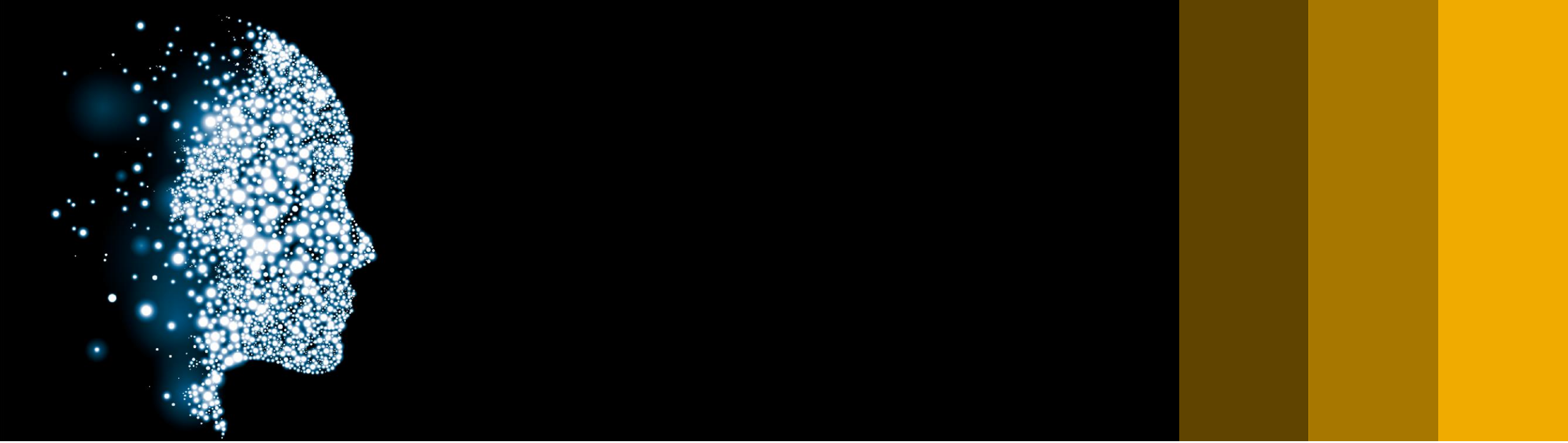


Week 6: Advanced Deep Learning Topics

Unit 1: Generating New Images Using GANs

SAP Run Simple

openSAP
open.sap.com

Generating New Images Using GANs

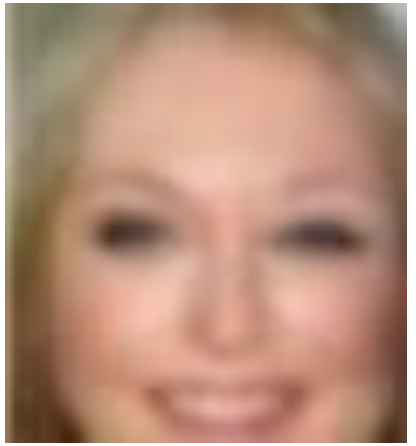
Intro: The what and why of GANs

What are GANs?

- A method for generating new images based on an existing dataset!
- Consist of two competing networks:
  - A generative network (G) that generates fake images
  - A discriminative network (D) that can pick apart fakes from your real dataset

Why GANs for image generation?

- Past methods (such as VAEs) have a visually distinct ‘blurriness’ compared to GANs
- Discriminative (classification) models would be able to tell the difference easily!



Before GANs:
Images tended to be blurry or unclear

GANs: Images look similar to real images – in this case a real face

Generating New Images Using GANs

The two networks: the generator

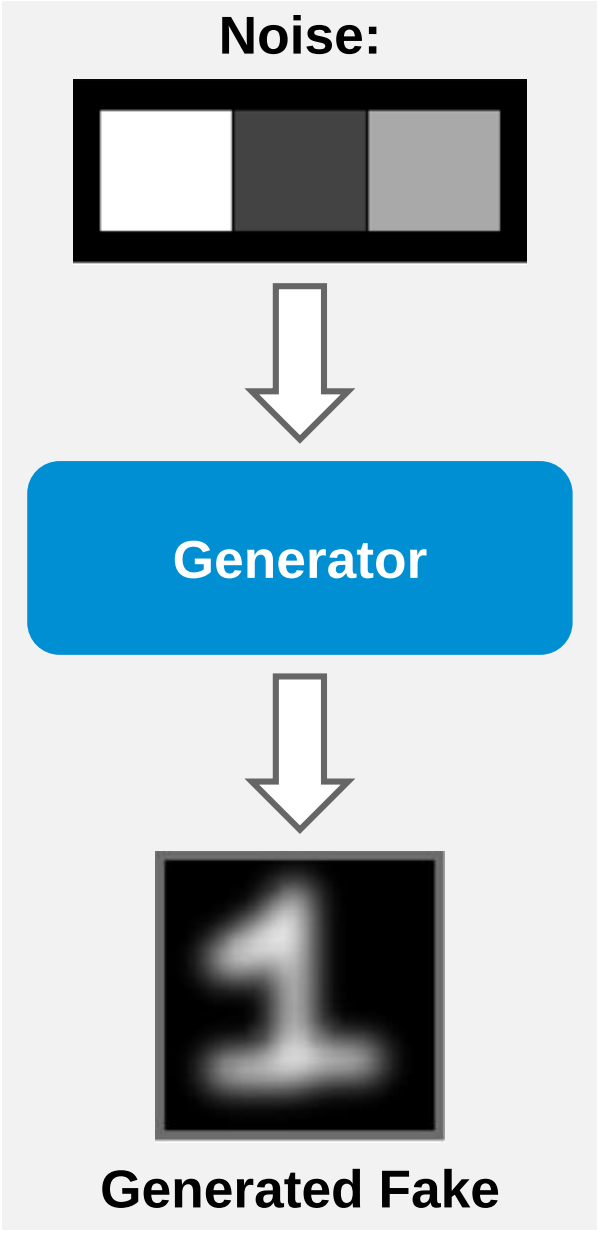
The **generator** is the most important part of the network, as it is what we use in the end!

Why do we want to generate images?

- Generating images is a difficult task
- GANs yield visually convincing results
- Can be used for data inspection or for dataset augmentation when samples are lacking

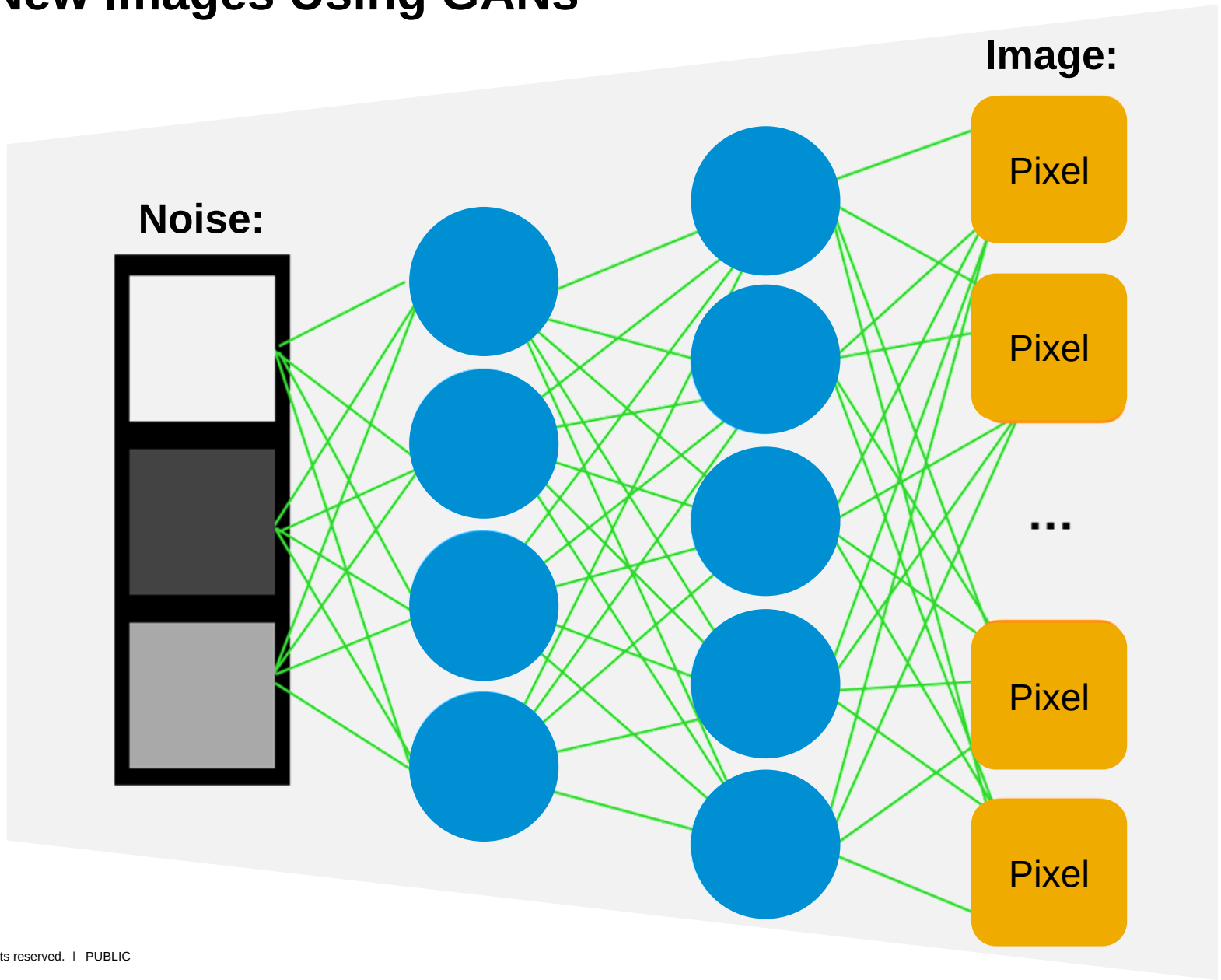
How do we generate images?

- Learn to transform Gaussian noise into a realistic looking image!
- Input: a vector of Gaussian noise
- Output: a flattened image – essentially just a vector of pixels



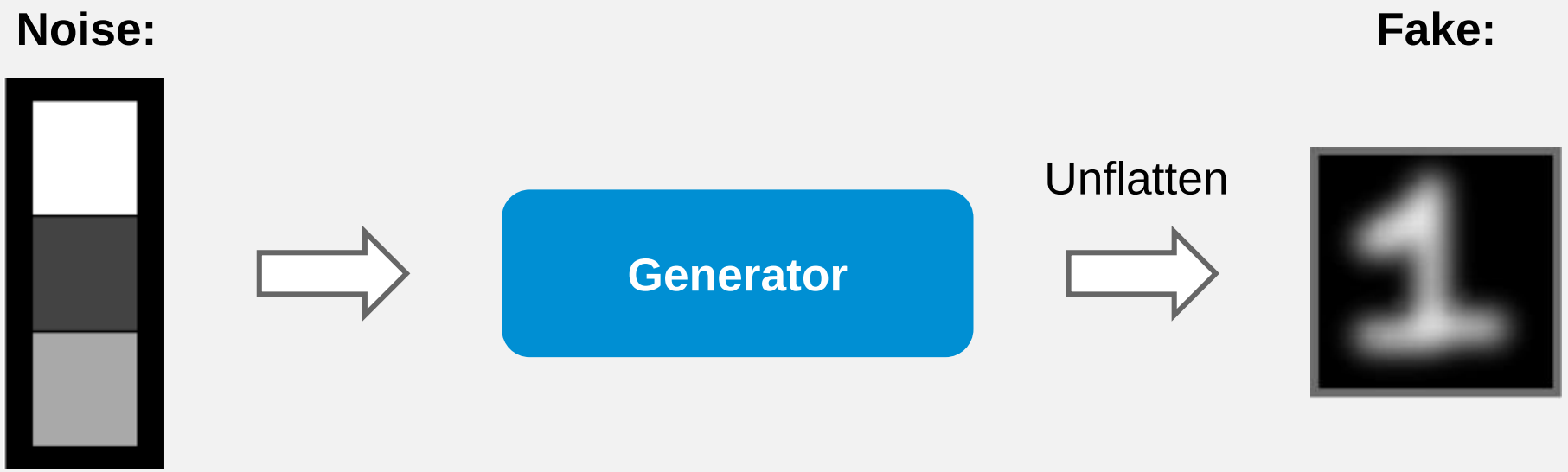
Generating New Images Using GANs

Generator



Generating New Images Using GANs

Generator



Generator: Turns noise into convincing fakes

Generating New Images Using GANs

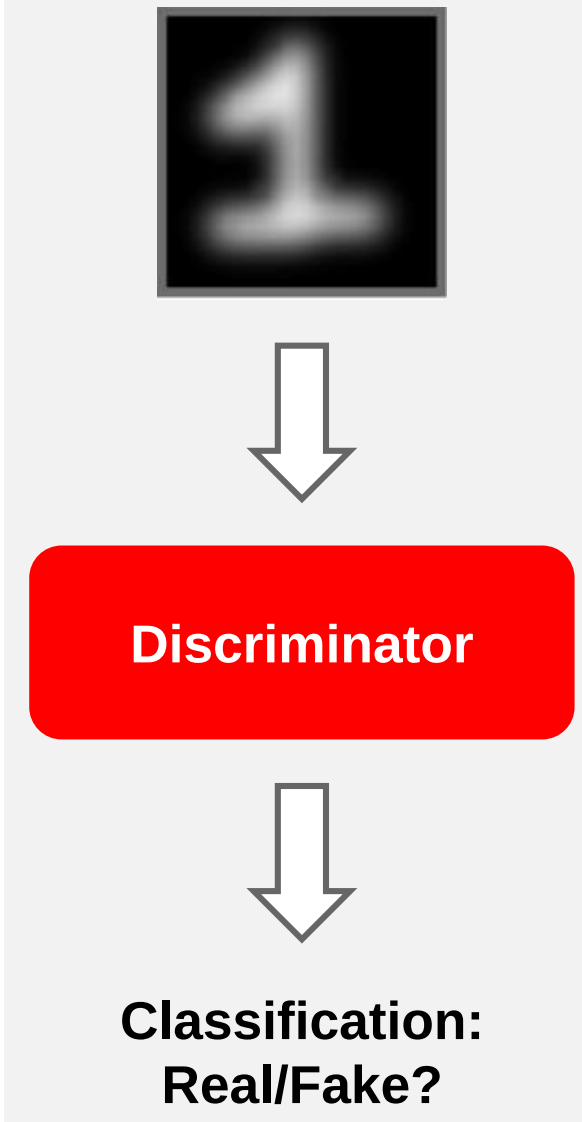
The two networks: the discriminator

Why do we want to discriminate images?

- Discriminative (classification) models have superhuman performance
- If generated images look ‘off’, discriminative models should be able to tell the difference
- Key: Even though we call the networks adversarial, the discriminator actually helps the generator improve!

