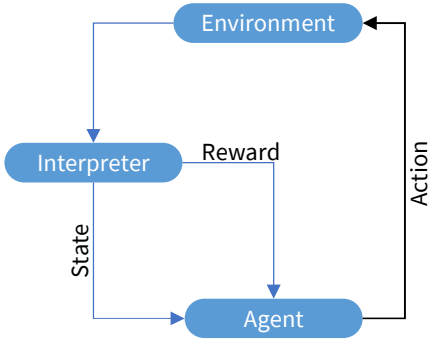
Analyses of unlabeled data



- Unknown Output
- No training Data Set

REINFORCEMENT

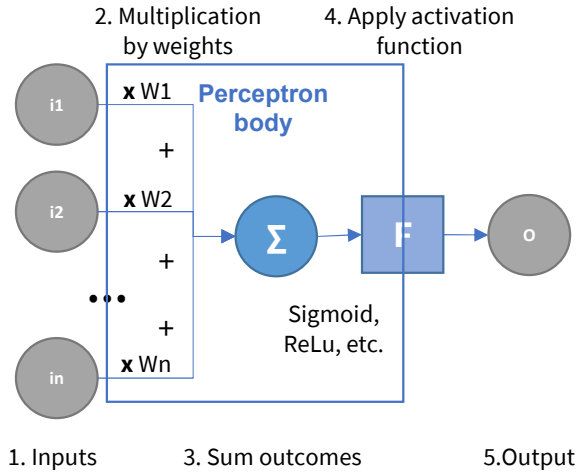
Mixed approach
Try-Reward model



HISTORICAL BACKGROUND FOR COMPUTER VISION AI

EVOLUTION

Perceptron



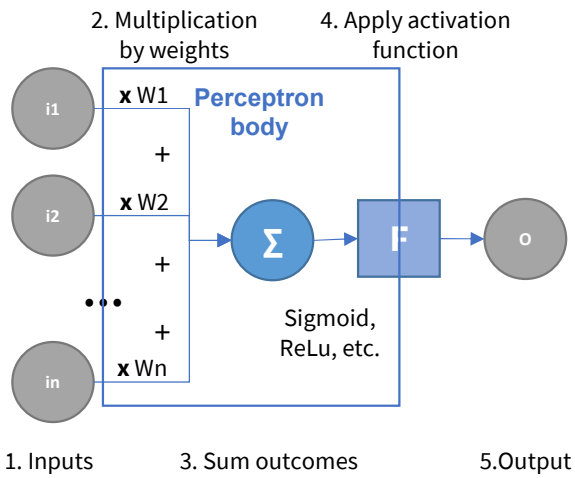
$$o = F(\sum_{k=1}^{n} i_k * w_k)$$

Less than 93% accuracy

EVOLUTION

MNIST dataset (illustration from Wiki)

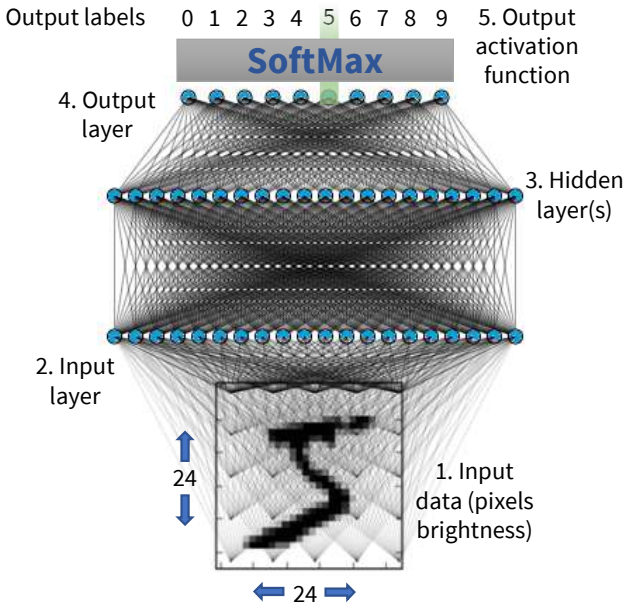
Perceptron



$$o = F(\sum_{k=1}^{n} i_k * w_k)$$

Less than 93% accuracy

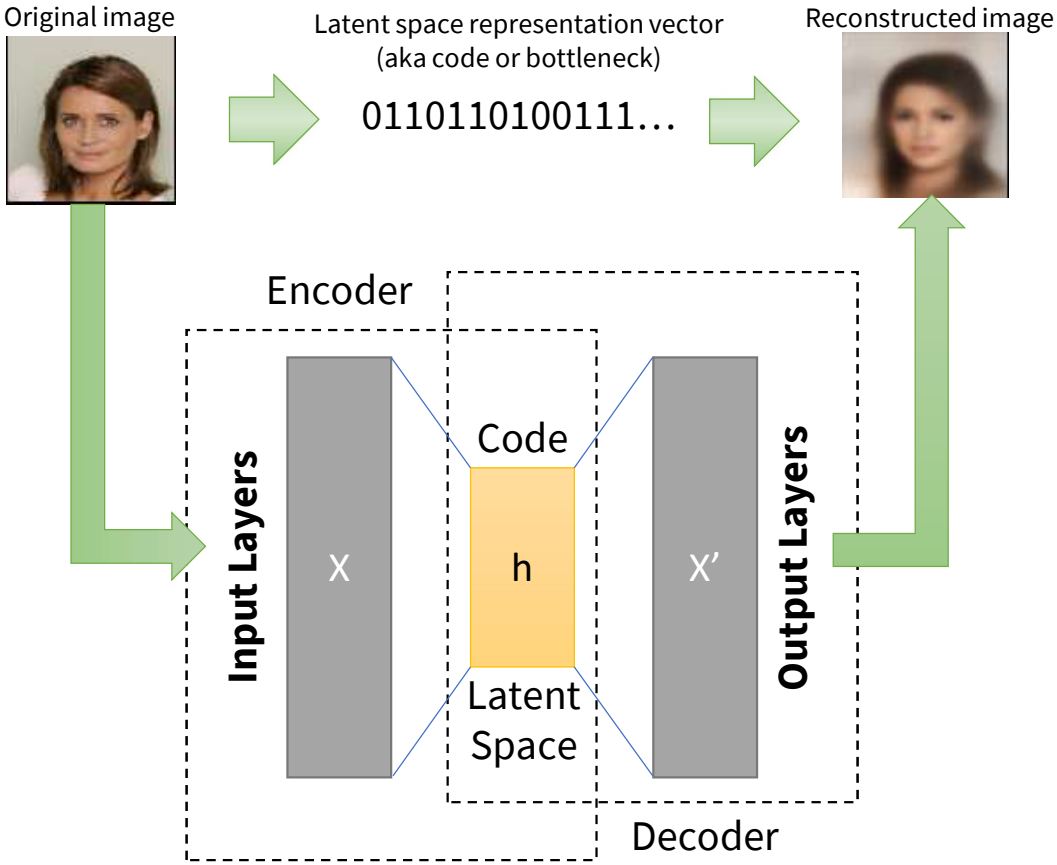
Multilayer perceptron (MLP or FCN or Dense layers)