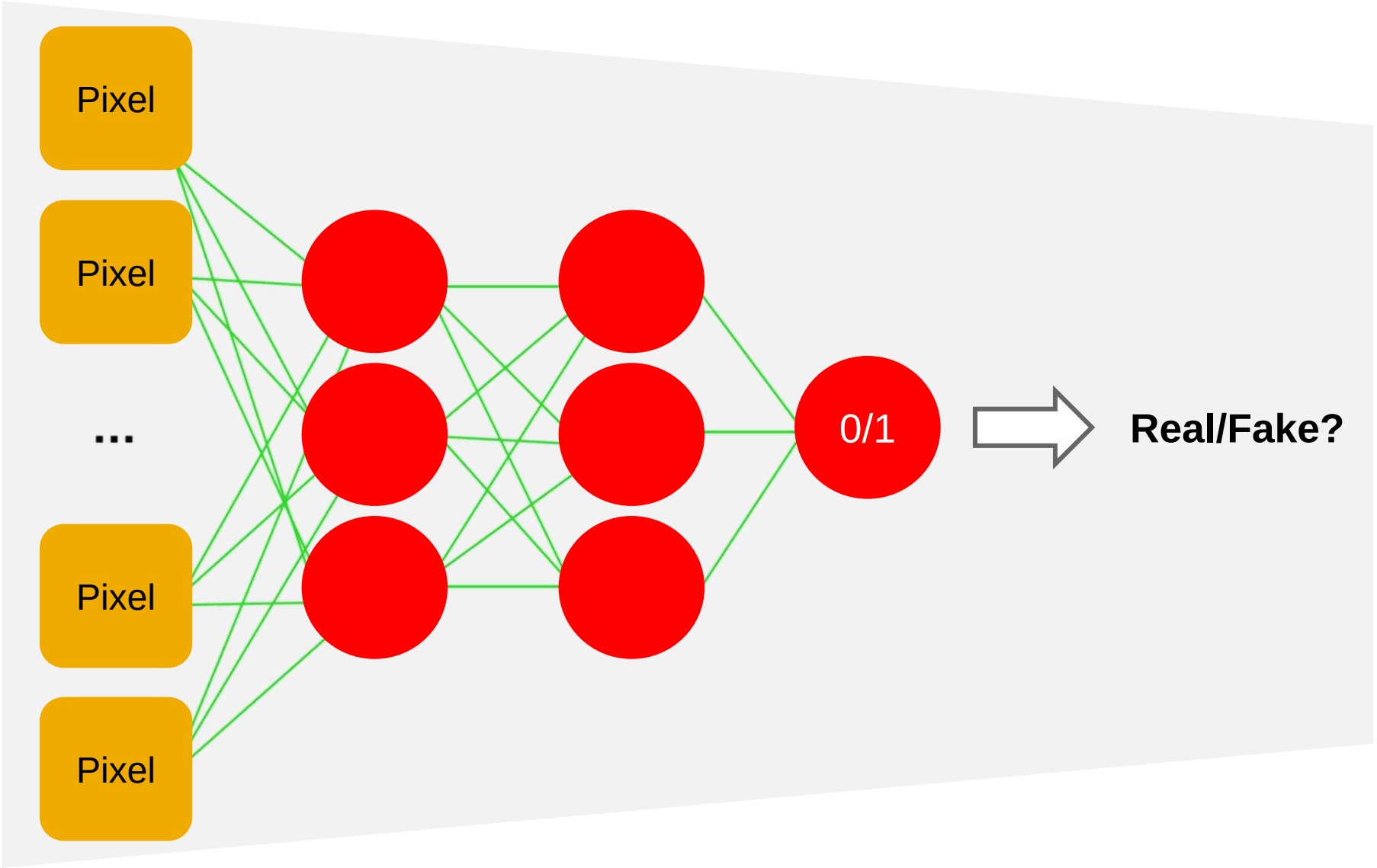
How do we discriminate images?

- Classic binary classification task – real image from dataset, or fake image from generator?
- No different than building e.g. a dog vs cat classifier, architectures can be identical



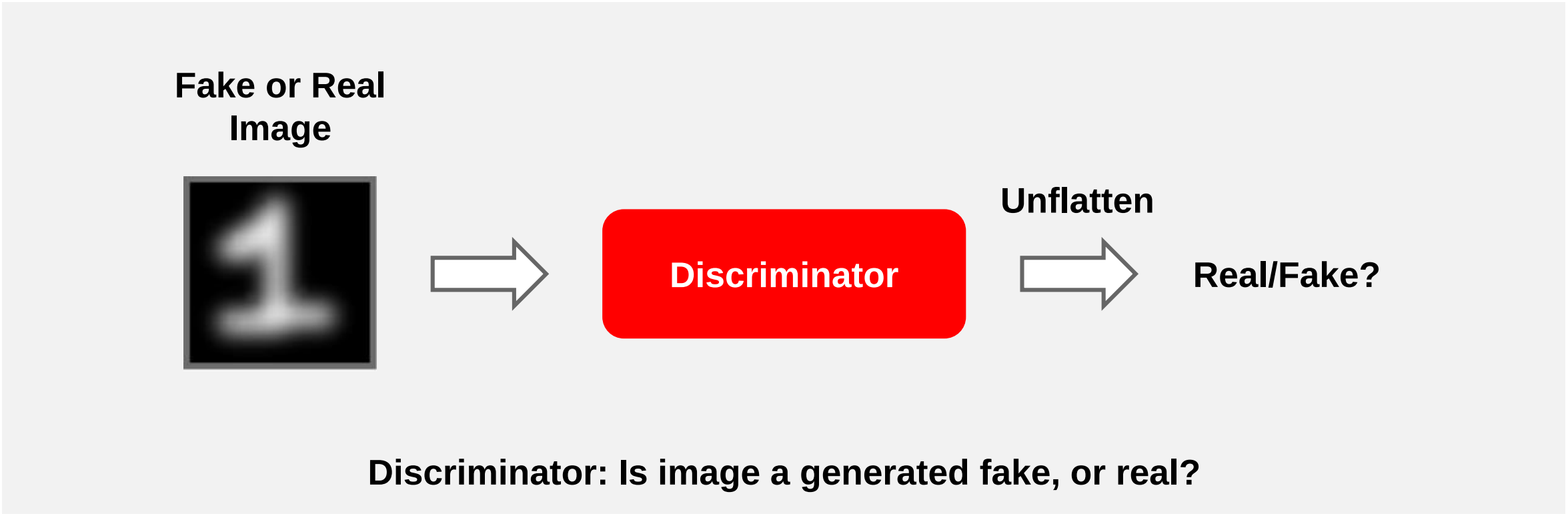
Generating New Images Using GANs

Discriminator



Generating New Images Using GANs

Discriminator



Discriminator: Is image a generated fake, or real?

Generating New Images Using GANs

Overview: the adversarial training process

Networks take turns training!

Discriminator:

- Takes the first turn
- Get several batches of images from training dataset, and several from generator
- Try to tell real dataset images vs generated fakes!

Generator:

- Key network we care about – it will do the final image generation when training is done
- Learns from discriminator how to generate better fake images based on a ‘combined network’
- Never directly sees a training image!

Generator:

Turn 0:

...

Turn X:

...

Final Turn:

Generating New Images Using GANs

Training the discriminator

Fake Image (from Generator)

Discriminator

Fake!

Real Image (from Dataset)

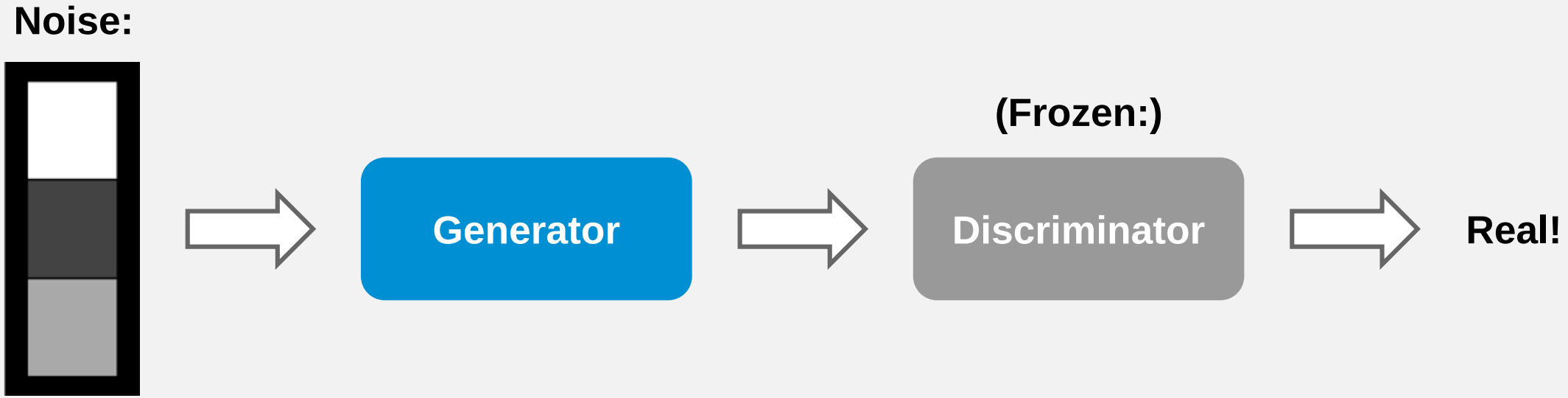
Discriminator

Real!

Straightforward Classification Task!

Generating New Images Using GANs

Training the generator

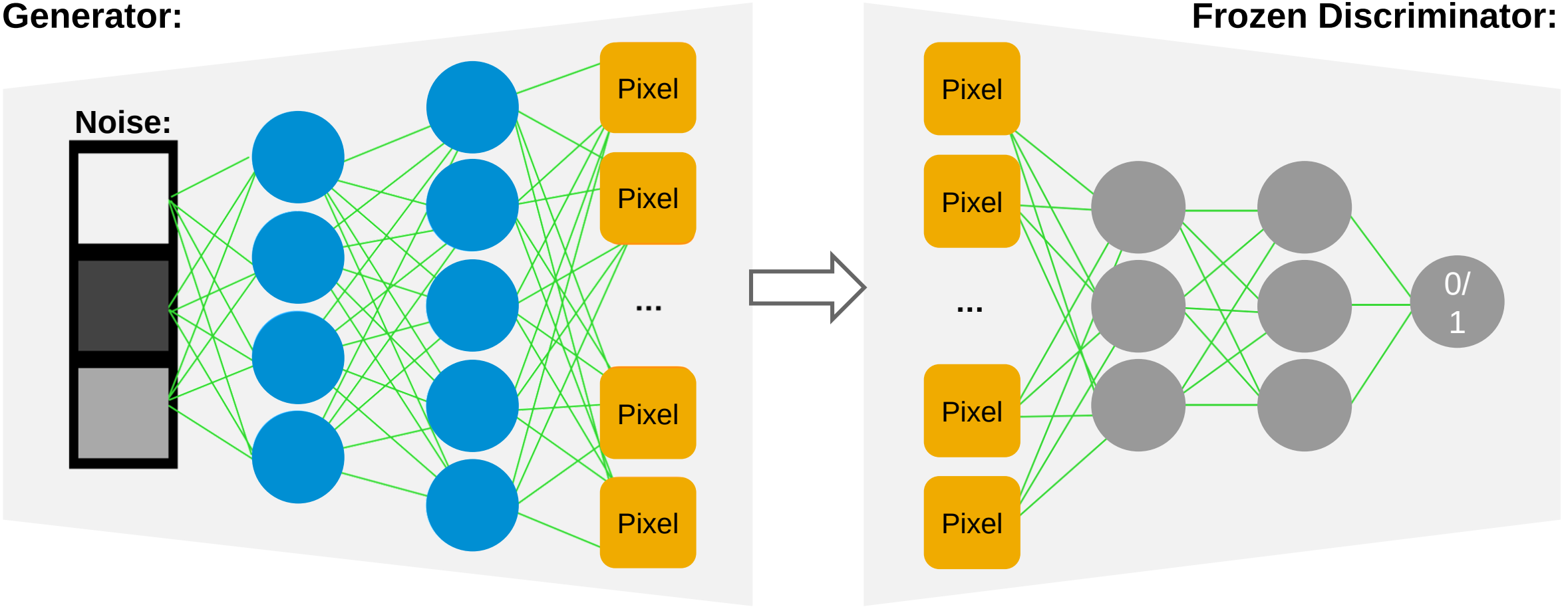


Train generator: Combine both networks!
Goal: G learns to get D to predict ‘Real’

Note: D is not updated – it is G‘s turn!

Generating New Images Using GANs

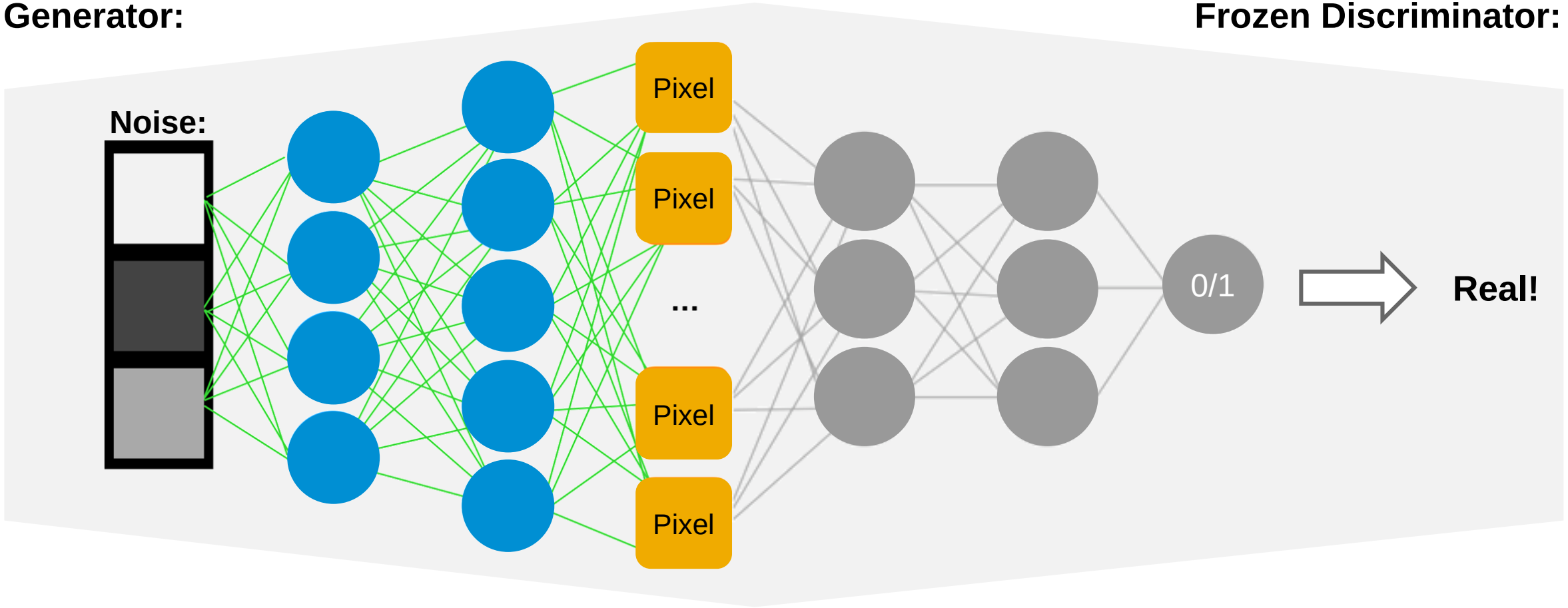
Discriminator ‘helps’ generator



Key: Discriminator tells G how to improve!!

Generating New Images Using GANs

G+D training network (to train generator)



Note the Label! G is learning to fool D.
Noise normally would be labeled ‘Fake’!

Generating New Images Using GANs

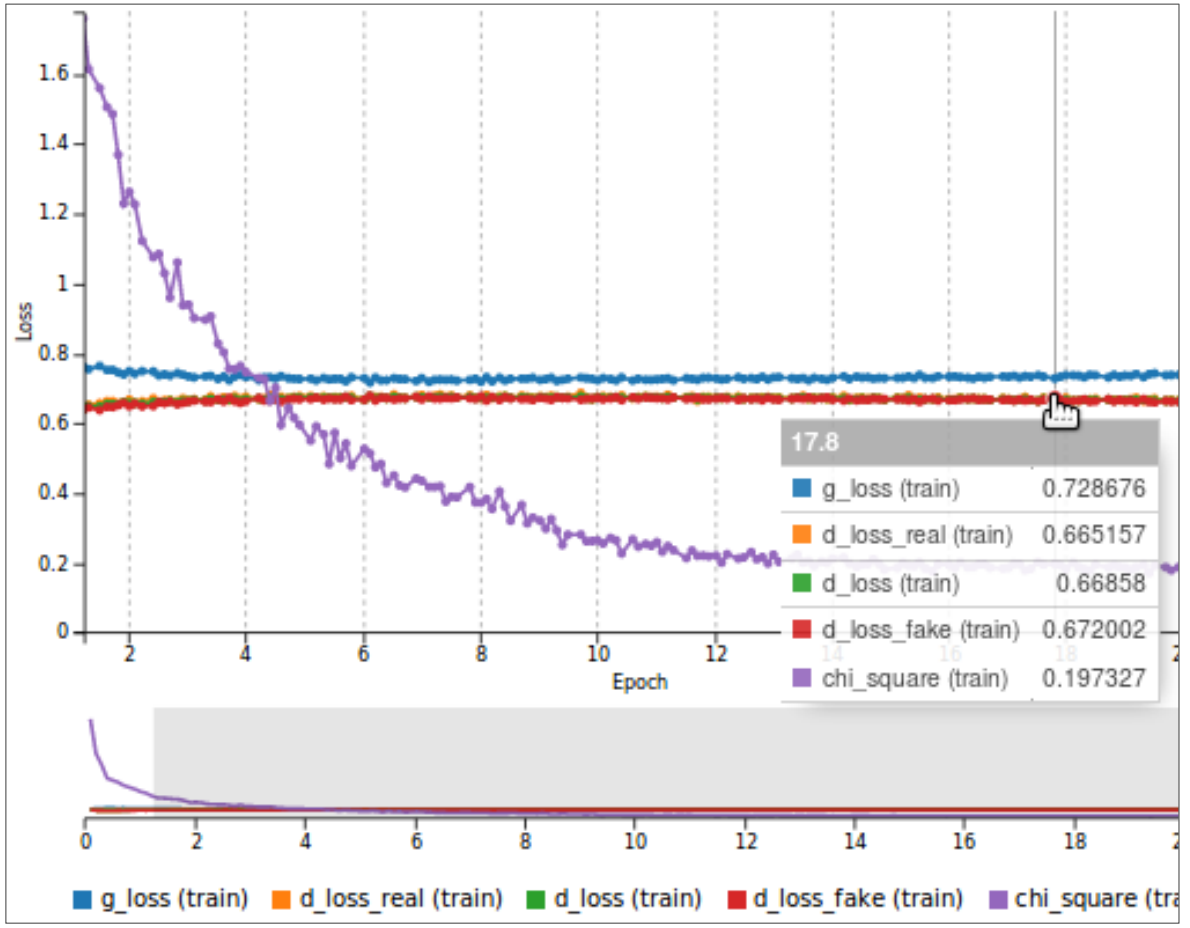
GANs: some key challenges

Getting Generator to Converge:

- No single metric to measure how well network does, so harder to optimize
- One important trick: Since D is the only source of info for G, a well-trained D is vital

Working with Discrete Data:

- GANs today are applied primarily to images, which are easy to represent as continuous
- Due to the G+D network training step, data in typical GAN structures must be continuous
- Some extensions of GANs attempt to address this, but in many situations VAEs and other approaches can be better for discrete data



Pictured: The losses of a GAN while it trains. Note the loss is continuous – it would not be defined for discrete data!

Generating New Images Using GANs

Extensions of ‘classic GAN’ architecture

Deep Convolutional GANs: (DC-GANs)

- One natural extension: instead of dense layers, use convolutional layers end to end!
- For image tasks, this yields significantly better results when properly tuned

Conditional GANs:

- GANs can also be used in a more supervised setting, where G and D are also fed labels
- This allows you to feed the network’s labels at runtime to generate specific outputs!

Pictured: A conditional DC-GAN morphing a generated female face into a generated male face with similar attributes

Generating New Images Using GANs

Summary

GANs are a useful way of **generating** visually convincing images based on a dataset

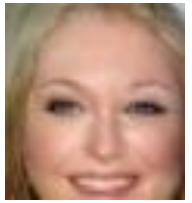
GANs consist of two **competing adversarial** networks

- The *generator* network creates convincing fake images
- The *discriminator* learns to tell faked images vs real images from the dataset

Networks take turns improving at their jobs!

Generating images with **adversarial** back-and-forth training of two **networks**!

Pictured: Another interesting conditional DC-GAN application: transforming an image by adding or subtracting labels



Original Generated Image

Original + Add Young

Original + Subtract Blond

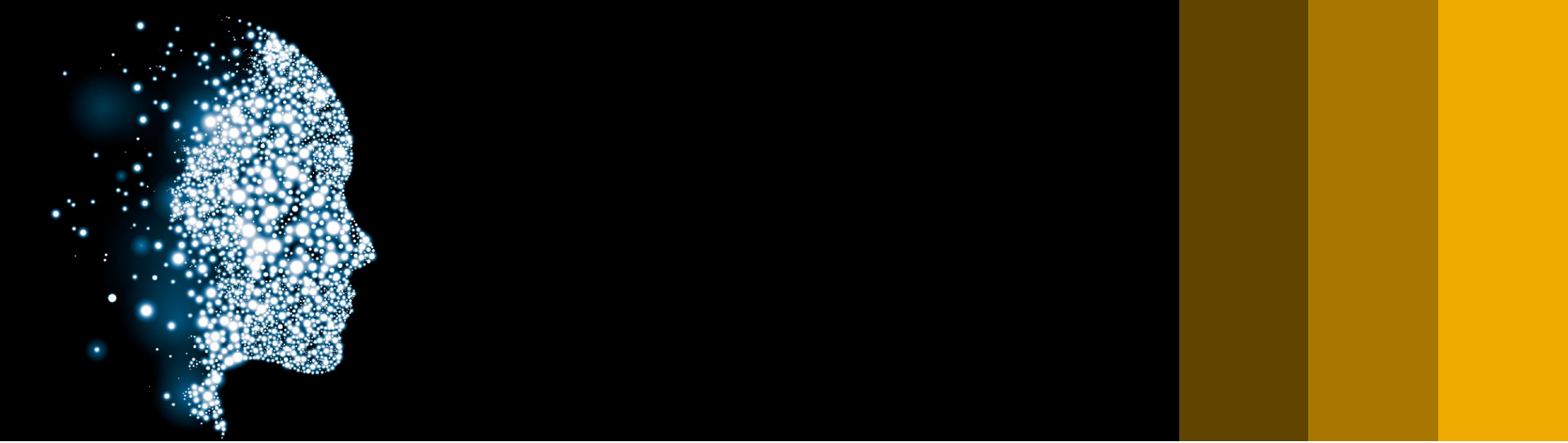
Original + Subtract Smile

Original + Add Bald, Add Male

Thank you.

Contact information:

open@sap.com



Week 6: Advanced Deep Learning Topics

Unit 2: Reinforcement Learning

SAP Run Simple

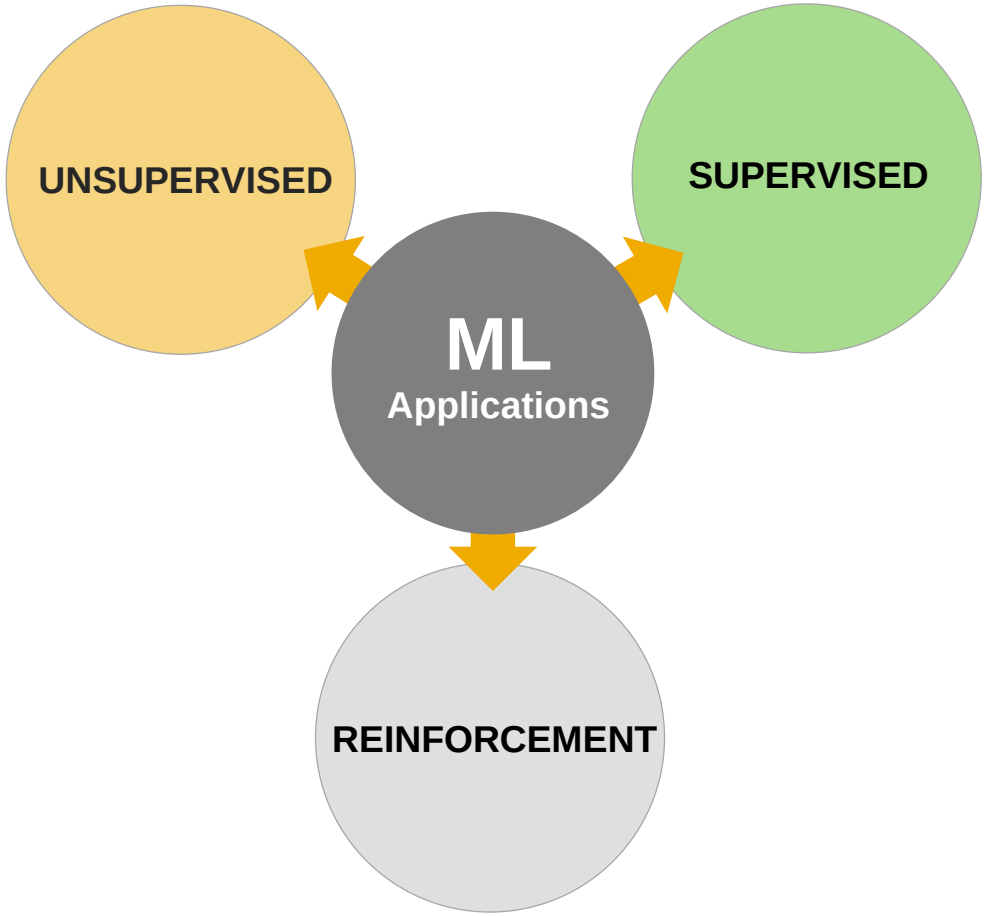
openSAP
open.sap.com

Reinforcement Learning

Applications of machine learning

ML applications fall into three broad contexts

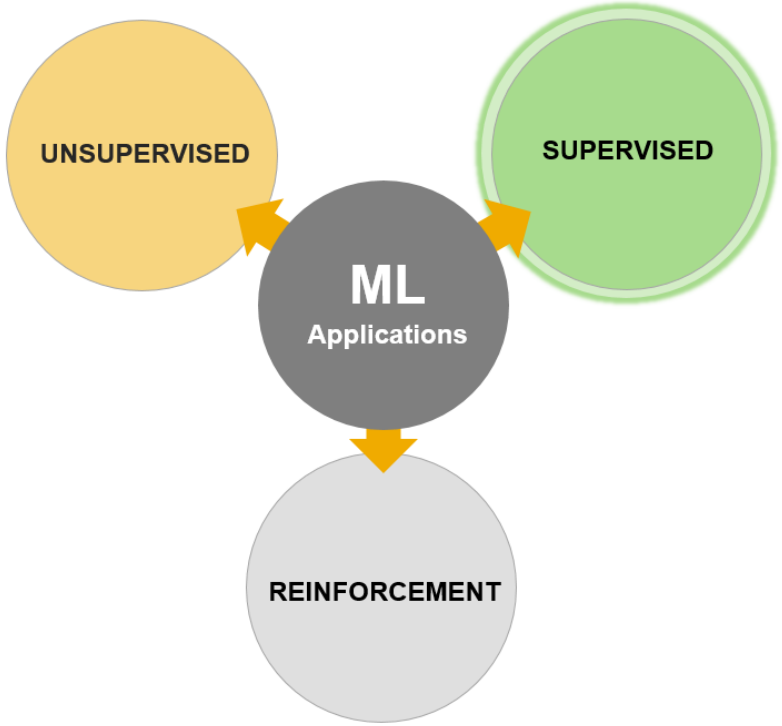
- Supervised learning
  - Dataset + labels/annotations
- Unsupervised learning
  - Dataset (without labels/annotations)
- Reinforcement learning
  - No initial dataset
    - Dataset accumulated with experience
    - ML agents interact with environment (trial and error)

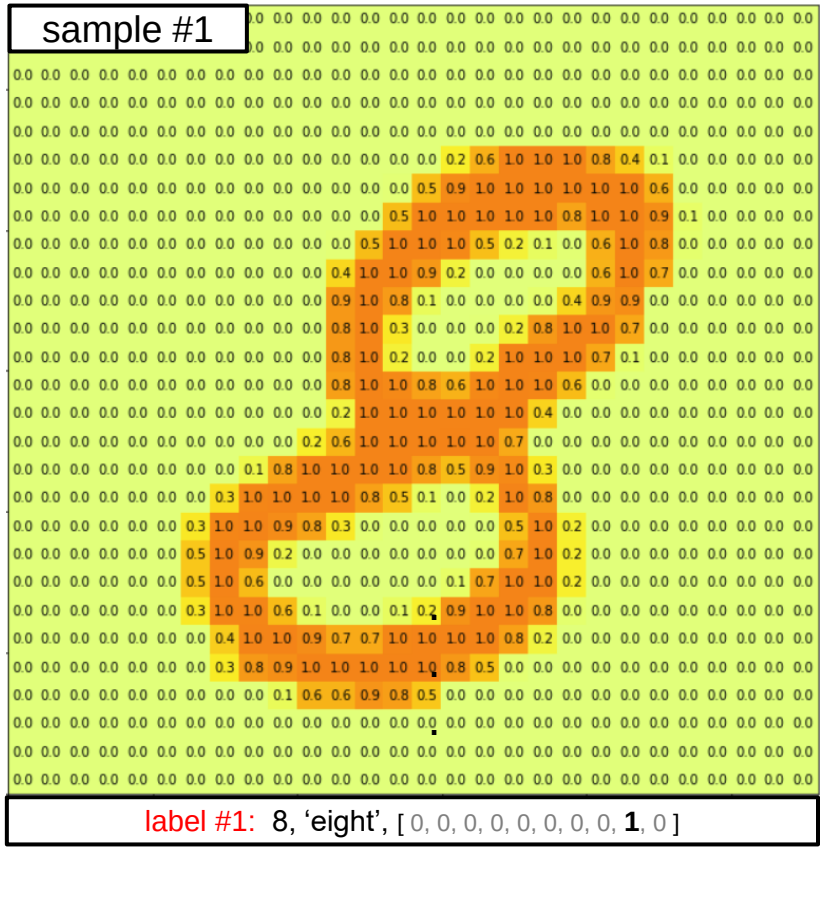


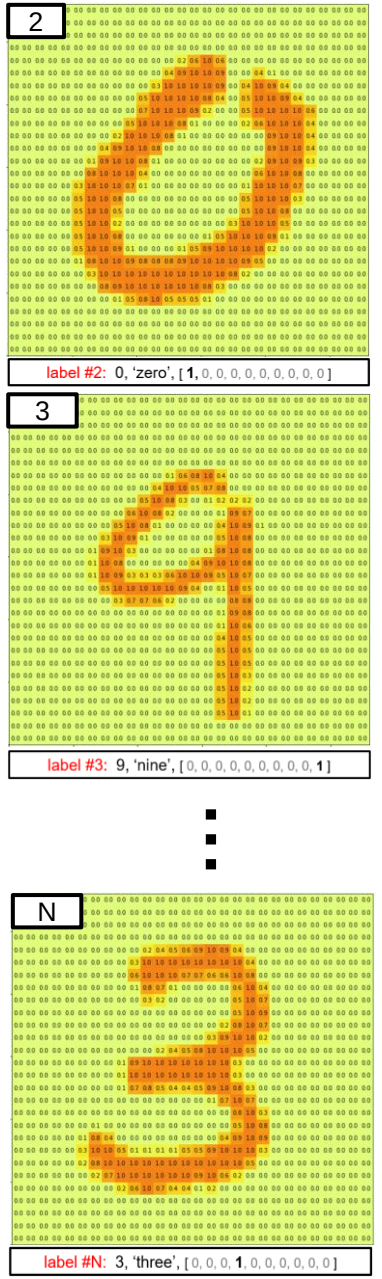
Reinforcement Learning

Machine learning – Supervised learning

- ~100,000s of examples
  - – Each example consists of
    - Data [Vector]
    - Label(s)