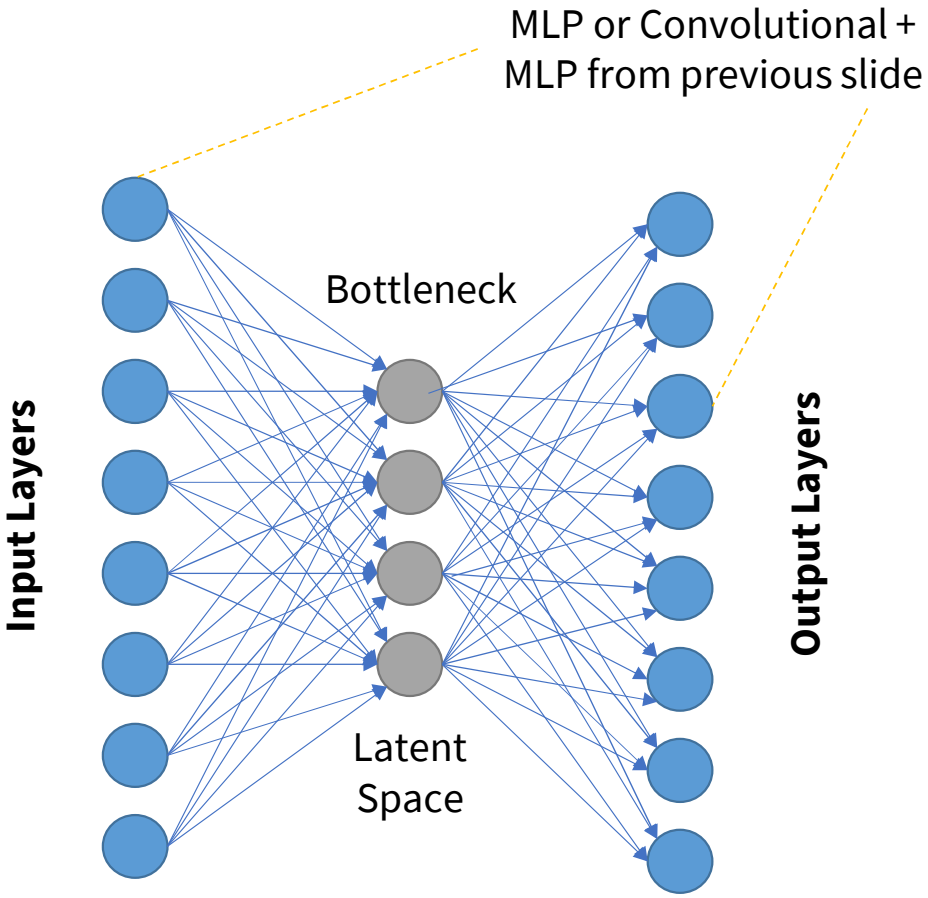


97%+

RECENT

Autoencoder

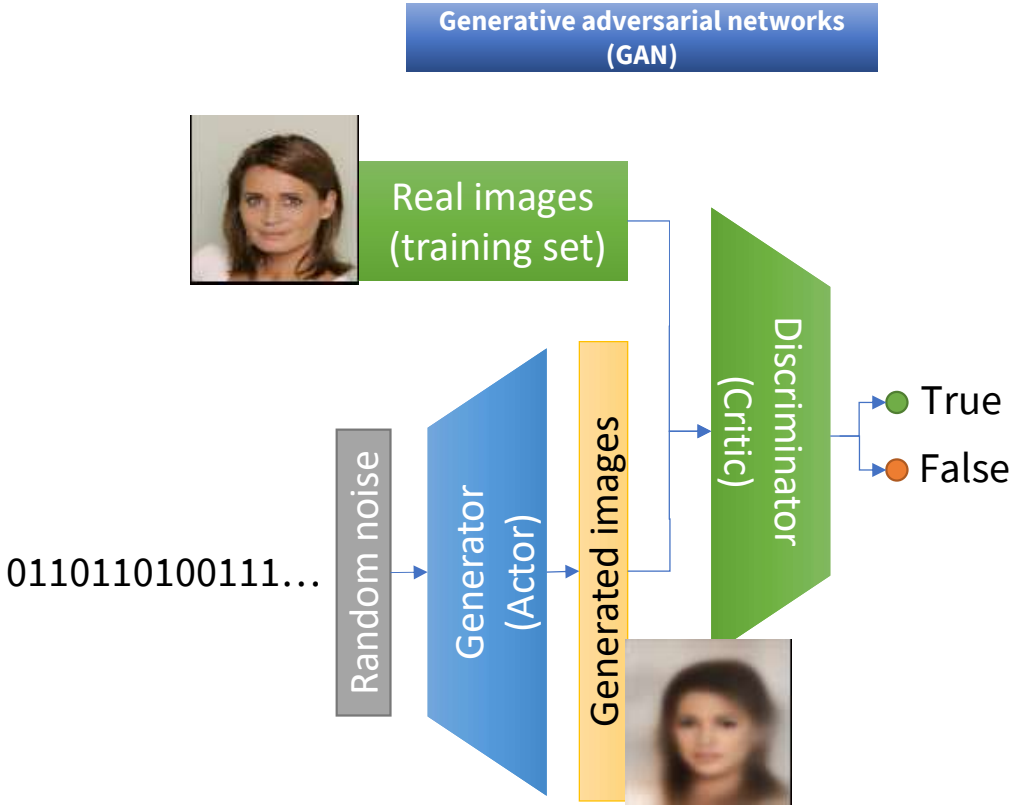


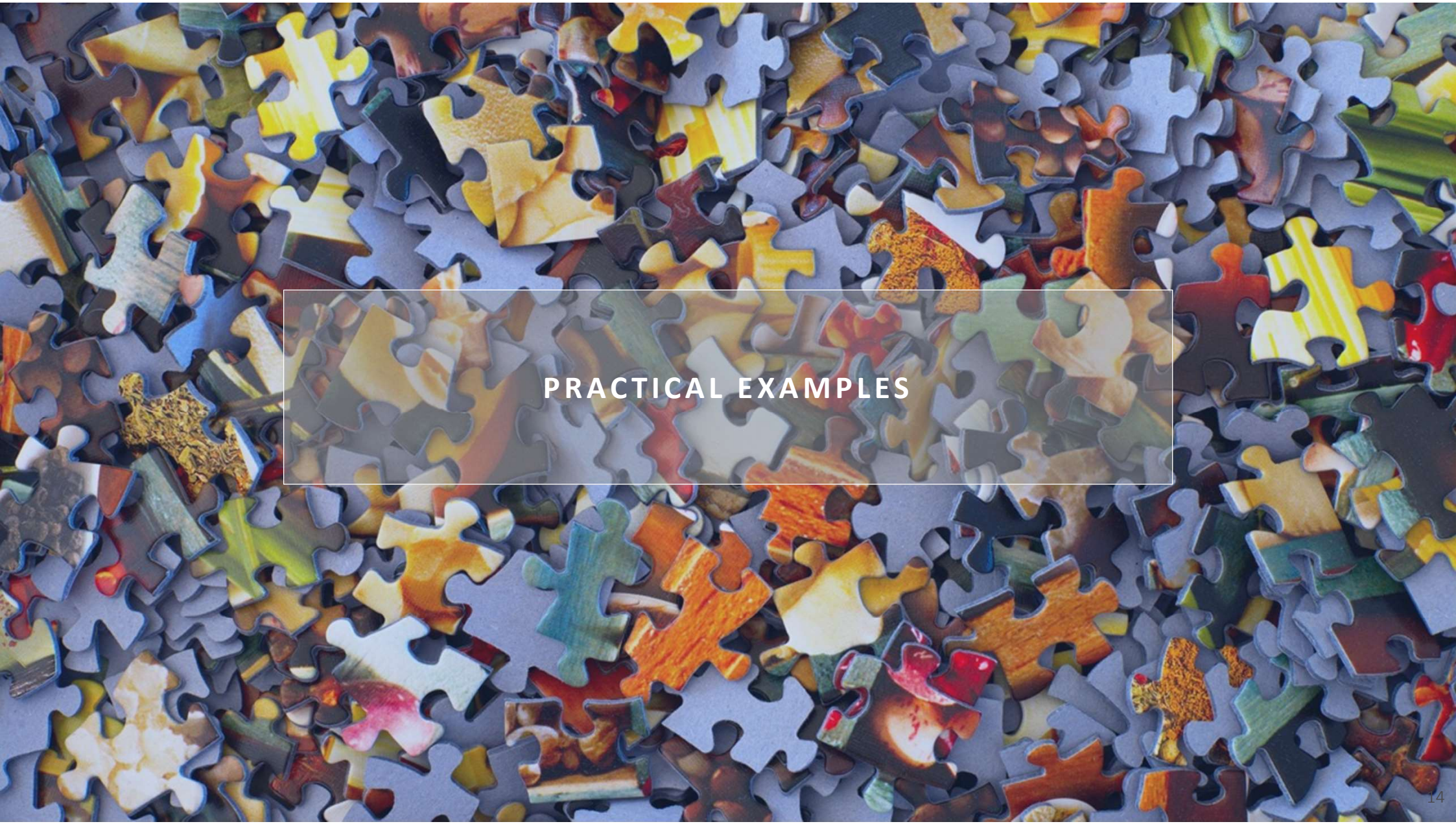


RECENT

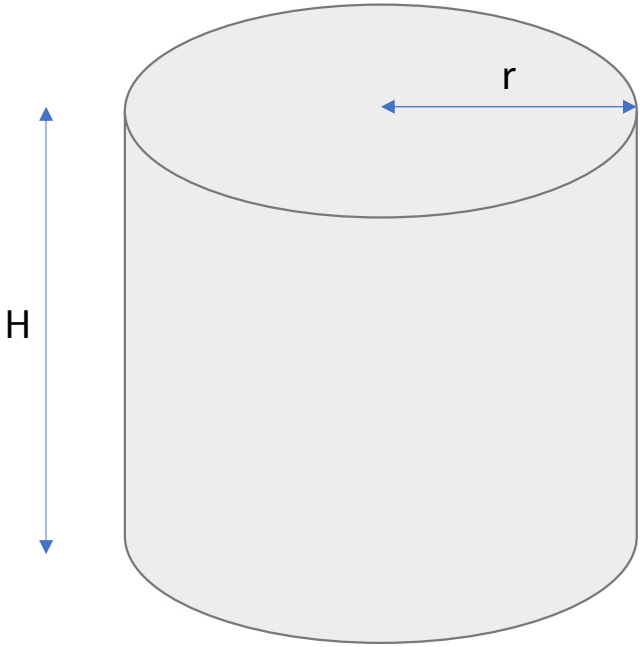
GAN training sequence:

1. Train critic (Actor's weights frozen)
2. Train Actor (Critic's weights are frozen)
3. Check progress and go to 1 if we have it




PRACTICAL EXAMPLES

DEFINITION



This shape is basically defined by the vector of 2 values – H and r

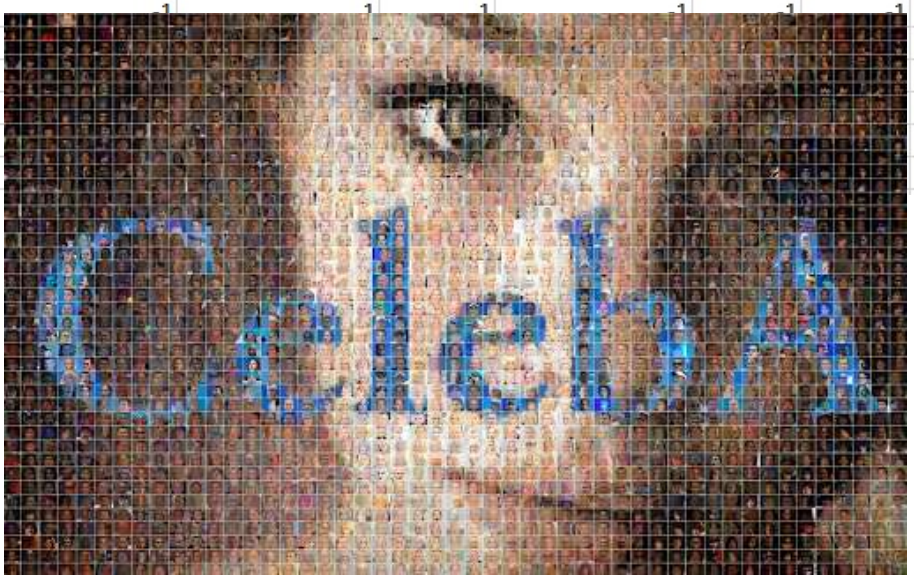
Can we represent something a bit more complex, like face with some kind of vector?

DATASET

For our practical lab we will use CelebA dataset.

This dataset contains more than 200K images with faces of celebrities, plus, tagged with attributes such as ‘smiling', ‘wearing glasses‘', etc.

image_id	5_o_Clock_Shadow	Arched_Eyebrows	Attractive	Bags_Under_Eyes	Bald	Bangs	Big_Lips	Big_Nose
000001.jpg	-1	1	1	-1	-1	-1	-1	-1
000002.jpg	-1	-1	-1	1	-1	-1	-1	1
000003.jpg	-1	-1	-1	-1	-1	-1	1	-1
000004.jpg	-1	-1	1	-1	-1	-1	-1	-1
000005.jpg	-1	1	1	-1	-1	-1	1	-1
000006.jpg							1	-1
000007.jpg							1	1
000008.jpg							1	-1
000009.jpg							1	-1
000010.jpg							-1	-1



TRAINING

In a couple of days after start

Epoch 00099: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 254s 965ms/step - batch: 131.5000 - size: 767.4205 - loss: 222.8837 - vae_r_loss: 163.3671 - vae_kl_loss: 59.5159 - lr: 1.0000e-06
Epoch 100/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 222.9158 - vae_r_loss: 163.4007 - vae_kl_loss: 59.5169
Epoch 00100: saving model to run/vae/0001_faces\weights\weights-100-222.92.h5

Epoch 00100: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 254s 965ms/step - batch: 131.5000 - size: 767.4205 - loss: 222.9158 - vae_r_loss: 163.4007 - vae_kl_loss: 59.5169 - lr: 1.0000e-06
Iteration 14 started...
Epoch 1/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 222.9469 - vae_r_loss: 163.4001 - vae_kl_loss: 59.5476
Epoch 00001: saving model to run/vae/0001_faces\weights\weights-001-222.95.h5

Epoch 00001: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 255s 962ms/step - batch: 131.5000 - size: 767.4205 - loss: 222.9469 - vae_r_loss: 163.4001 - vae_kl_loss: 59.5476 - lr: 1.0000e-06
Epoch 2/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 222.9248 - vae_r_loss: 163.3783 - vae_kl_loss: 59.5461
Epoch 00002: saving model to run/vae/0001_faces\weights\weights-002-222.92.h5

Epoch 00002: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 255s 965ms/step - batch: 131.5000 - size: 767.4205 - loss: 222.9248 - vae_r_loss: 163.3783 - vae_kl_loss: 59.5461 - lr: 1.0000e-06
Epoch 3/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: **222.9249** - vae_r_loss**: 163.3844** - vae_kl_loss: **59.5423**
Epoch 00003: saving model to run/vae/0001_faces\weights\weights-003-222.93.h5

TRAINING

A couple of weeks more…

Epoch 00001: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 253s 956ms/step - batch: 131.5000 - size: 767.4205 - loss: 214.9664 - vae_r_loss: 156.0279 - vae_kl_loss: 58.9384 - lr: 1.0000e-06
Epoch 2/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 214.9495 - vae_r_loss: 156.0112 - vae_kl_loss: 58.9376
Epoch 00002: saving model to run/vae/0001_faces\weights\weights-002-214.95.h5

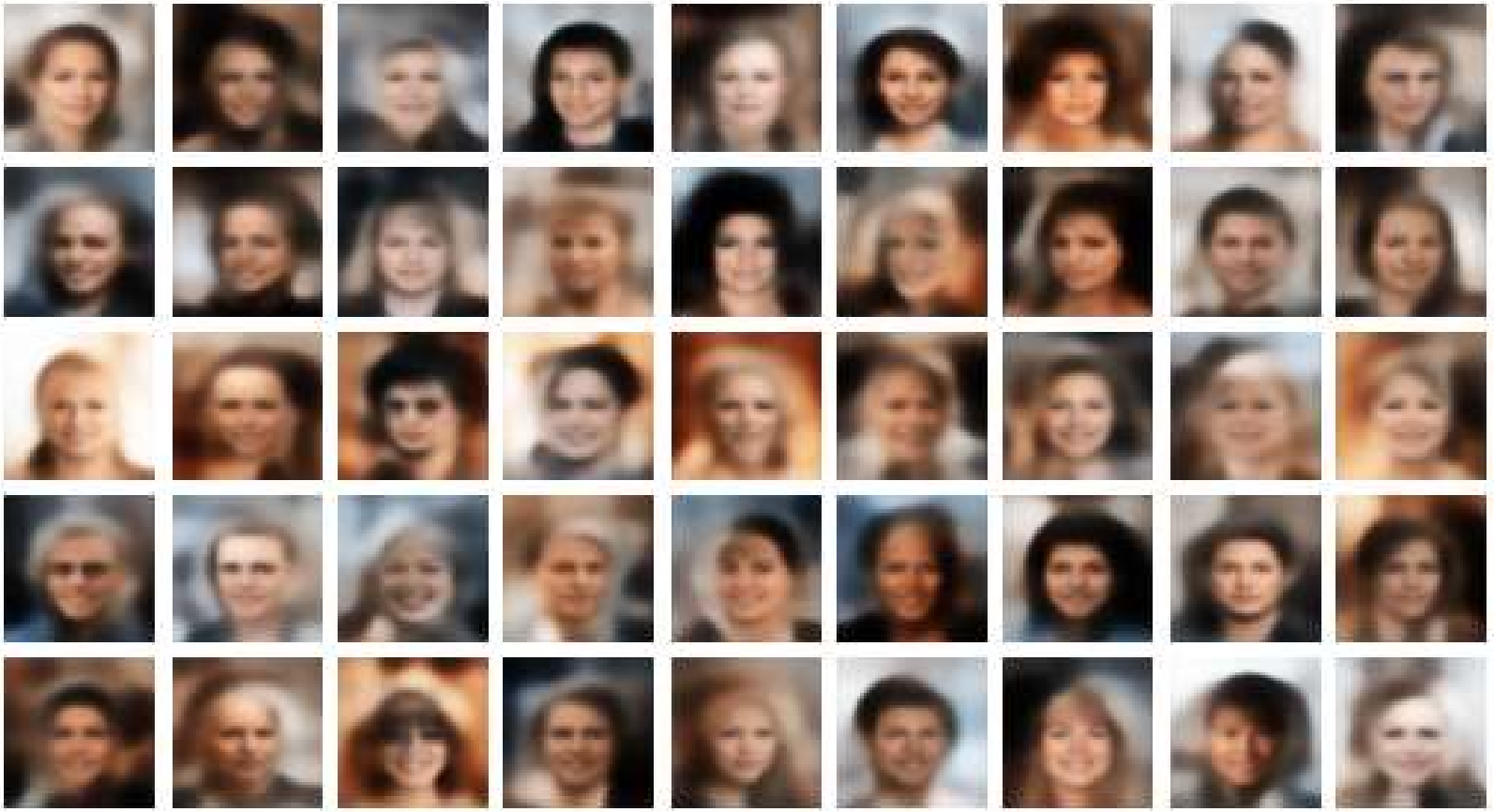
Epoch 00002: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 252s 956ms/step - batch: 131.5000 - size: 767.4205 - loss: 214.9495 - vae_r_loss: 156.0112 - vae_kl_loss: 58.9376 - lr: 1.0000e-06
Epoch 3/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 214.9720 - vae_r_loss: 156.0549 - vae_kl_loss: 58.9175
Epoch 00003: saving model to run/vae/0001_faces\weights\weights-003-214.97.h5

Epoch 00003: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 252s 957ms/step - batch: 131.5000 - size: 767.4205 - loss: 214.9720 - vae_r_loss: 156.0549 - vae_kl_loss: 58.9175 - lr: 1.0000e-06
Epoch 4/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 214.9620 - vae_r_loss: 156.0432 - vae_kl_loss: 58.9192
Epoch 00004: saving model to run/vae/0001_faces\weights\weights-004-214.96.h5

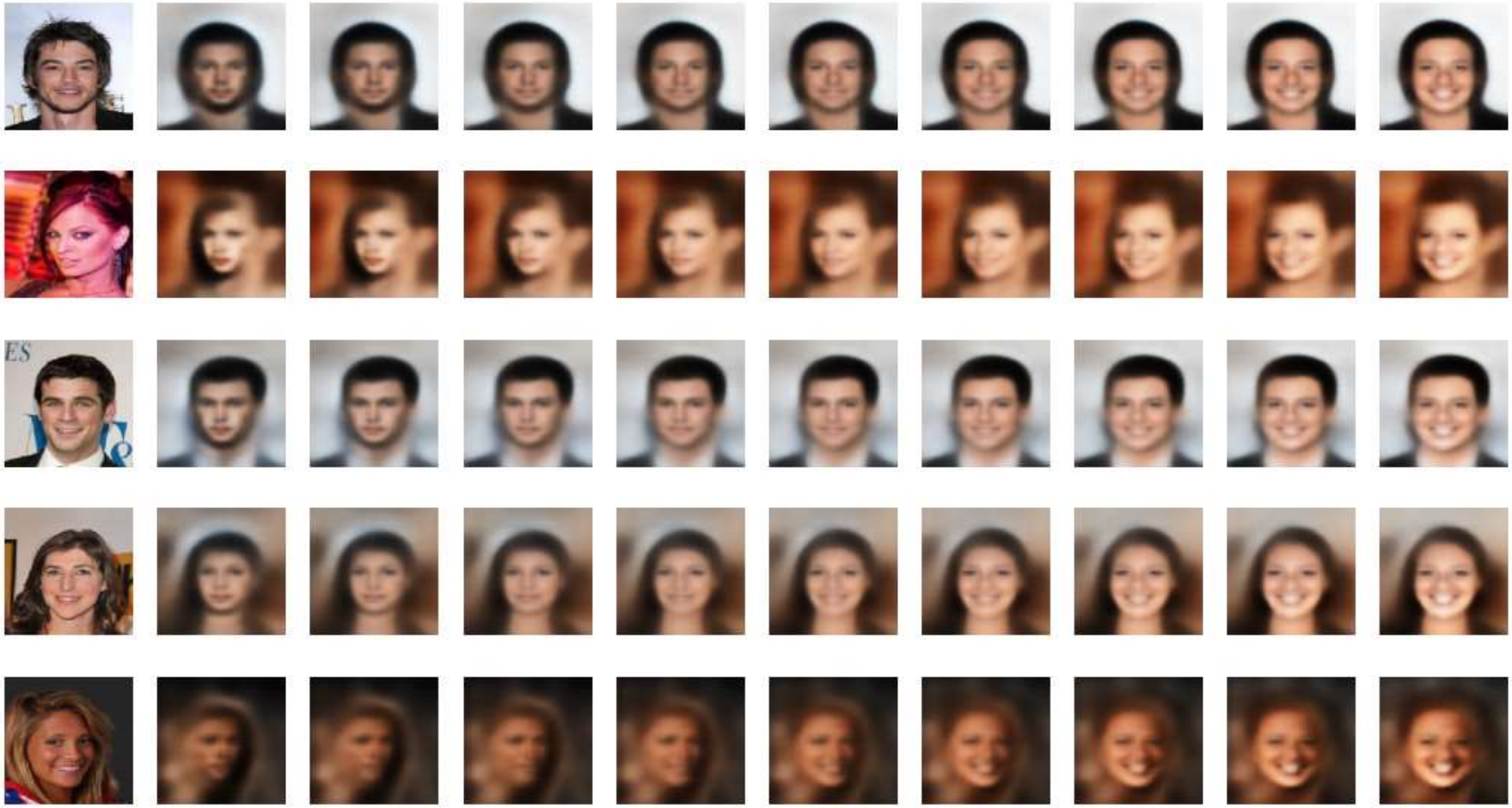
Epoch 00004: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 252s 957ms/step - batch: 131.5000 - size: 767.4205 - loss: 214.9620 - vae_r_loss: 156.0432 - vae_kl_loss: 58.9192 - lr: 1.0000e-06
Epoch 5/100
264/263 [==============================] - ETA: 0s - batch: 131.5000 - size: 767.4205 - loss: 214.9365 - vae_r_loss: 155.9996 - vae_kl_loss: 58.9345
Epoch 00005: saving model to run/vae/0001_faces\weights\weights-005-214.93.h5

Epoch 00005: saving model to run/vae/0001_faces\weights\weights.h5
263/263 [==============================] - 253s 957ms/step - batch: 131.5000 - size: 767.4205 - loss: **214.9365** - vae_r_loss: **155.9996** - vae_kl_loss: **58.9345** - lr: 1.0000e-06