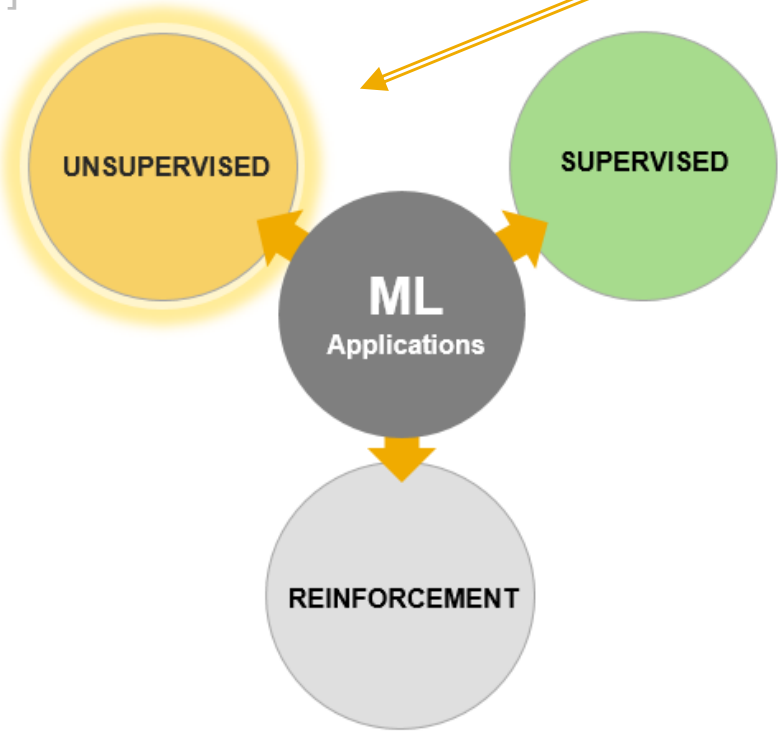




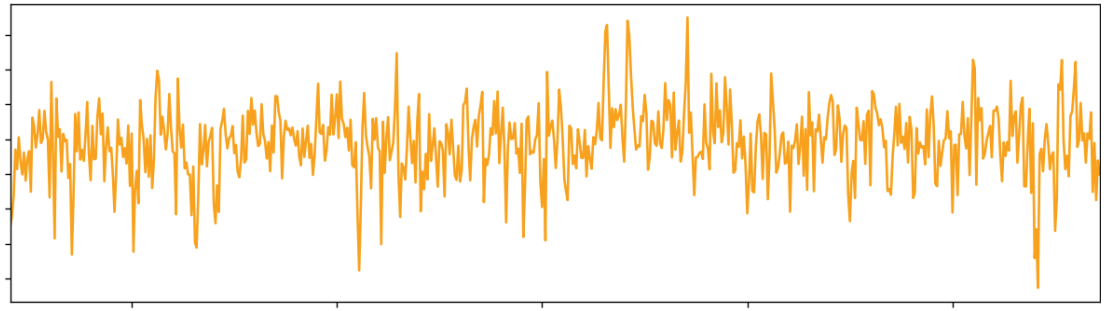
Reinforcement Learning

Machine learning – Unsupervised learning

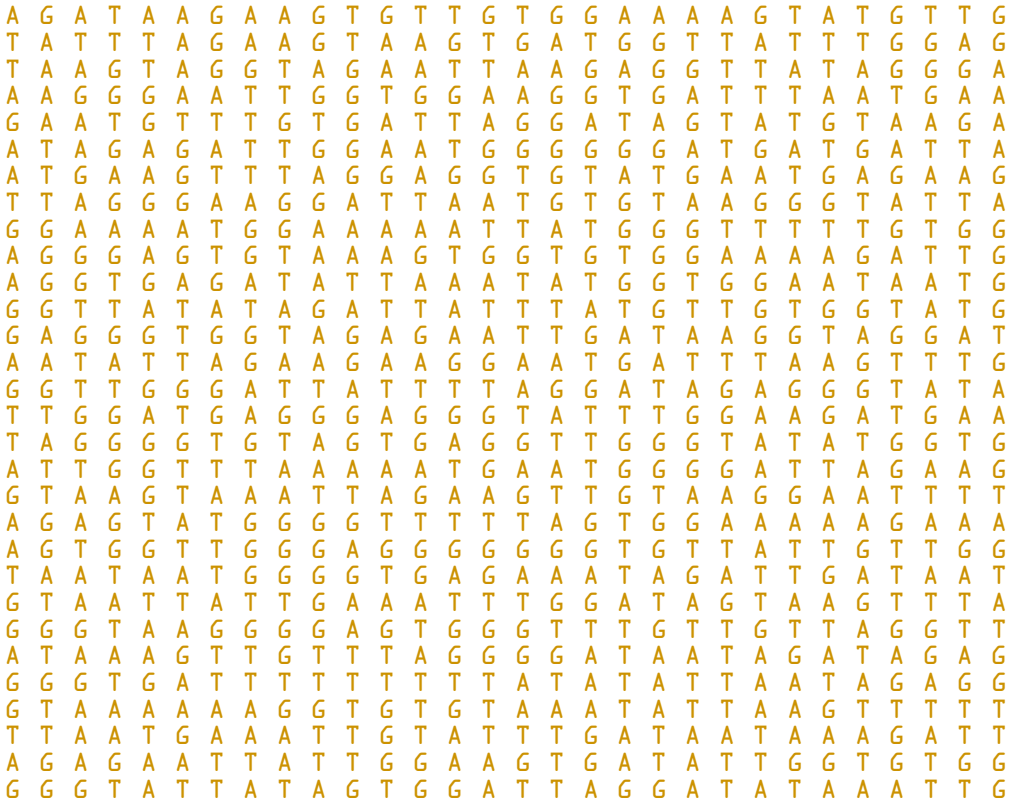
- In unsupervised learning, no labels are available
  - Typically larger datasets (millions)
  - Each example consists of
    - Data [Vector]
    - No Labels



Time Series Data



Gene Sequence Data

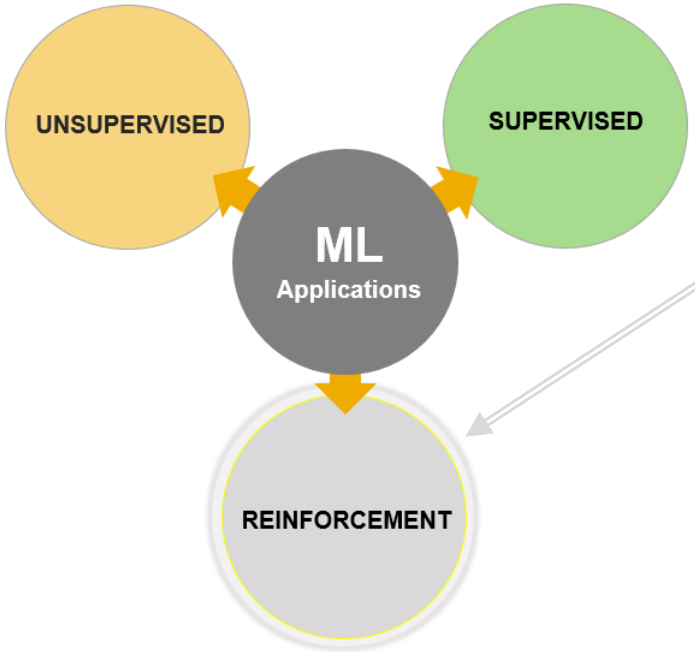


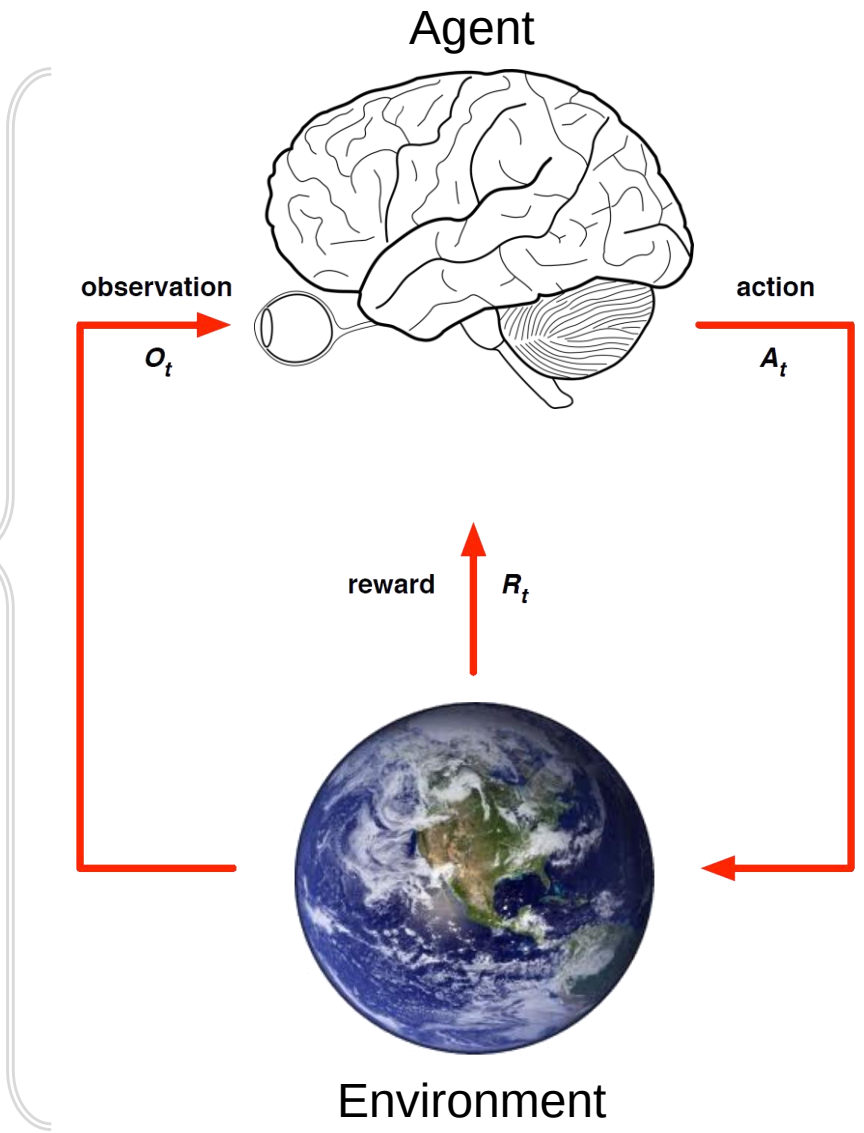
Reinforcement Learning

Machine Learning – Reinforcement Learning

RL: Dataset built with experience

- Experience = Env. State + Action + Next Env. State + Reward





Reinforcement Learning

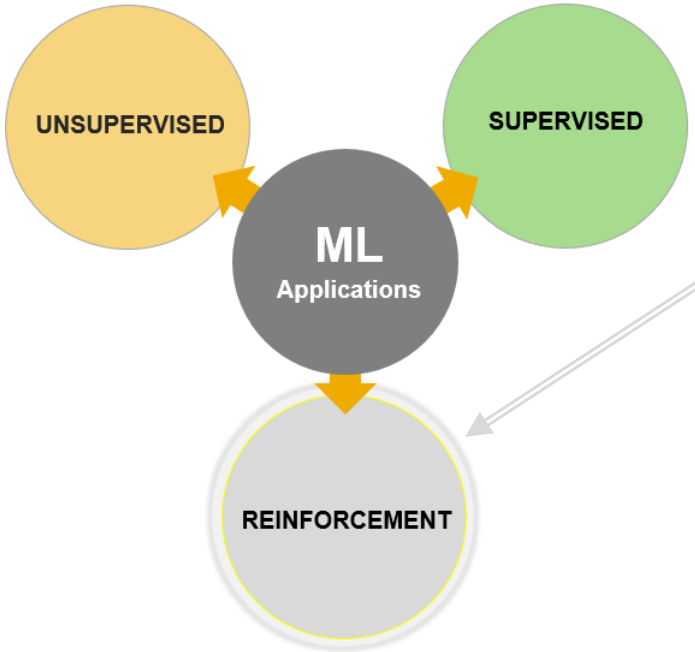
Machine Learning – Reinforcement Learning

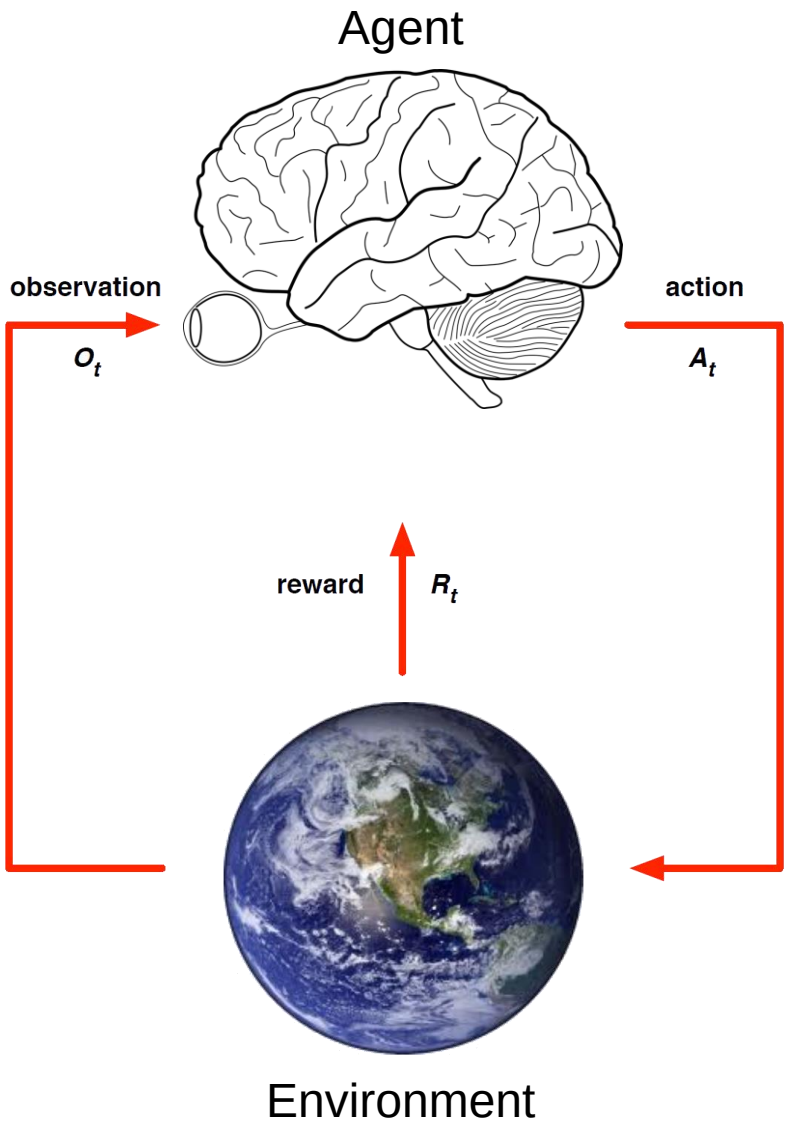
RL: Dataset built with experience

- Experience = Env. State + Action + Next Env. State + Reward

RL Feedback Loop

- At each step the **agent**
  - Executes action: A_t
  - Receives observation: O_t
  - Receives reward: R_t
- The **environment**
  - Receives action: A_t
  - Emits observation: O_{t+1}
  - Emits reward: R_{t+1}



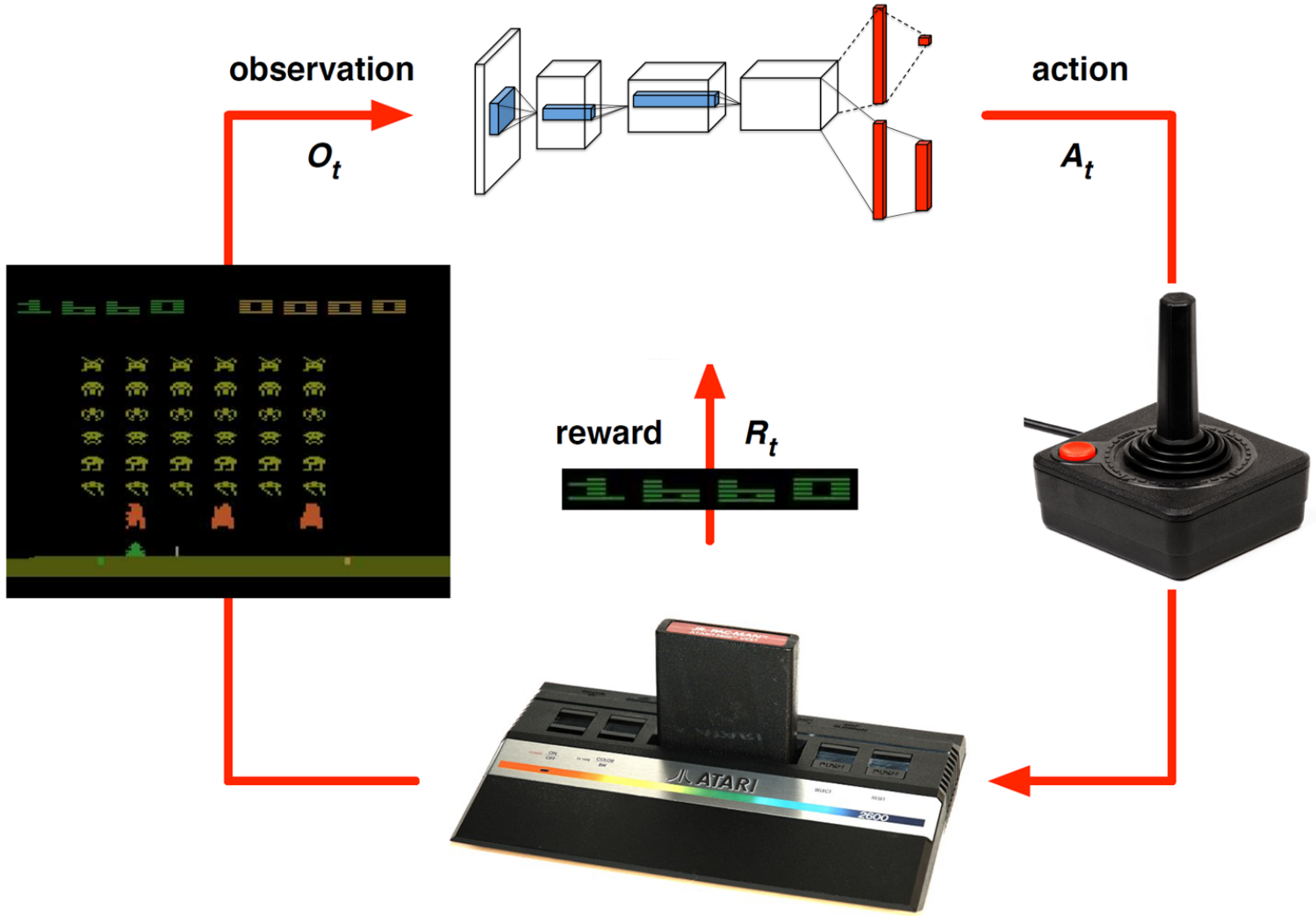


Reinforcement Learning

Applying RL to games

Atari Example

- Agent is a DL network
  - Interprets screen pixels
  - Outputs game action
  - Possible to use CNN+RNN
- Environment is Atari Emulator
  - Game AI
- Reward is game score

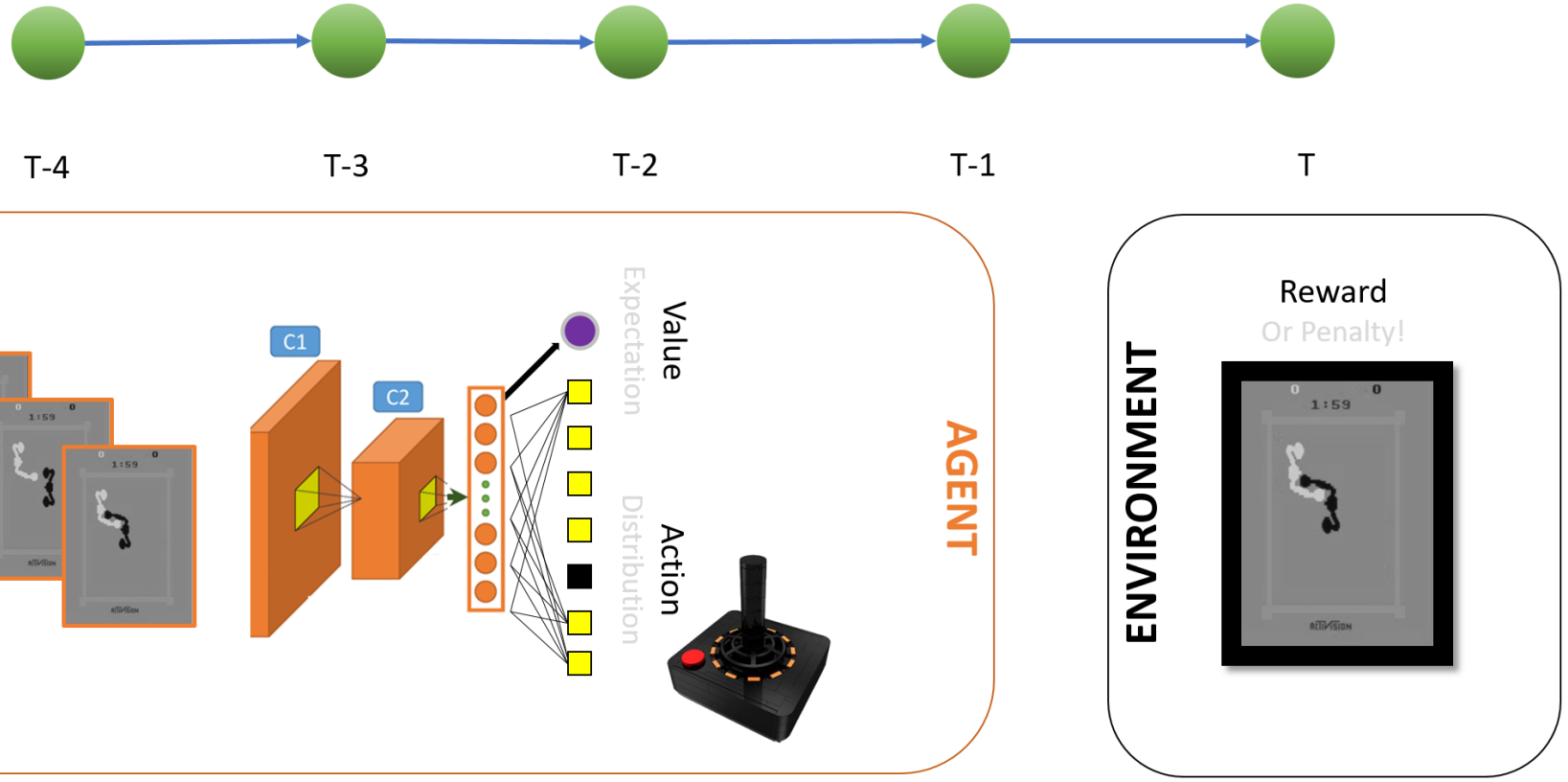


Reinforcement Learning

Application to Atari Boxing

Multiple frames make a single percept

- Enables network to learn dynamics

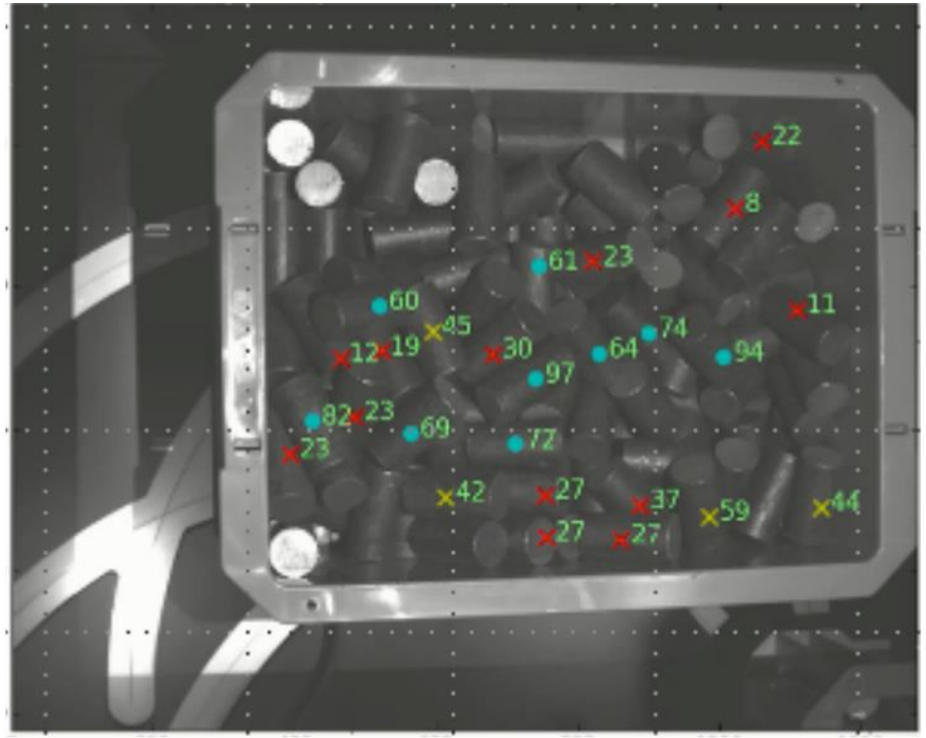


Reinforcement Learning

Application to bin picking

Industrial robot reaches human level performance in 8hrs

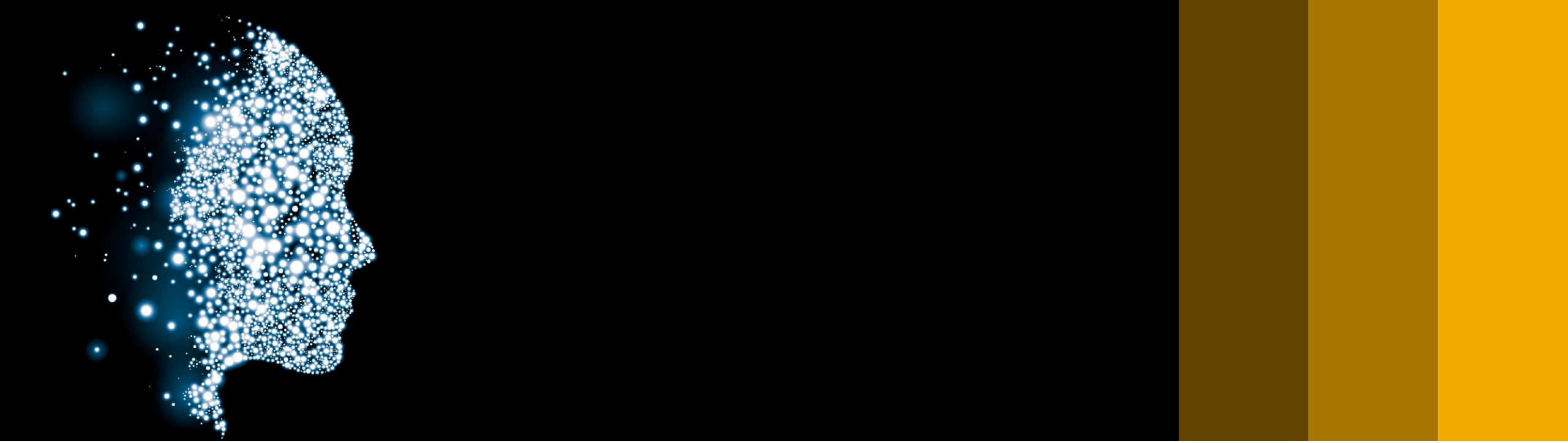
- Input is RGB+depth
- Output is probability map
- Reward is success of pick action



Thank you.

Contact information:

open@sap.com



Week 6: Advanced Deep Learning Topics

Unit 3: Unsupervised Learning

SAP Run Simple

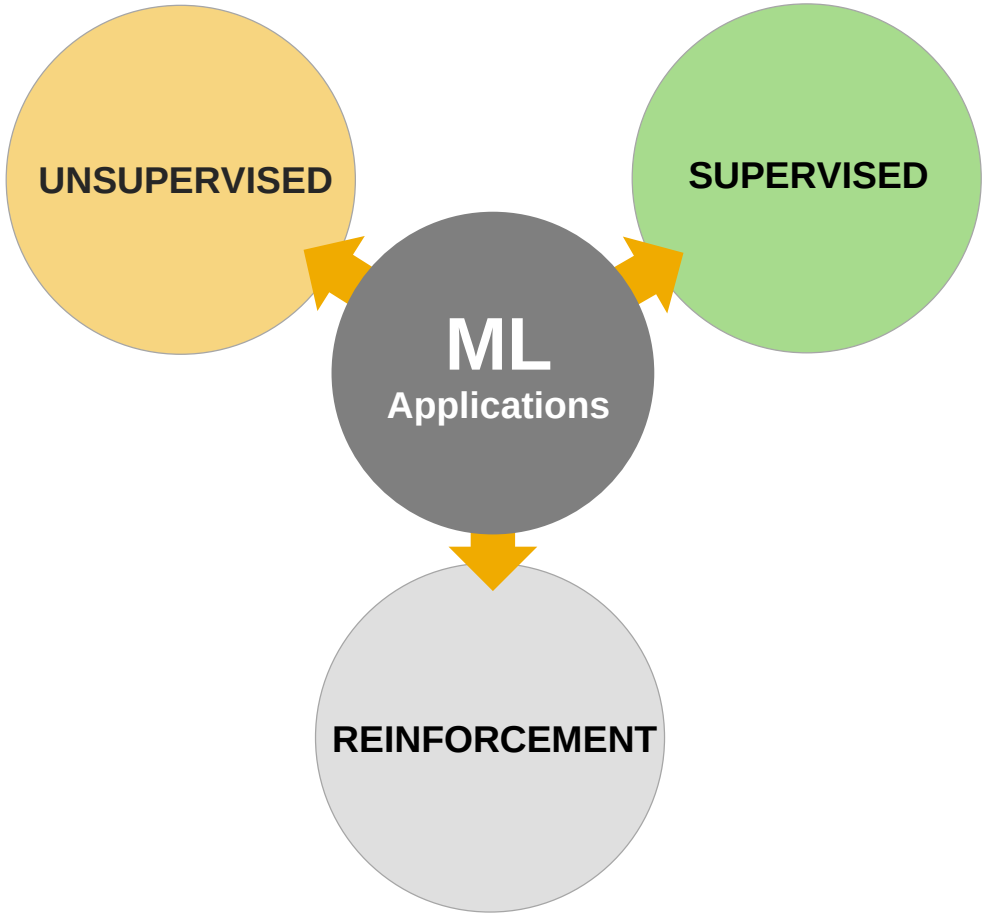
openSAP
open.sap.com

Unsupervised Learning

Applications of machine learning

ML applications fall into three broad contexts

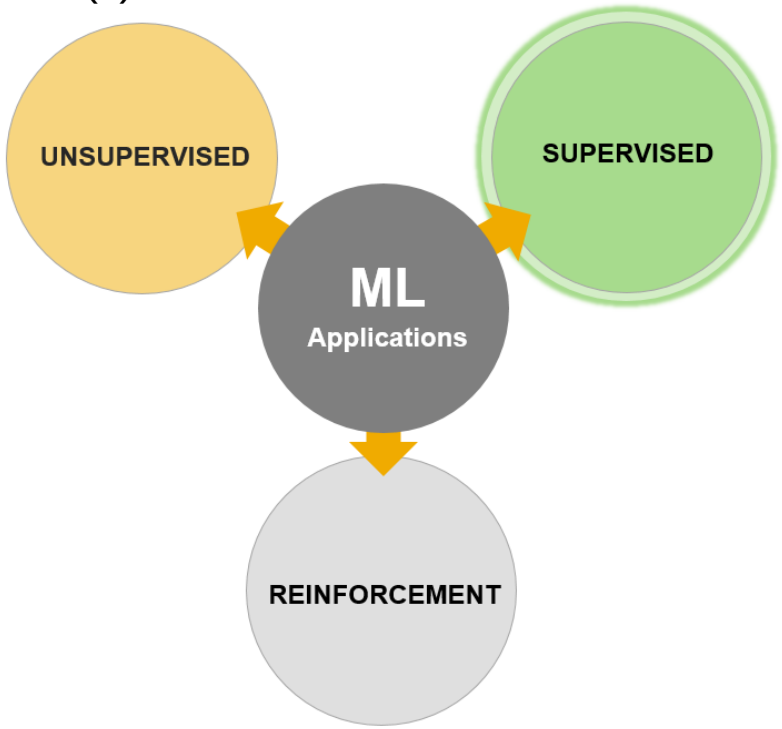
- Supervised learning
  - – Dataset + labels/annotations
- Unsupervised learning
  - – Dataset (without labels/annotations)
- Reinforcement learning
  - – No initial dataset
    - Dataset accumulated with experience
    - ML agents interact with environment (trial and error)

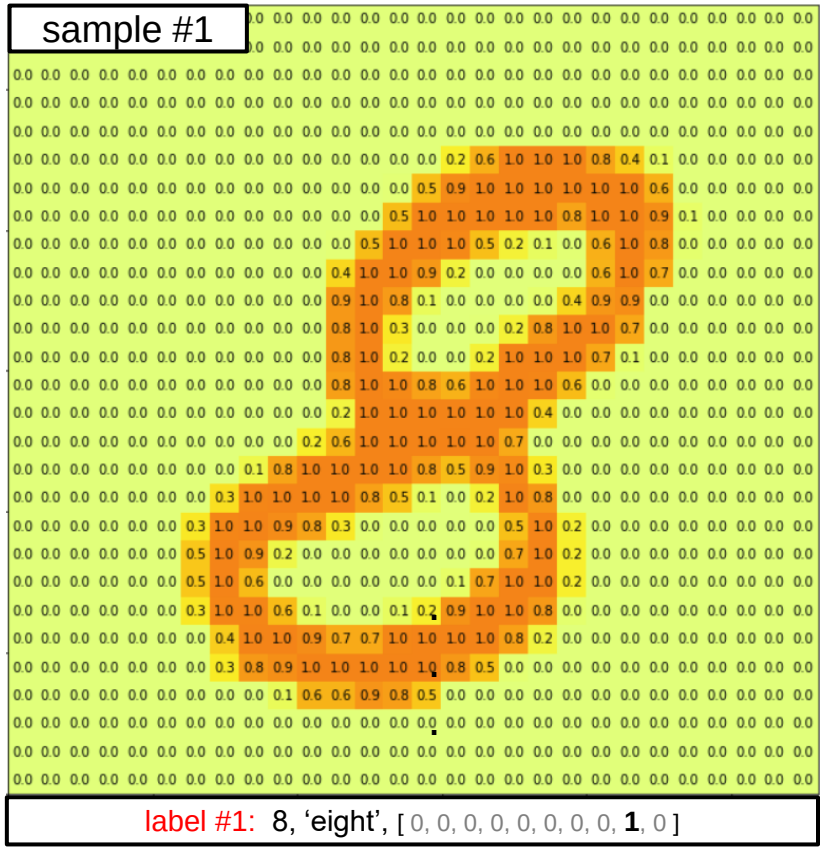


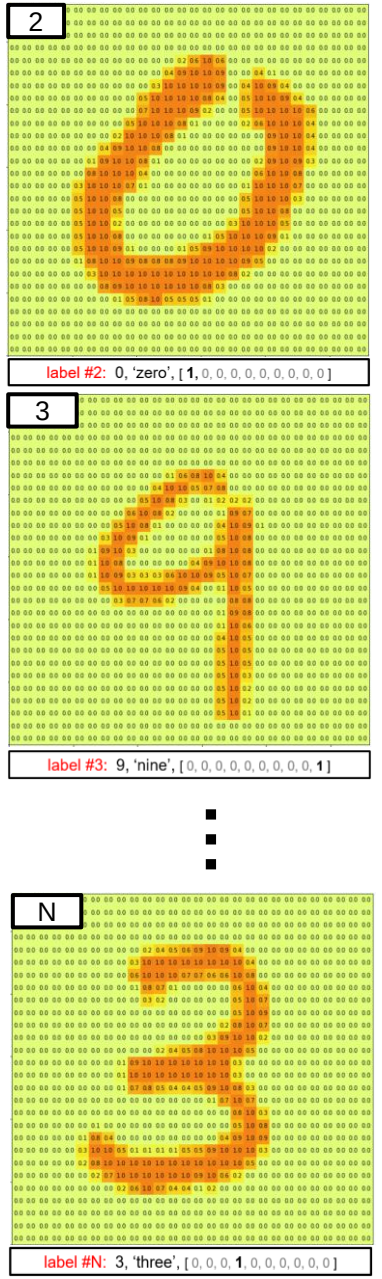
Unsupervised Learning

Machine learning – Supervised learning

- ~100,000s of examples
  - Each example consists of
    - Data [Vector]
    - Label(s)