NEW FACES



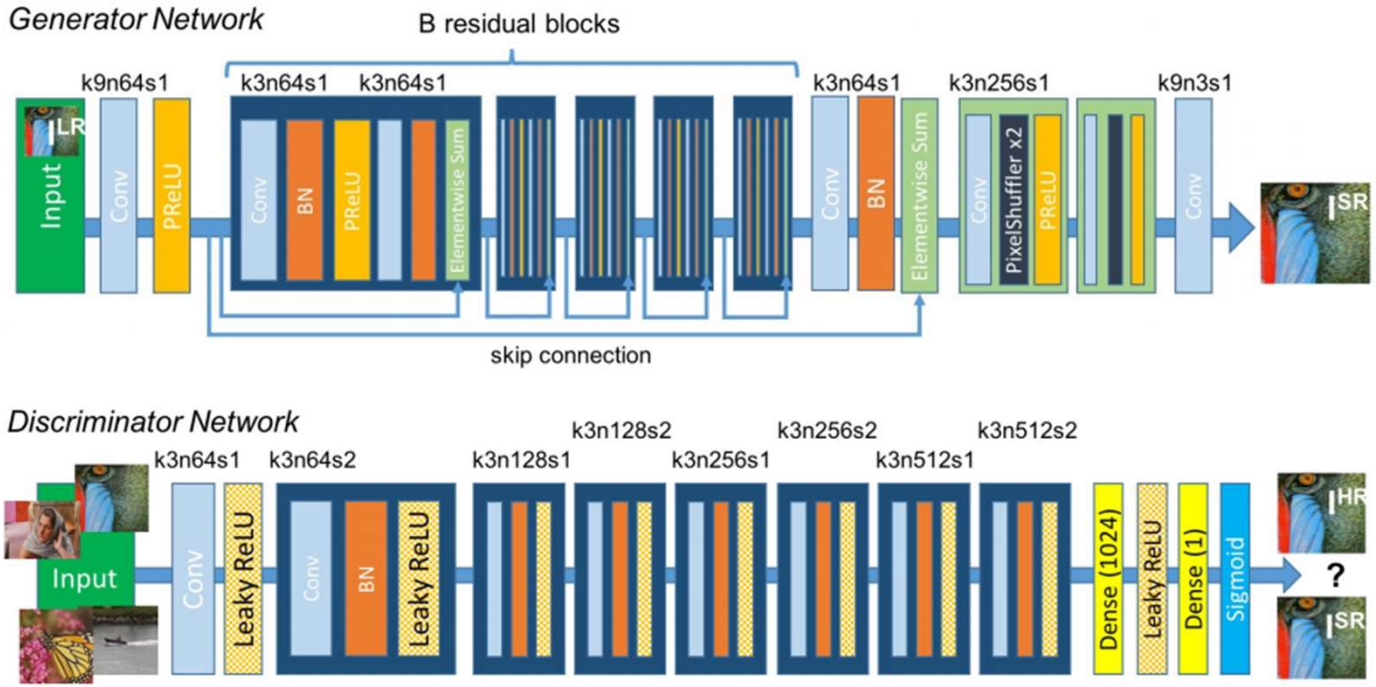
SMILING FACES

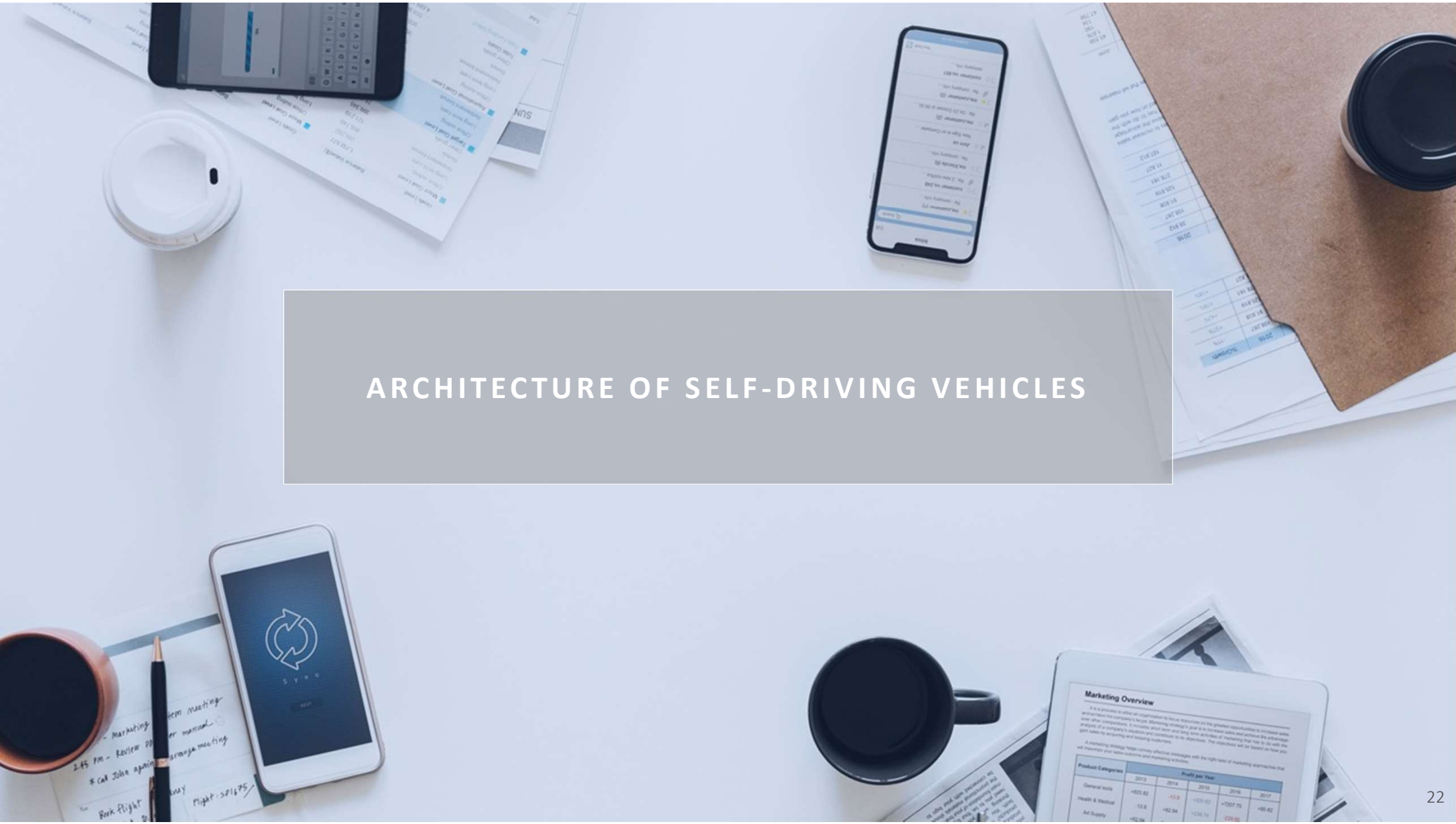


GAN

Super-Resolution GAN

https://github.com/eriklindernoren/PyTorch-GAN#super-resolution-gan





ARCHITECTURE OF SELF-DRIVING VEHICLES

ARCHITECTURE

Environment

Decoder (trained with VAE)

h

Actual Latent Space

Memories

h_{-n}

LSTM

Cell 1 Cell 2 ... Cell n

h_{+1} ... h_{+m}

Predictions

Controller

Steering wheel

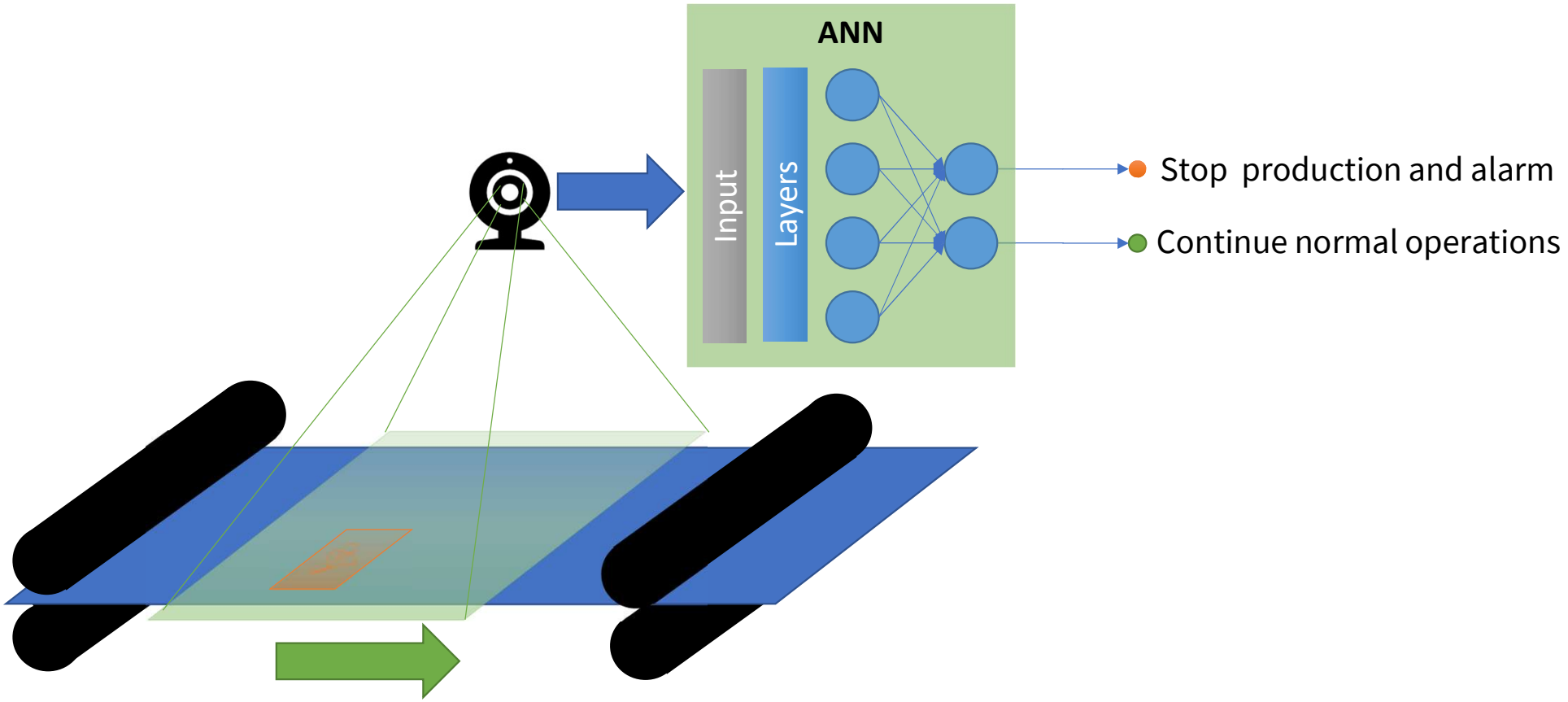
Accelerator pedal

Brakes

APPLICATIONS FOR BUSINESS & HOBBY

BUSINESS APPLICATIONS

Fabric factory – defects (anomaly) detection



BUSINESS APPLICATIONS

Face id, objects detection and classification

Decoder (trained with VAE)

Representation vector

Face id

We can use previously saved vectors from photos

Person 1 saved vector

Person 2 saved vector

...