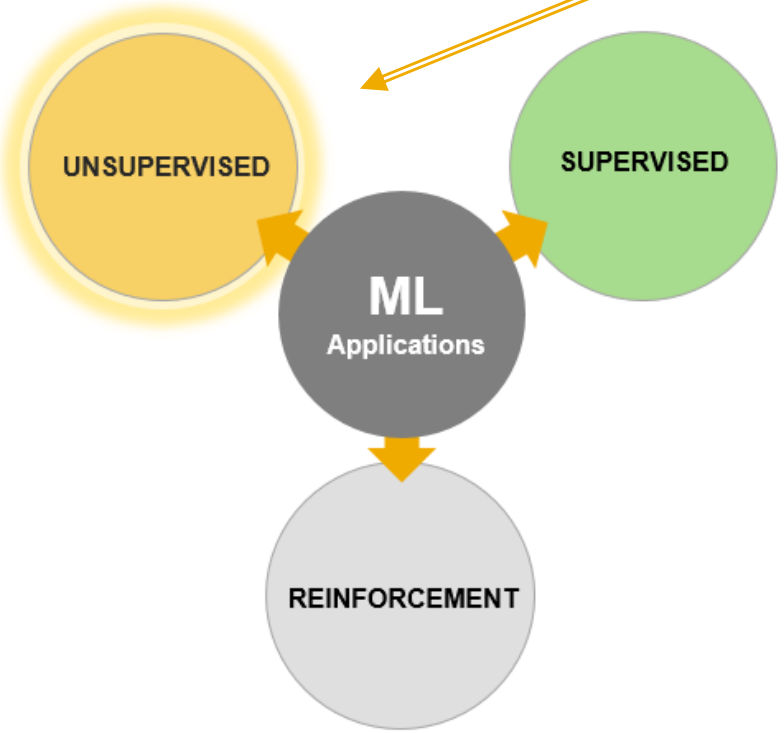




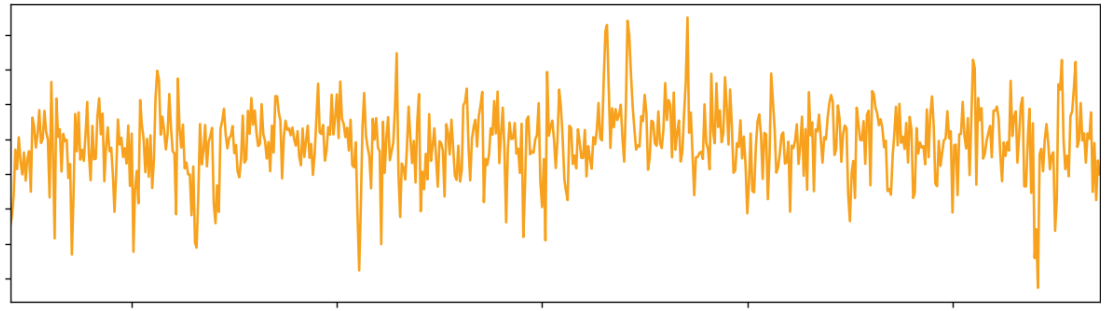
Unsupervised Learning

Machine learning – Unsupervised learning

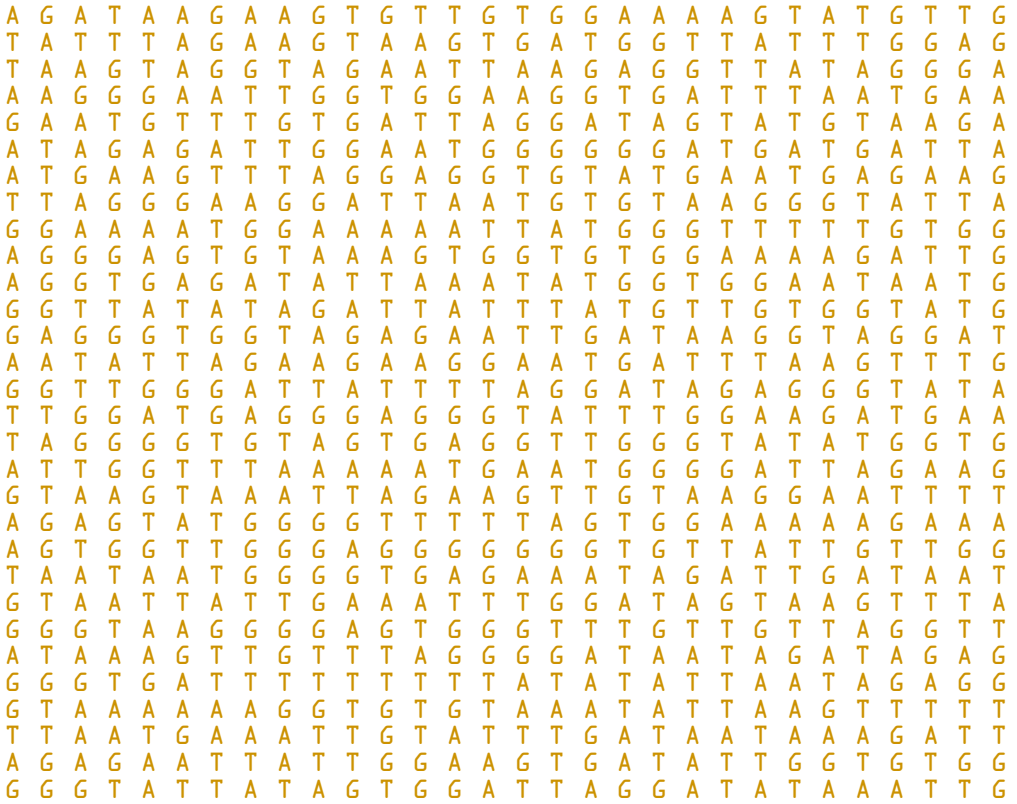
- In unsupervised learning, no labels are available
  - Typically larger datasets (millions)
  - Each example consists of
    - Data [Vector]
    - No Labels



Time Series Data



Gene Sequence Data

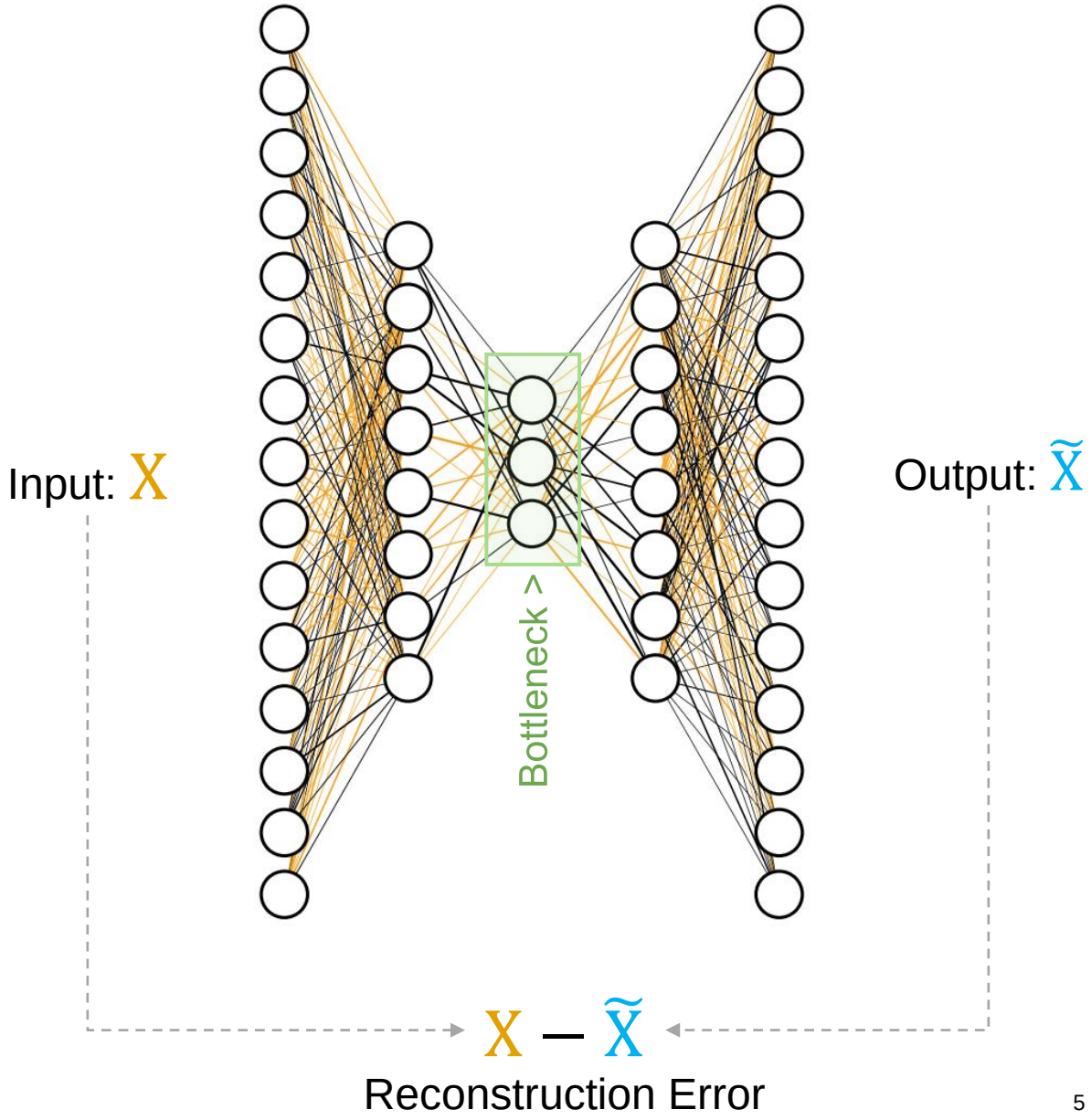


Unsupervised Learning

Anomaly detection using deep learning

Deep Autoencoder Network

- Input layer
  - Size of data vector
- Bottleneck layer
  - Summarized representation
    - ‘embedding’
- Output layer
  - Same dimensionality as input
- Reconstruction error
  - High errors indicate potential anomaly

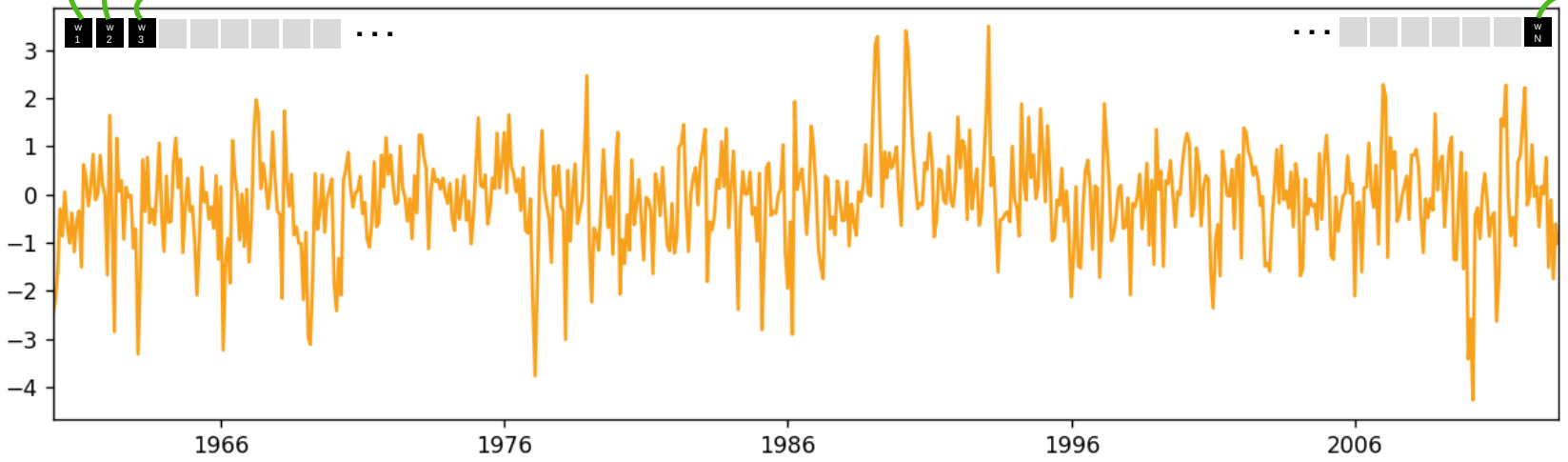


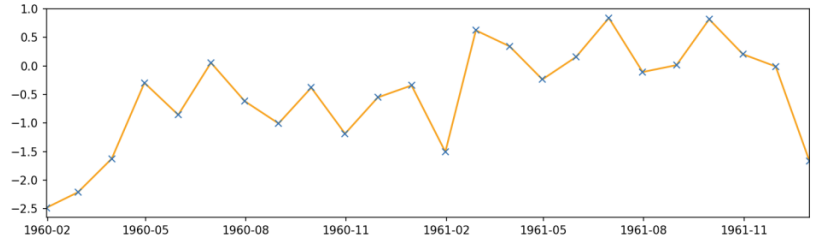
Unsupervised Learning

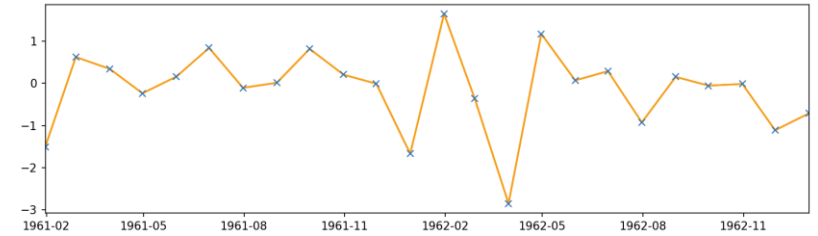
DL anomaly detection in time series

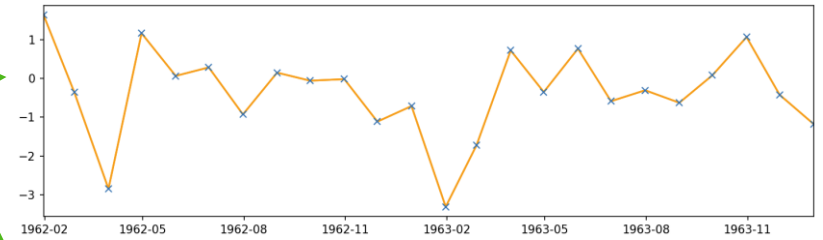
Time Series Signals

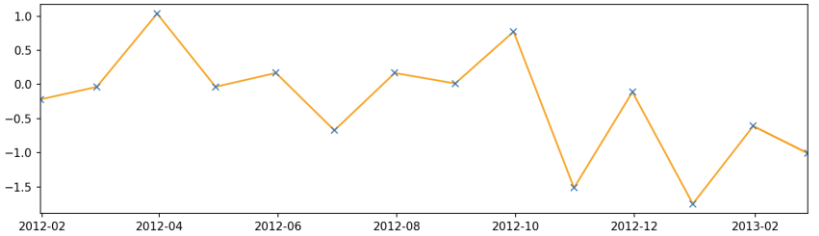
- Split into sliding windows [size is hyperparameter]
  - Normalization and preprocessing