Person n saved vector

to identify a person

Objects detection/classification

Detector

Output

MLP with Softmax

MLP or direct

Labels:
0. Face
1. Car
2. …

Bounding boxes

"Scanning window" approach

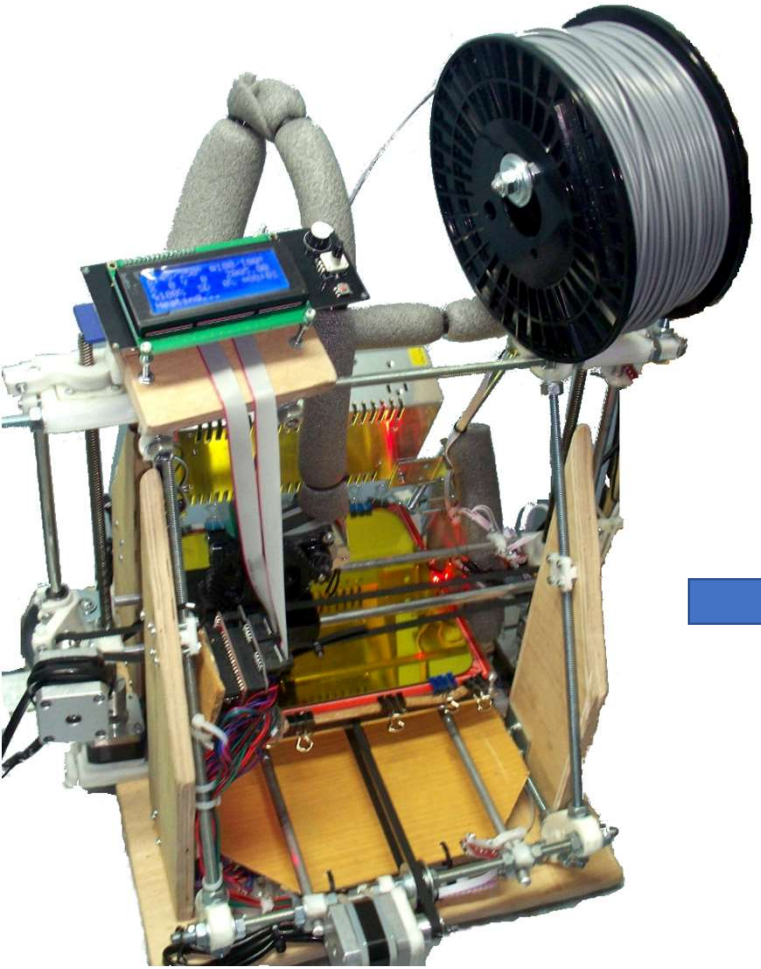
HOBBY

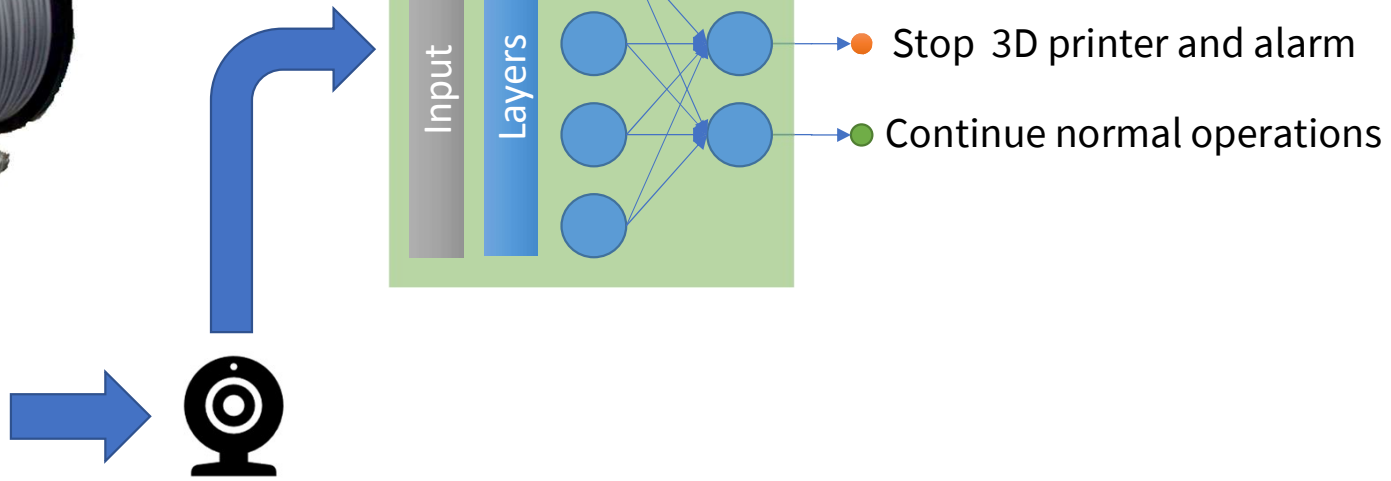
Intelligent homemade surveillance
with You Only Look Once (YOLO) 5

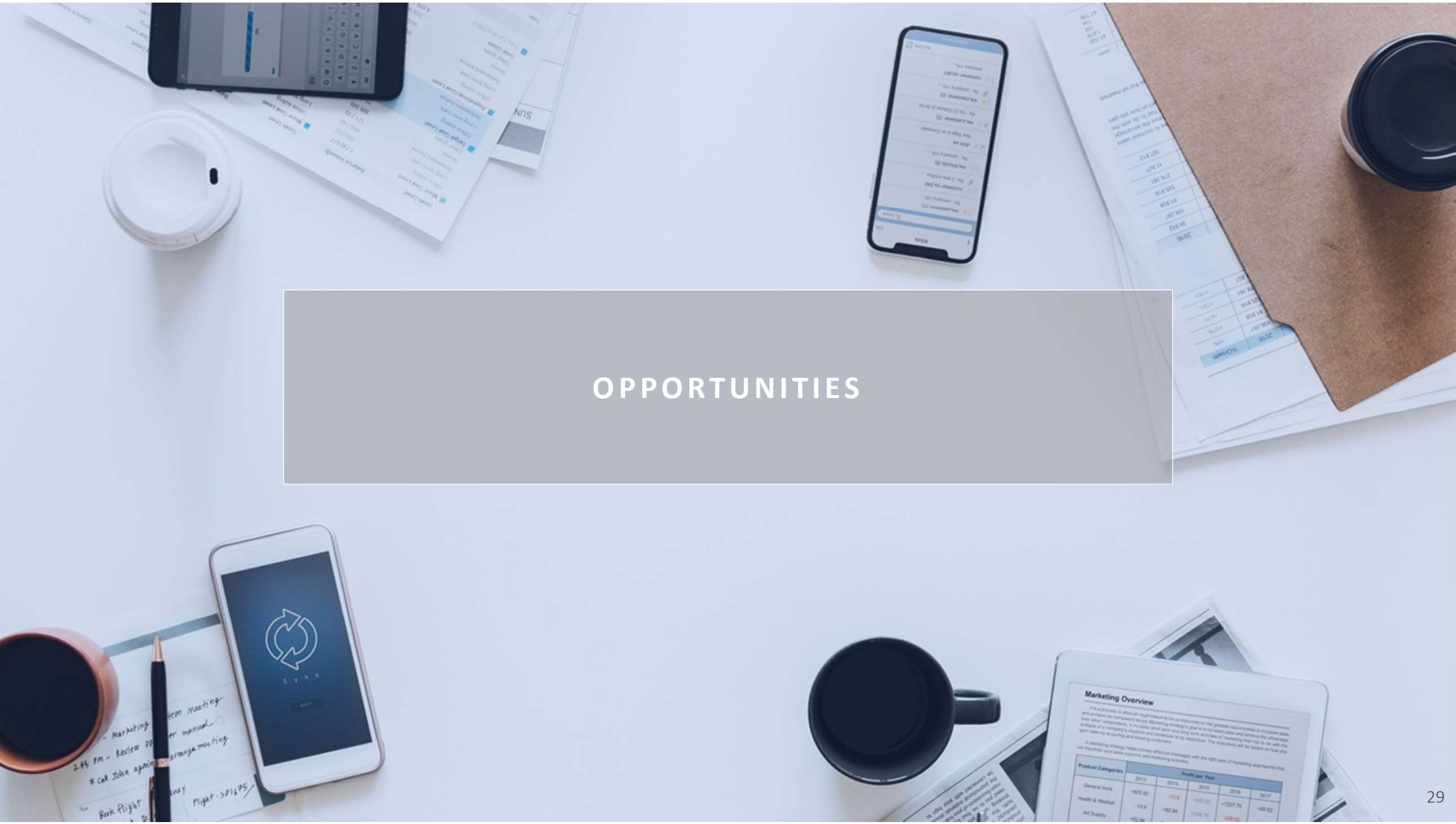
Reference - https://github.com/ultralytics/yolov5