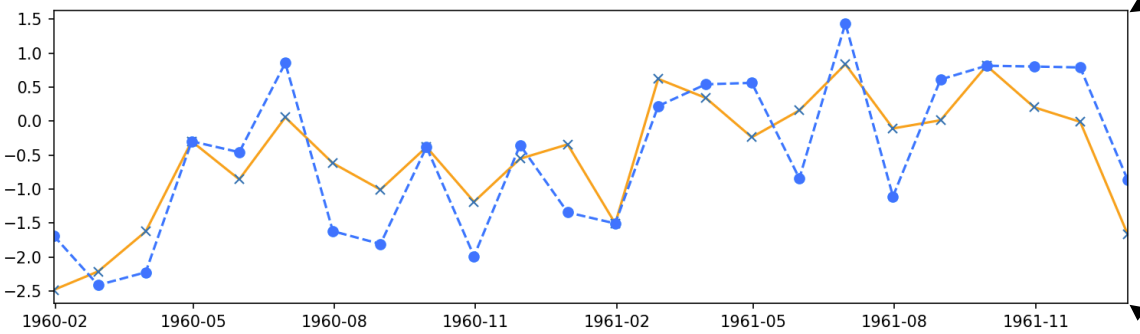


Unsupervised Learning

Detecting anomalies via reconstruction error

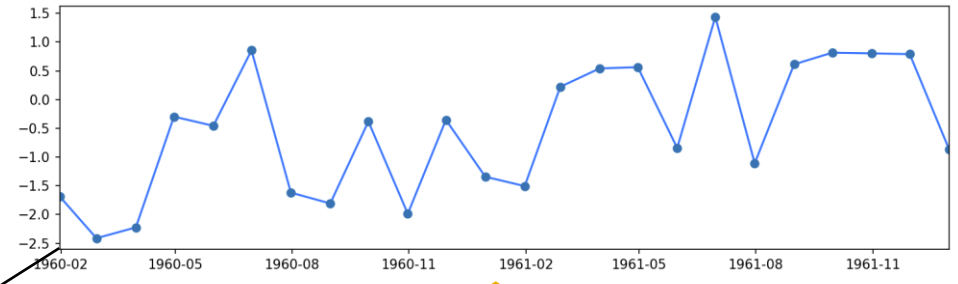
Reconstruction error (RE) as a proxy to outliers

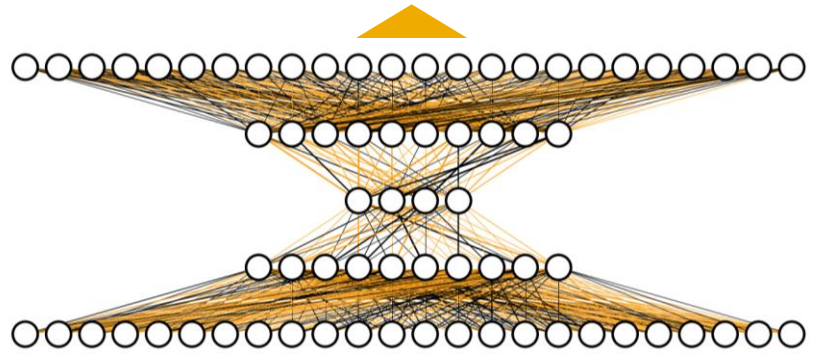
- Whenever RE is high, it indicates something
  - Threshold can be set using statistical bounds

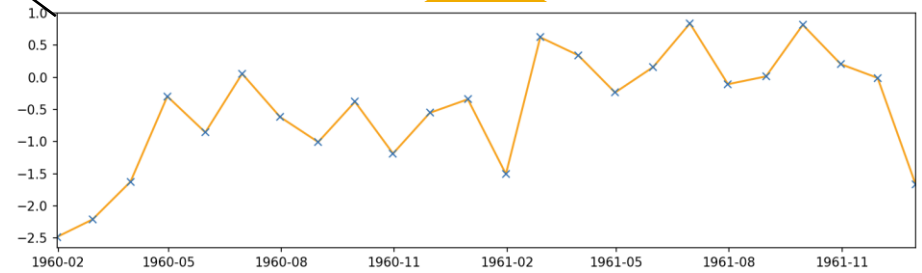


Reconstruction vs Input

Output (Reconstruction)







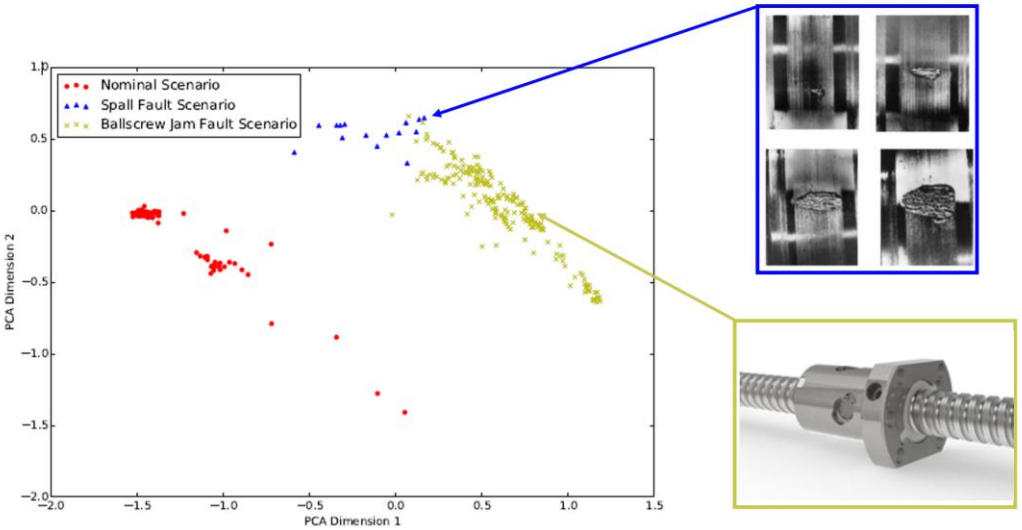
Input

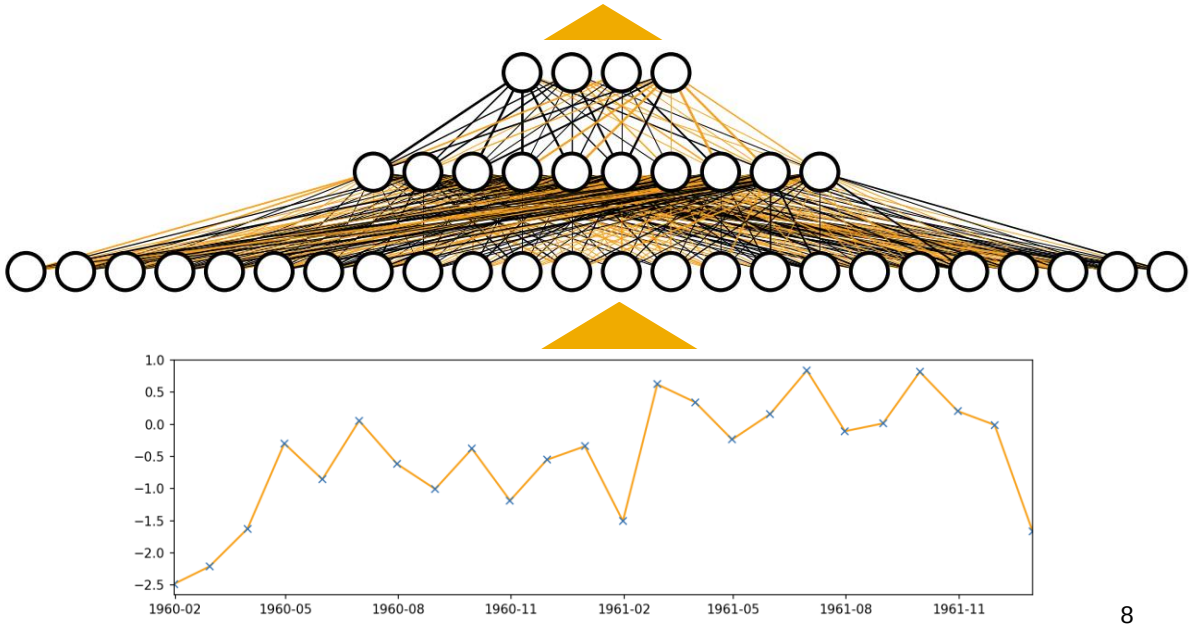
Unsupervised Learning

Interpreting anomalies

Projecting Bottleneck Activations to 2D/3D

- Maximize domain expert interpretation
  - Only reason about cluster centers in embedding space
- Future examples can be classified
  - Semi-supervised model

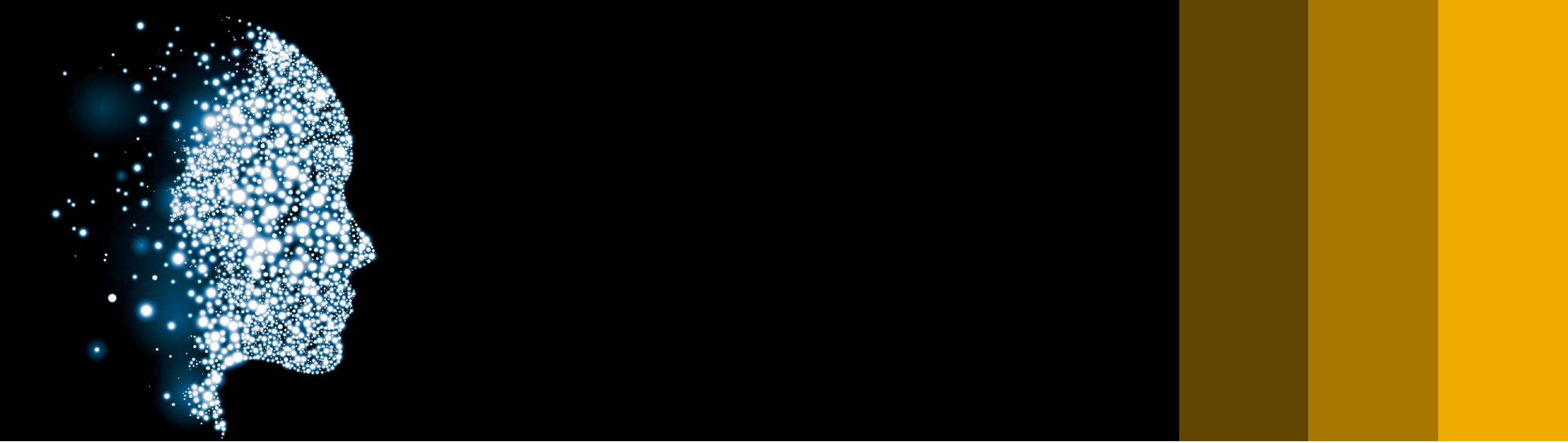




Thank you.

Contact information:

open@sap.com



Week 6: Advanced Deep Learning Topics

Unit 4: Deep Learning on Mobile

SAP Run Simple

openSAP
open.sap.com

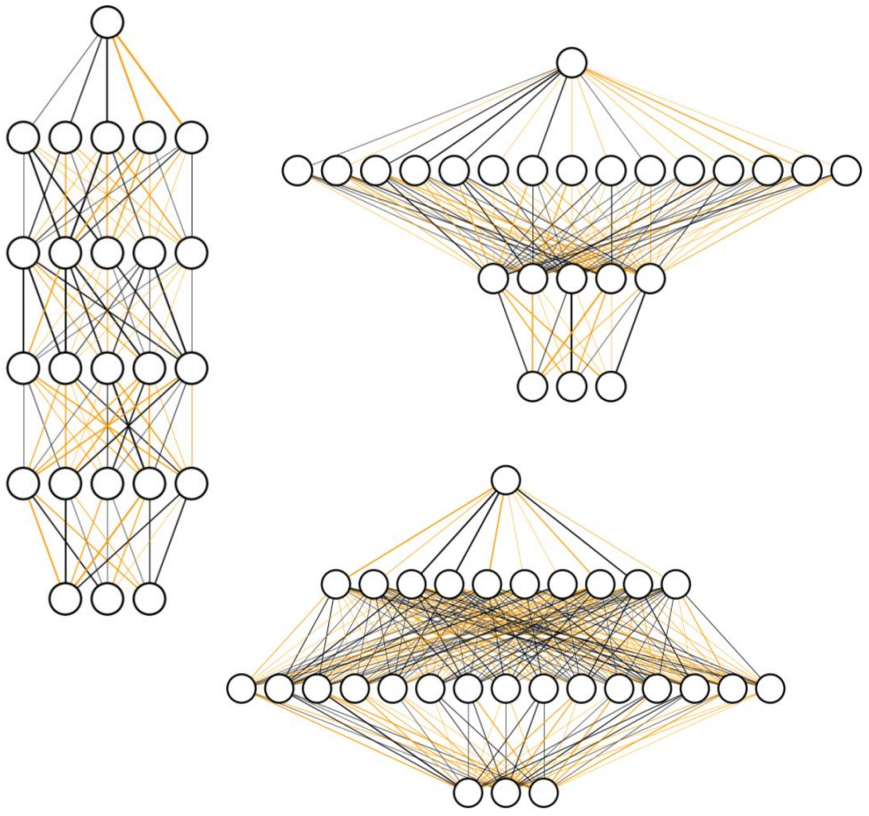
Deep Learning on Mobile

Getting to a trained model

Typical Deep Learning Workflow:

- Prepare dataset
  - Normalize, augment, handle missing data
- Search for best model architecture
  - Iteratively train and evaluate model variants
  - [High compute requirement – e.g. cloud + GPUs]
- Deploy
  - Introduce model into production system
  - [High throughput requirement – e.g. cloud + Hadoop cluster]
- Re-train with incoming data

Candidate Neural Networks

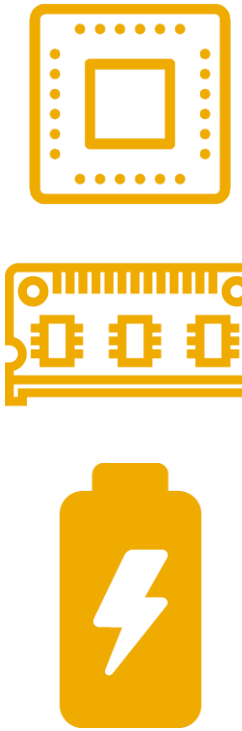


Deep Learning on Mobile

Deploying trained model on a mobile device

Mobile platforms present a unique challenge

- Although mobile CPUs are fast…
  – e.g. Quad core 1.9Ghz
- And have large memory…
  – e.g. 64GBs
- Battery efficiency is a bottleneck!
  – 20-44 hrs depending on mode & consumption