HOBBY

Detect issues while 3D printing or during work of CNC





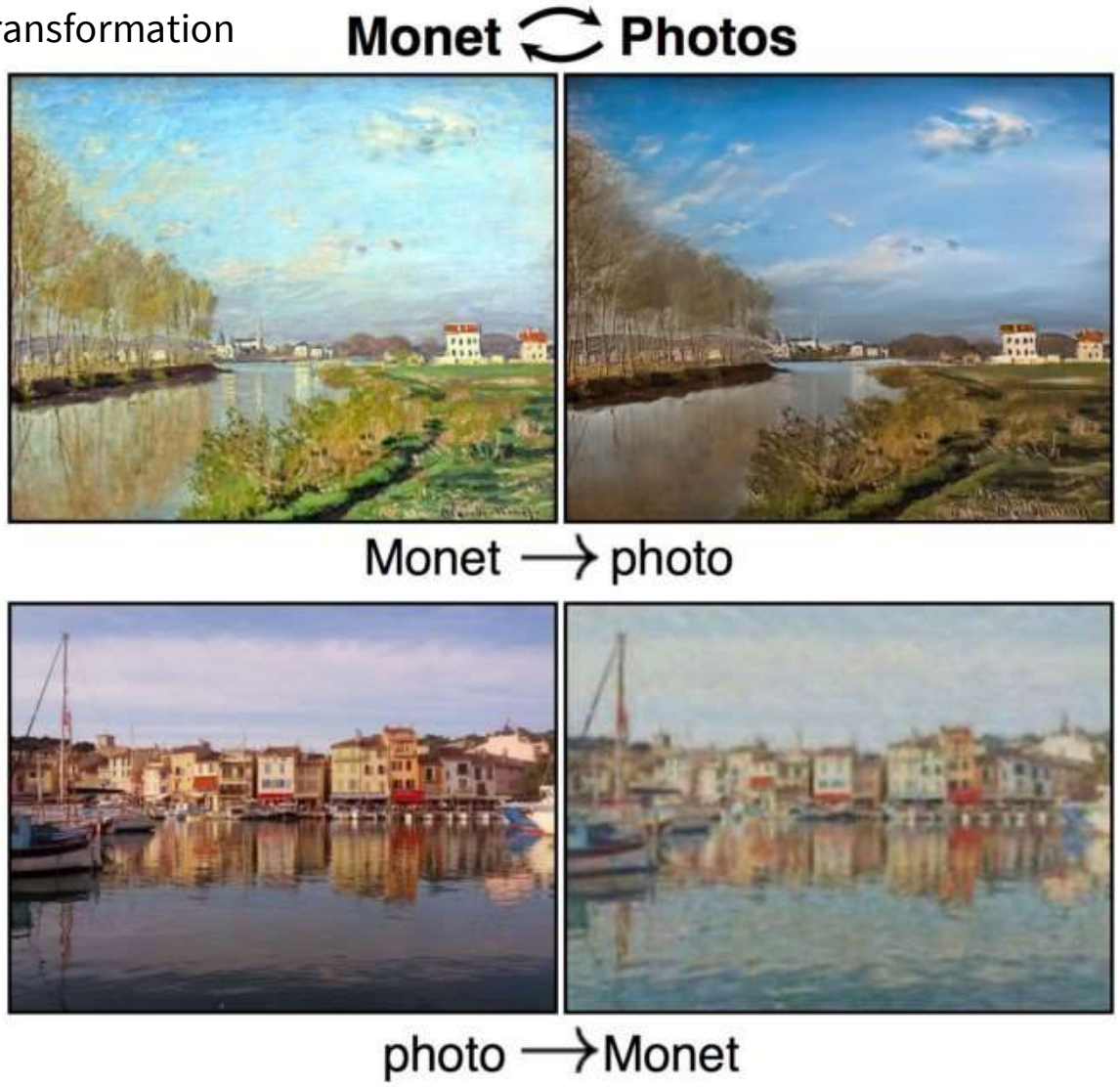


OPPORTUNITIES

OPPORTUNITIES

- What is creativity task in reality?
- Does it really fact that machines cannot perform some of creativity tasks?

Transformation



This example from medium:
https://medium.datadriveninvestor.com/style-transferring-of-image-using-cyclegan-3cc7aff4fe61

FUTURE TRENDS

CURRENT AND FUTURE

Future trends

Self-driving cars. Here and still, they are performing worse than ordinary driver - noticed accidents with self-driving cars

Home robots. Only simple commands and actions, even turning on lights can be an issue, let alone using tools like screwdriver and so

Self-replicating machines. It is hard to replicate in metal and other components on that simple organisms like bacteria do perfectly, like it with their self-replication

AI enriched cars and other vehicles. CV can detect potential collisions and brake much faster than people can

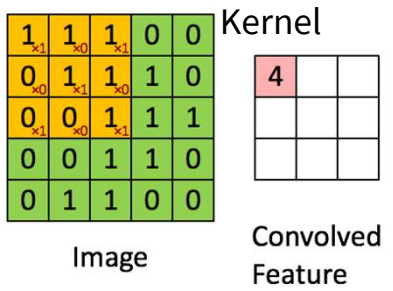
AI enabled devices. Like we see it with surveillance cameras and CV, also Commands understanding with LSTM/VAE can bring human to machine interaction to a new level

Factory automation. As we can see from the fabric factory example, we can bring both production effectiveness and quality to another level

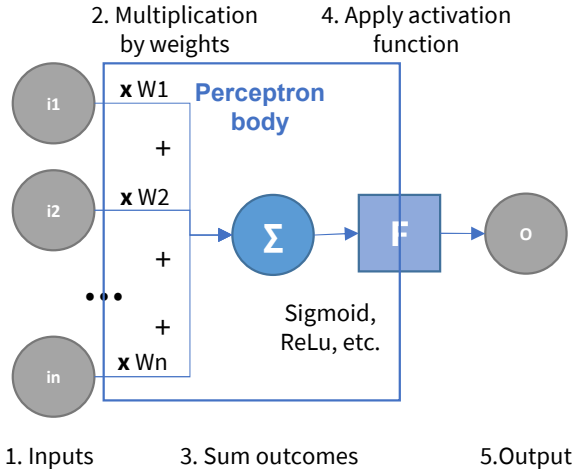

THANK YOU

EVOLUTION

MNIST dataset (illustration from Wiki)



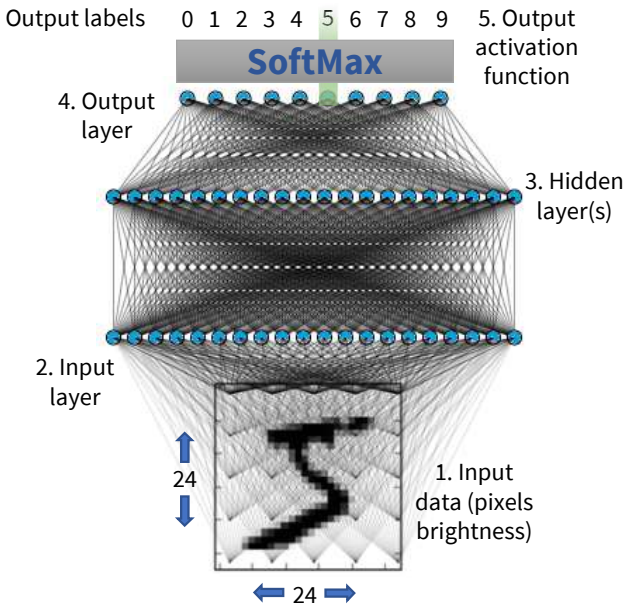
Perceptron



$$o = F(\sum_{k=1}^{n} i_k * w_k)$$

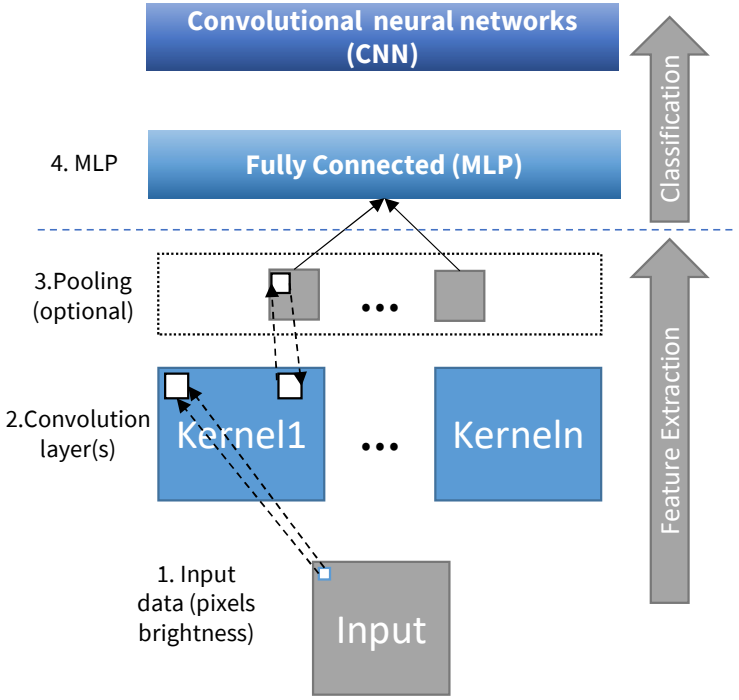
Less than 93% accuracy

Multilayer perceptron (MLP or FCN or Dense layers)



97%+

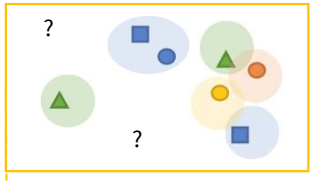
Convolutional neural networks (CNN)



99%+

RECENT

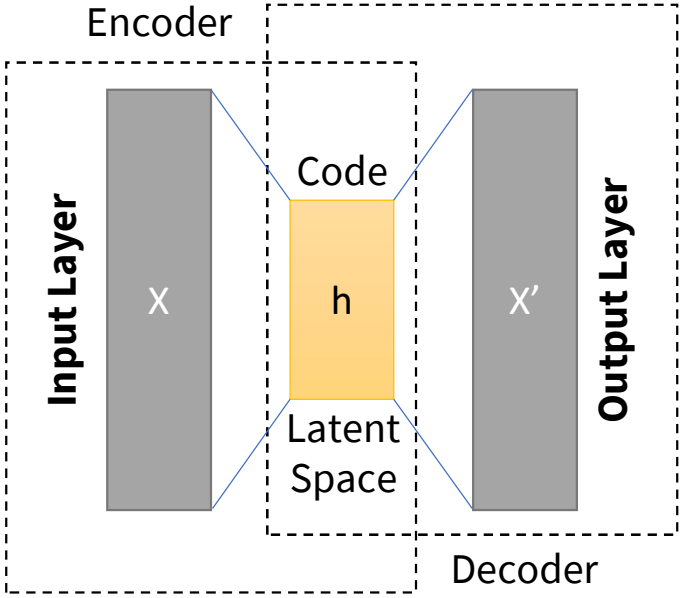
Latent Space regularization



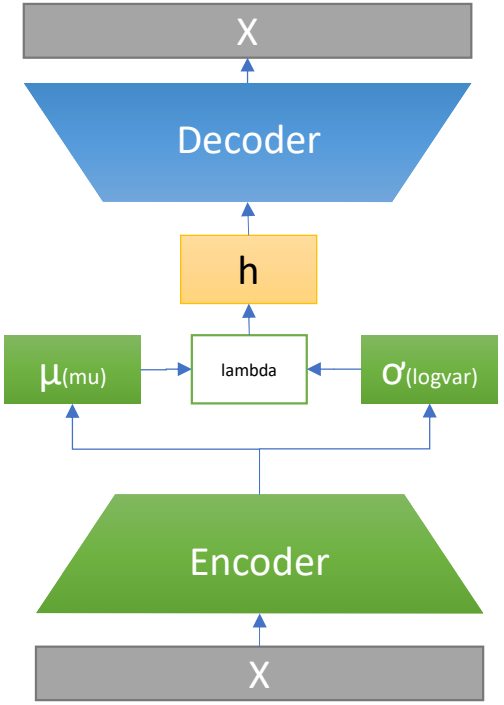
Near points can be different after decoding

Points close in latent space are similar after decoding

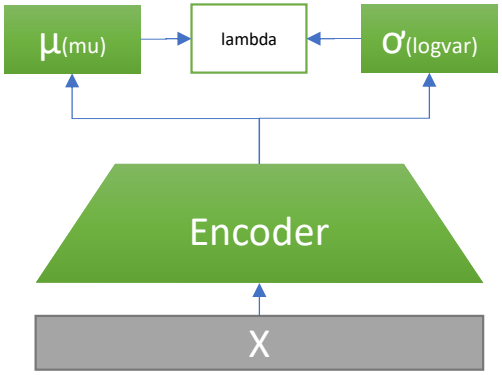
Autoencoder



Variational autoencoder (VAE)



ARCHITECTURE

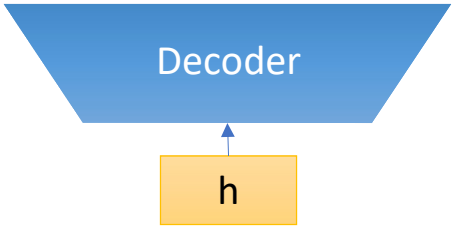


Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
encoder_input (InputLayer)	[(None, 128, 128, 3)]	0	[]
encoder_conv_0 (Conv2D)	(None, 64, 64, 32)	896	['encoder_input[0][0]']
batch_normalization (BatchNormalization)	(None, 64, 64, 32)	128	['encoder_conv_0[0][0]']
leaky_re_lu (LeakyReLU)	(None, 64, 64, 32)	0	['batch_normalization[0][0]']
dropout (Dropout)	(None, 64, 64, 32)	0	['leaky_re_lu[0][0]']
encoder_conv_1 (Conv2D)	(None, 32, 32, 64)	18496	['dropout[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 64)	256	['encoder_conv_1[0][0]']
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 64)	0	['batch_normalization_1[0][0]']
dropout_1 (Dropout)	(None, 32, 32, 64)	0	['leaky_re_lu_1[0][0]']
encoder_conv_2 (Conv2D)	(None, 16, 16, 64)	36928	['dropout_1[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256	['encoder_conv_2[0][0]']
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 64)	0	['batch_normalization_2[0][0]']
dropout_2 (Dropout)	(None, 16, 16, 64)	0	['leaky_re_lu_2[0][0]']
encoder_conv_3 (Conv2D)	(None, 8, 8, 64)	36928	['dropout_2[0][0]']
batch_normalization_3 (BatchNormalization)	(None, 8, 8, 64)	256	['encoder_conv_3[0][0]']
leaky_re_lu_3 (LeakyReLU)	(None, 8, 8, 64)	0	['batch_normalization_3[0][0]']
dropout_3 (Dropout)	(None, 8, 8, 64)	0	['leaky_re_lu_3[0][0]']
flatten (Flatten)	(None, 4096)	0	['dropout_3[0][0]']
mu (Dense)	(None, 200)	819400	['flatten[0][0]']
log_var (Dense)	(None, 200)	819400	['flatten[0][0]']
encoder_output (Lambda)	(None, 200)	0	['mu[0][0]', 'log_var[0][0]']

Total params: 1,732,944
Trainable params: 1,732,496
Non-trainable params: 448

ARCHITECTURE




```
vae = VariationalAutoencoder(
        input_dim = INPUT_DIM
        , encoder_conv_filters=[32,64,64, 64]
        , encoder_conv_kernel_size=[3,3,3,3]
        , encoder_conv_strides=[2,2,2,2]
        , decoder_conv_t_filters=[64,64,32,3]
        , decoder_conv_t_kernel_size=[3,3,3,3]
        , decoder_conv_t_strides=[2,2,2,2]
        , z_dim=200
        , use_batch_norm=True
        , use_dropout=True)
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
decoder_input (InputLayer)   [(None, 200)]             0

dense (Dense)                (None, 4096)              823296

reshape (Reshape)            (None, 8, 8, 64)          0

decoder_conv_t_0 (Conv2DTra  (None, 16, 16, 64)        36928
nspose)

batch_normalization_4 (Batc  (None, 16, 16, 64)        256
hNormalization)

leaky_re_lu_4 (LeakyReLU)    (None, 16, 16, 64)        0

dropout_4 (Dropout)          (None, 16, 16, 64)        0

decoder_conv_t_1 (Conv2DTra  (None, 32, 32, 64)        36928
nspose)

batch_normalization_5 (Batc  (None, 32, 32, 64)        256
hNormalization)

leaky_re_lu_5 (LeakyReLU)    (None, 32, 32, 64)        0

dropout_5 (Dropout)          (None, 32, 32, 64)        0

decoder_conv_t_2 (Conv2DTra  (None, 64, 64, 32)        18464
nspose)

batch_normalization_6 (Batc  (None, 64, 64, 32)        128
hNormalization)

leaky_re_lu_6 (LeakyReLU)    (None, 64, 64, 32)        0

dropout_6 (Dropout)          (None, 64, 64, 32)        0

decoder_conv_t_3 (Conv2DTra  (None, 128, 128, 3)       867
nspose)

activation (Activation)      (None, 128, 128, 3)       0

=================================================================
Total params: 917,123
Trainable params: 916,803
Non-trainable params: 320
```

HOW JAVA

SPARK ML, BIGDL
- Highly scalable solution
- Average number of algorithms
- TensorFlow on worker node

DATAROBOT
- Distributed CPUs and GPUs
- AUTOML

H20
- Deep learning (composed of multiple linear and non-linear transformations)
- Fine-Grain distributed processing

HOW PYTHON

SCIKIT LEARN

- Simple and efficient tools for data mining and data analysis used by Data Scientists
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib

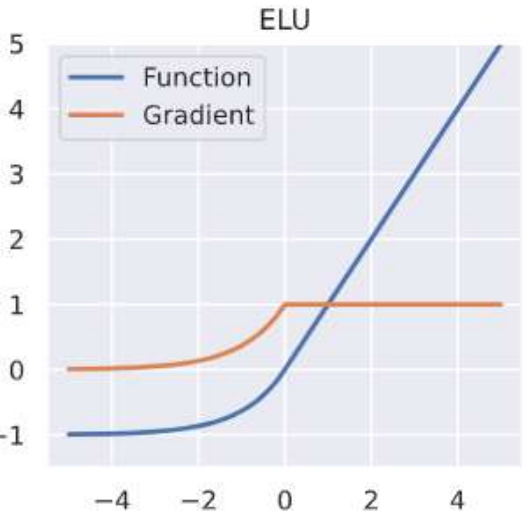
TENSORFLOW (KERAS)

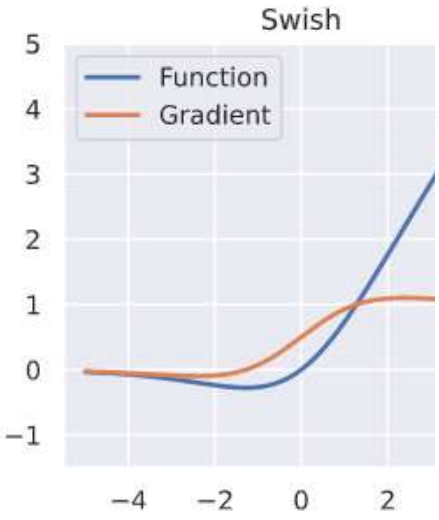
- Library for numerical computation using data flow graphs. Graphs are static
- Deployable to one or more CPUs or GPUs or Mobile

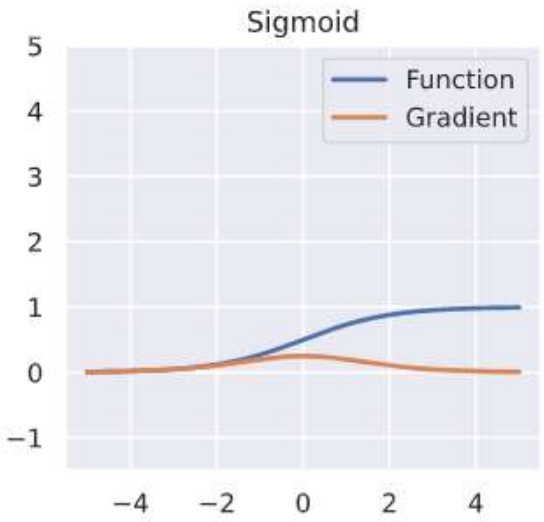
PYTORCH

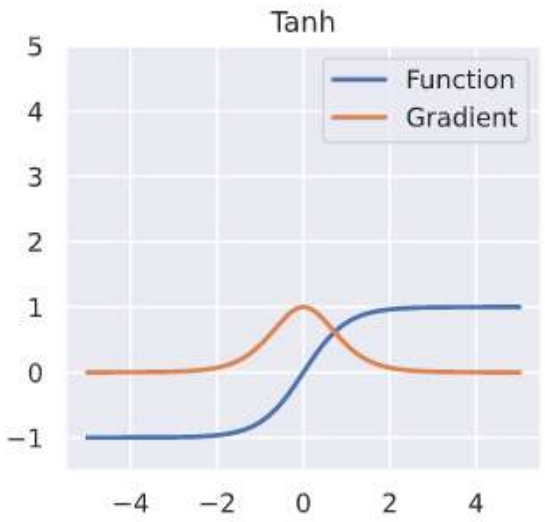
- Deep learning framework that puts Python first
- Supports for Dynamic NN
- Strong GPU acceleration