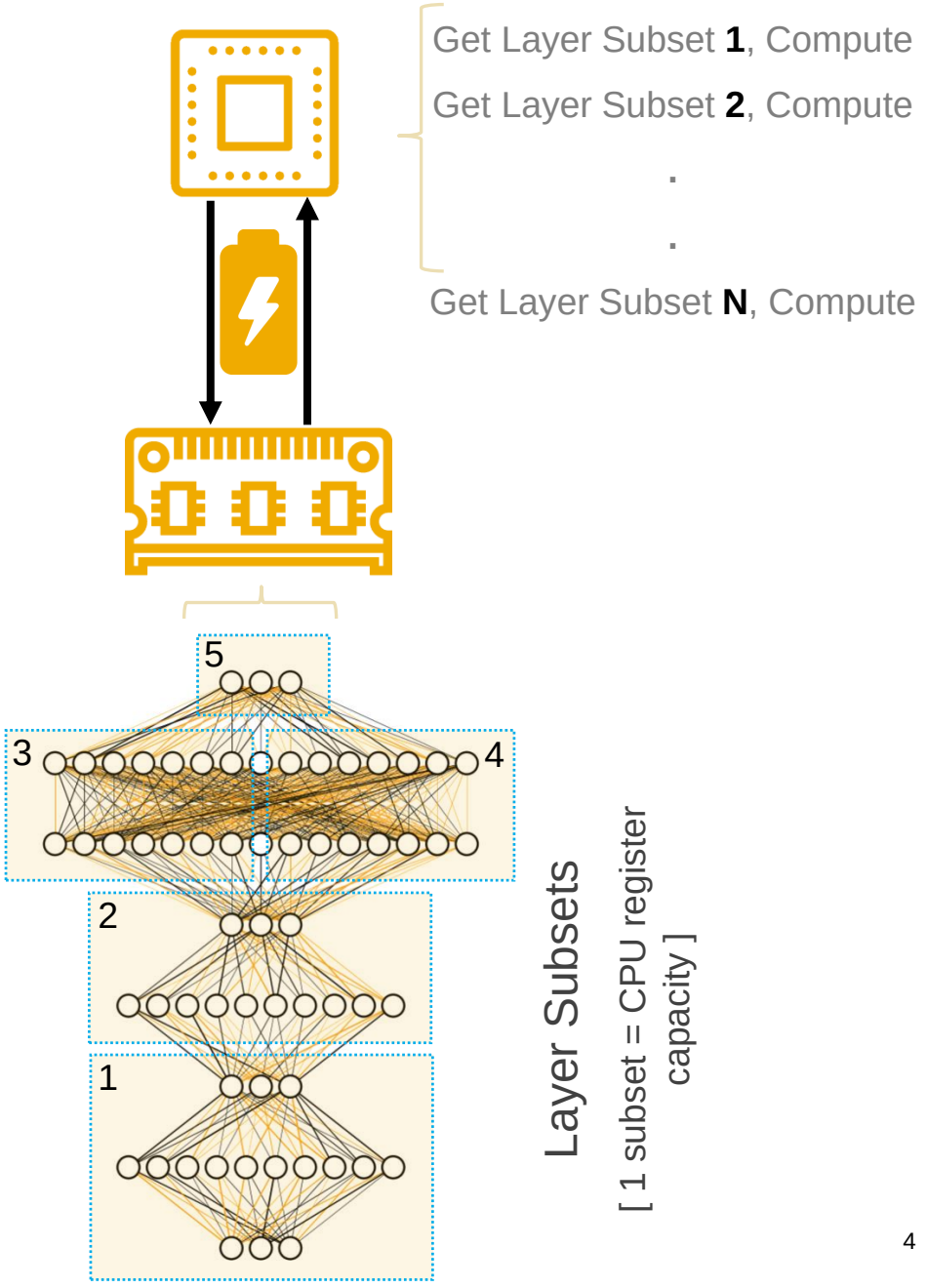


Deep Learning on Mobile

Deploying trained model on a mobile device

Large DL models require lots of data movement

- CPU can only compute a subset of the network at a time
  - # of weights & input/outputs overwhelms available registers
- Each network subset has to be loaded from memory
  - Large networks may require many subsets
- Each read from memory and transfer to CPU is costly in terms of energy
  - Many fetches may be needed for a single inference pass
  - Especially critical for applications that process streaming data

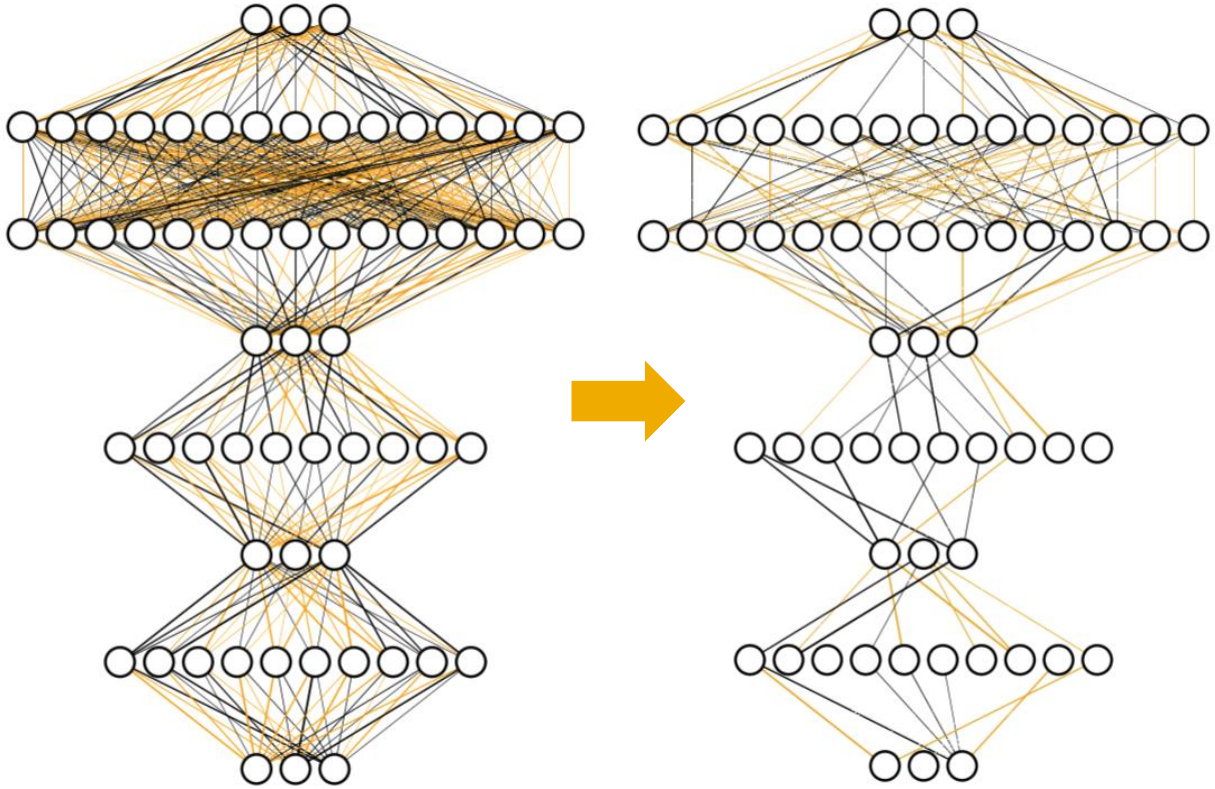


Deep Learning on Mobile

Optimization approach – Pruning

Weight Pruning

- Opportunistically remove weights
  - Can lead to significant parameter reduction
  - Requires a [full] model to be initially trained
  - Inspired by synaptic pruning in biological neurons
- Continue streaming training data
  - Monitor performance loss due to reduced weights
- Target layers with highest battery impact
  - Requires energy estimation model
    - Track load, multiply, and accumulate calls

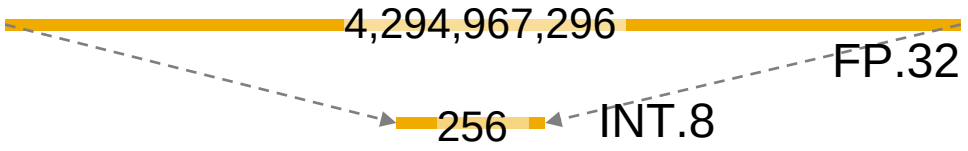


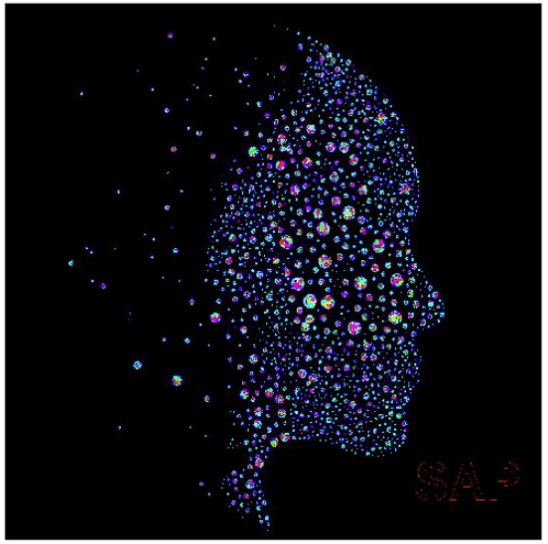
Deep Learning on Mobile

Optimization approach – Quantization

Weights & activations can be expressed using fewer bits

- Quantization leads to a significant compression in memory
  - Up to 4x reduction for each parameter
  - Especially critical for very large networks
- Typically, quantization leads to minimal accuracy loss
  - Note: weights & activations still require full precision in training
- Finding best quantization mapping is hard (and lossy)
  - Tools like TensorRT automate this process
- Preprocessing and postprocessing steps need to be added
  - Inputs need to be quantized
  - Outputs need to be de-quantized





Deep Learning on Mobile

Steps to mobile deployment

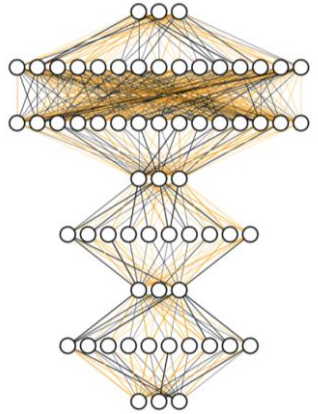
DL Model Mobile Optimization Workflow

- High level model definition
  - e.g. Python code
- Intermediate representation
  - Compilation and optimization
  - Target-independent
- Target-dependent optimizations
  - Code generation
- Runtime
  - Packaged binary

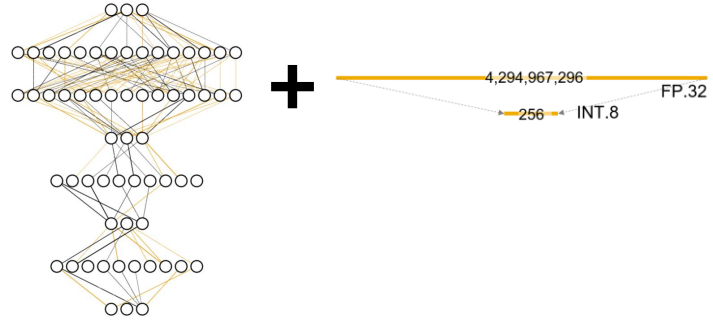
Model Definition [high-level language]

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
import keras

model = Sequential()
model.add( Dense( 10, input_dim = 3 ))
model.add( Dense( 3, activation = 'sigmoid' ))
model.add( Dense( 10, activation = 'sigmoid' ))
model.add( Dense( 3, activation = 'sigmoid' ))
model.add( Dense( 15, activation = 'sigmoid' ))
model.add( Dense( 3, activation = 'sigmoid' ))
model.add( Dense( 3, activation = 'linear' ))

model.compile(optimizer='adam', loss='mse')
```

Compute Graph [intermediate representation]



Optimizations [pruning + quantization]



Runtime

Deep Learning on Mobile

Available tools

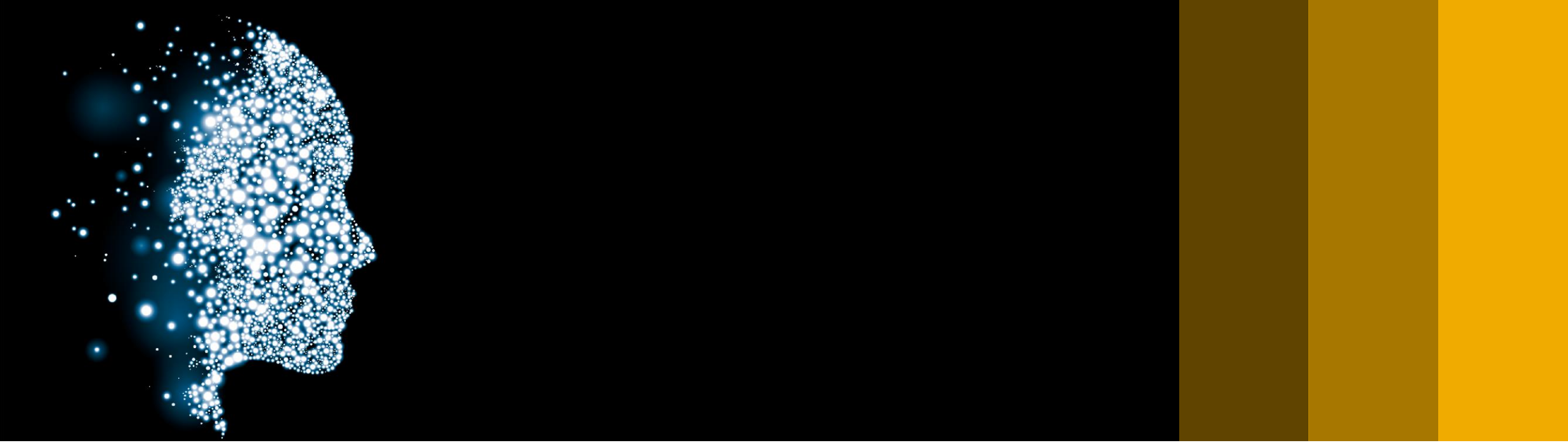
Mobile Optimization Platforms

- TensorRT [Real Time] – NVIDIA
  - https://developer.nvidia.com/tensorrt
- TensorFlow XLA – Google
  - https://www.tensorflow.org/performance/xla/
- Core ML – Apple
  - https://developer.apple.com/machine-learning/
- NVML – UW & Amazon
  - http://www.tvmlang.org/2017/10/06/nnvm-compiler-announcement.html

Thank you.

Contact information:

open@sap.com



Week 6: Advanced Deep Learning Topics

Unit 5: Summary, Recap, and Further Resources

Summary, Recap, and Further Resources

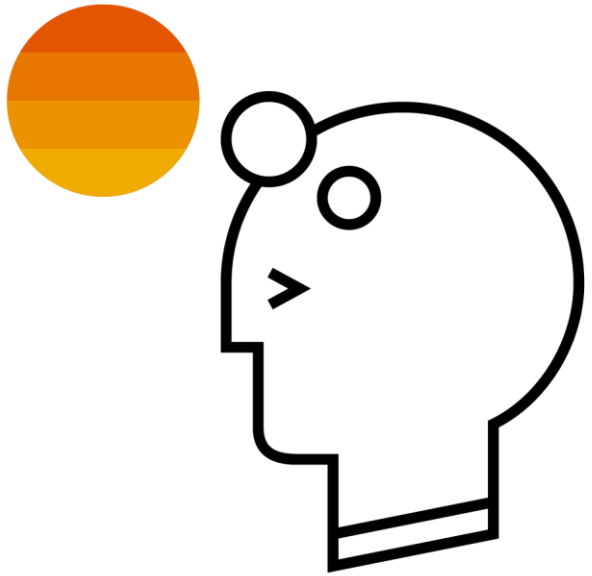
Summary and recap

- Getting Started with Deep Learning
- Building TensorFlow Applications
- Deep Networks and Sequence Models
- Convolutional Networks
- Industry Applications of Deep Learning
- Advanced Deep Learning Topics

Summary, Recap, and Further Resources

Key Messages

- Deep learning is real-world relevant today
- Deep learning is an engineering skill for practical problems
- You have first hands-on experience with industry use cases
- Practice makes perfect!



Summary, Recap, and Further Resources

Selected resources

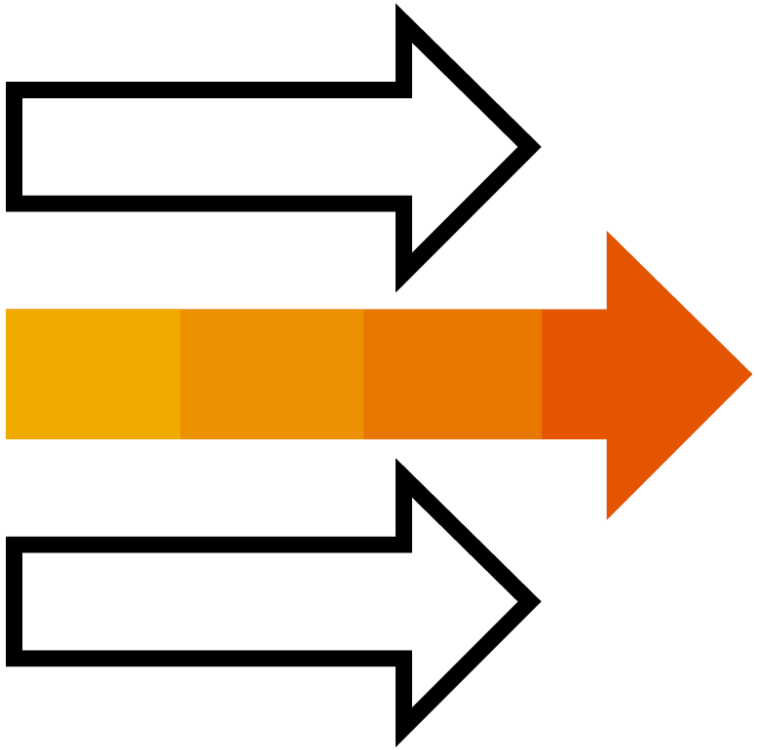
deeplearning.ai specialization at Coursera (MooC)

CS231n, CS224n at Stanford University
(image and text processing)

www.deeplearningbook.org (theoretical foundations)

SAP Leonardo Machine Learning microsite:
sap.com/ml

openSAP Leonardo Machine Learning course
(forthcoming)



Thank you.

Contact information:

open@sap.com