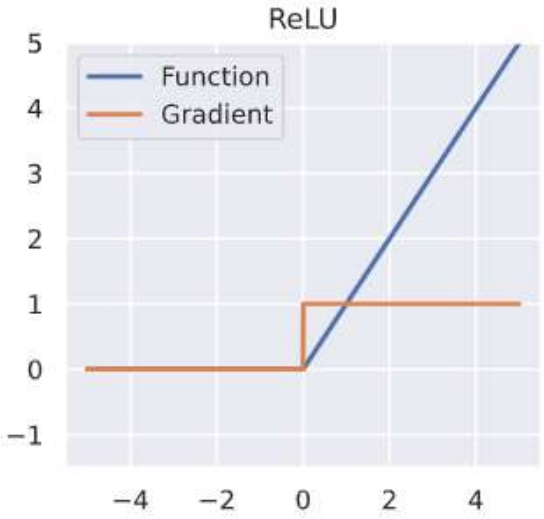
ACTIVATION FUNCTIONS

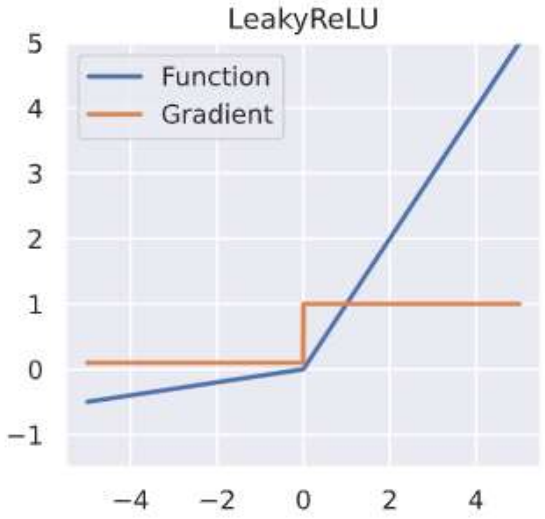






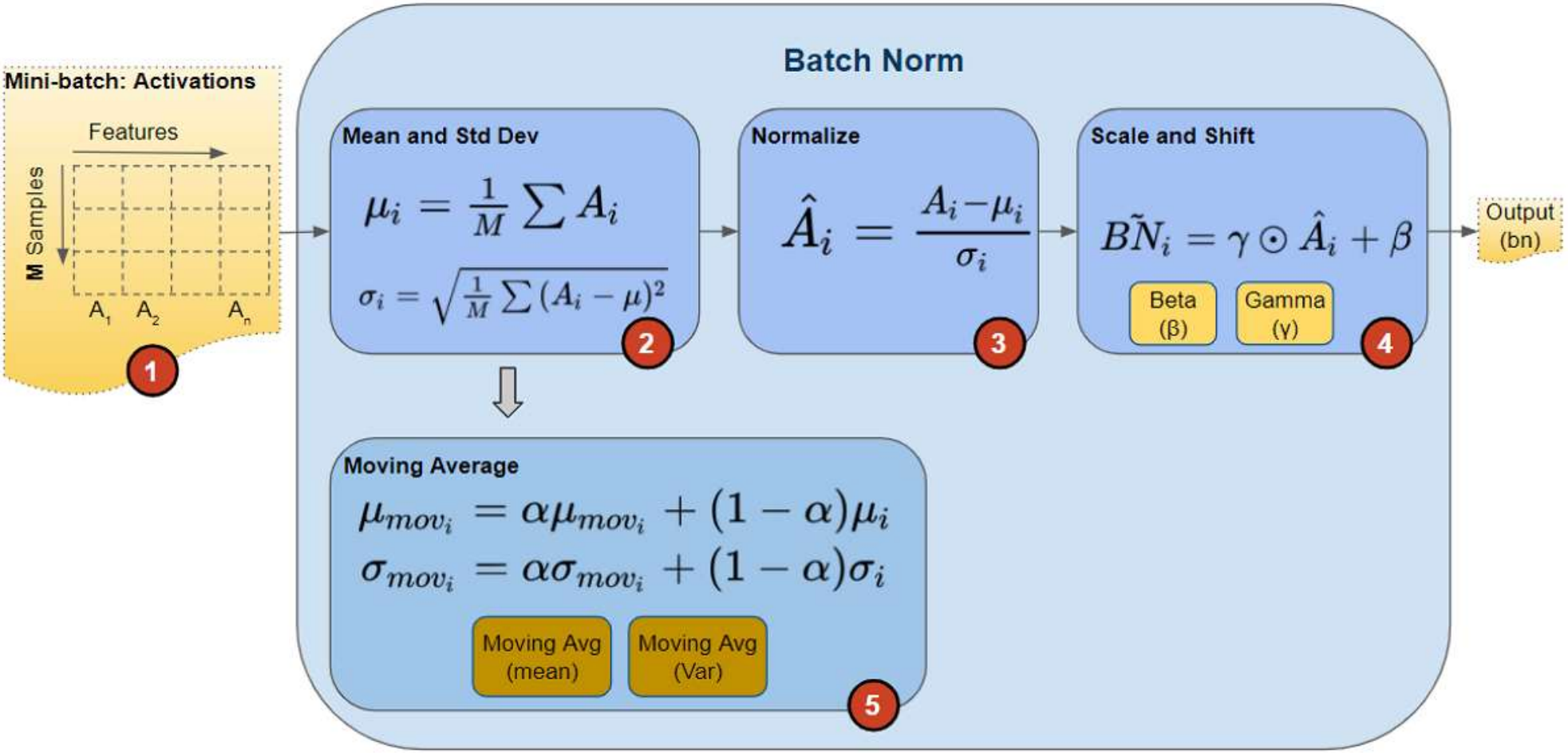






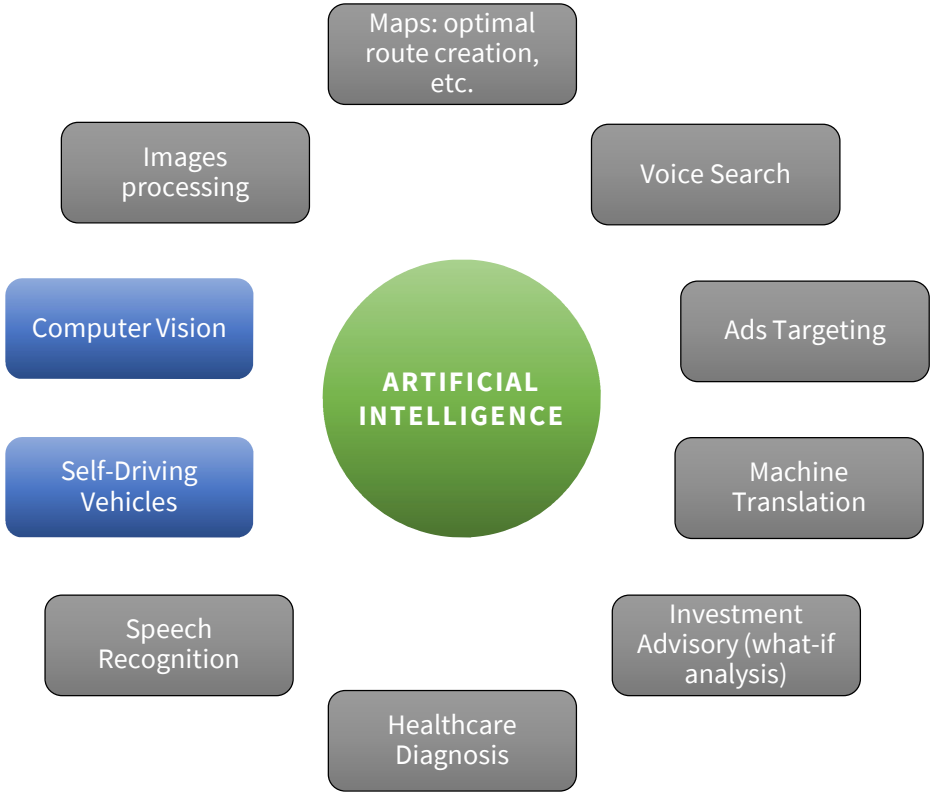
$$\text{SoftMax}(\mathbf{X})i' = \frac{\exp(\mathbf{X}i)}{\sum_k \exp(\mathbf{X}_k)}$$

BATCH NORMALIZATION



MOTIVATION

- Security
  - Face ID, People, objects and anomaly detection
- Automation
  - Work processes, Logistics and QC automation
- Predictions
  - Health condition, mechanical failure, insurance claim severity, etc.
- Recommendations
  - Amazon, Netflix, etc.
- Search
  - Atlas, Google, Wolfram, etc.
- Information Structuring
  - Chatbots, Alexa, etc.

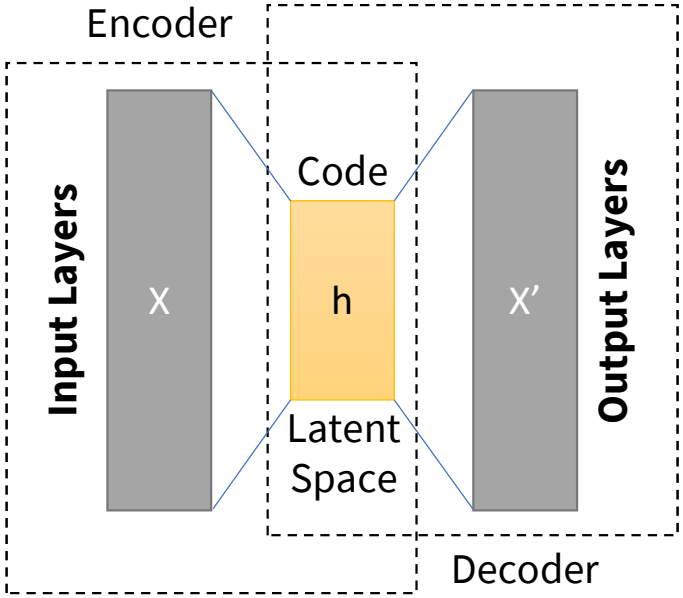


RECENT

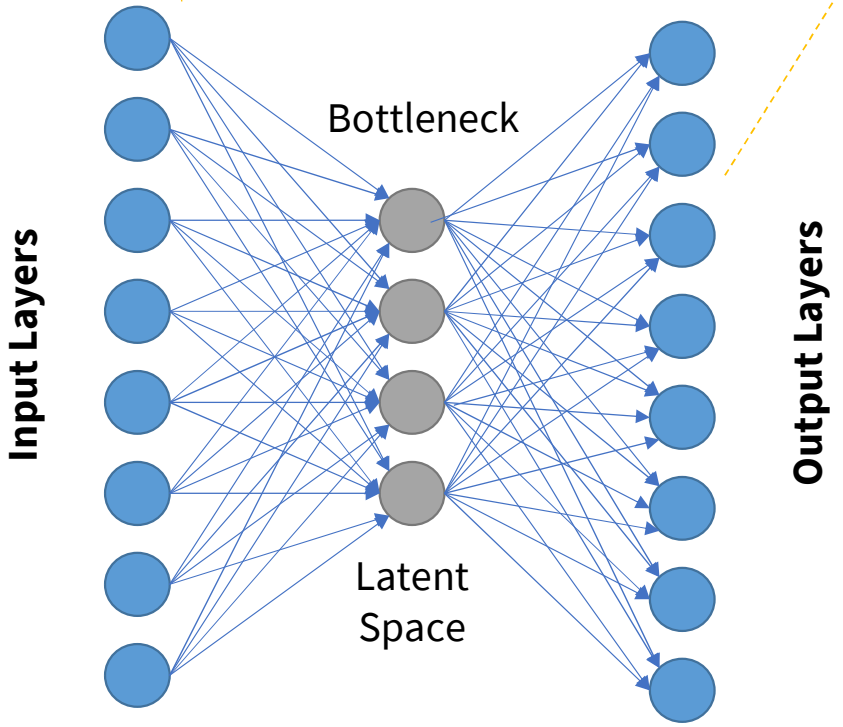
Latent Space regularization issue

Near points can be different after decoding

Autoencoder



MLP or Convolutional + MLP from previous slide



DROPOUT