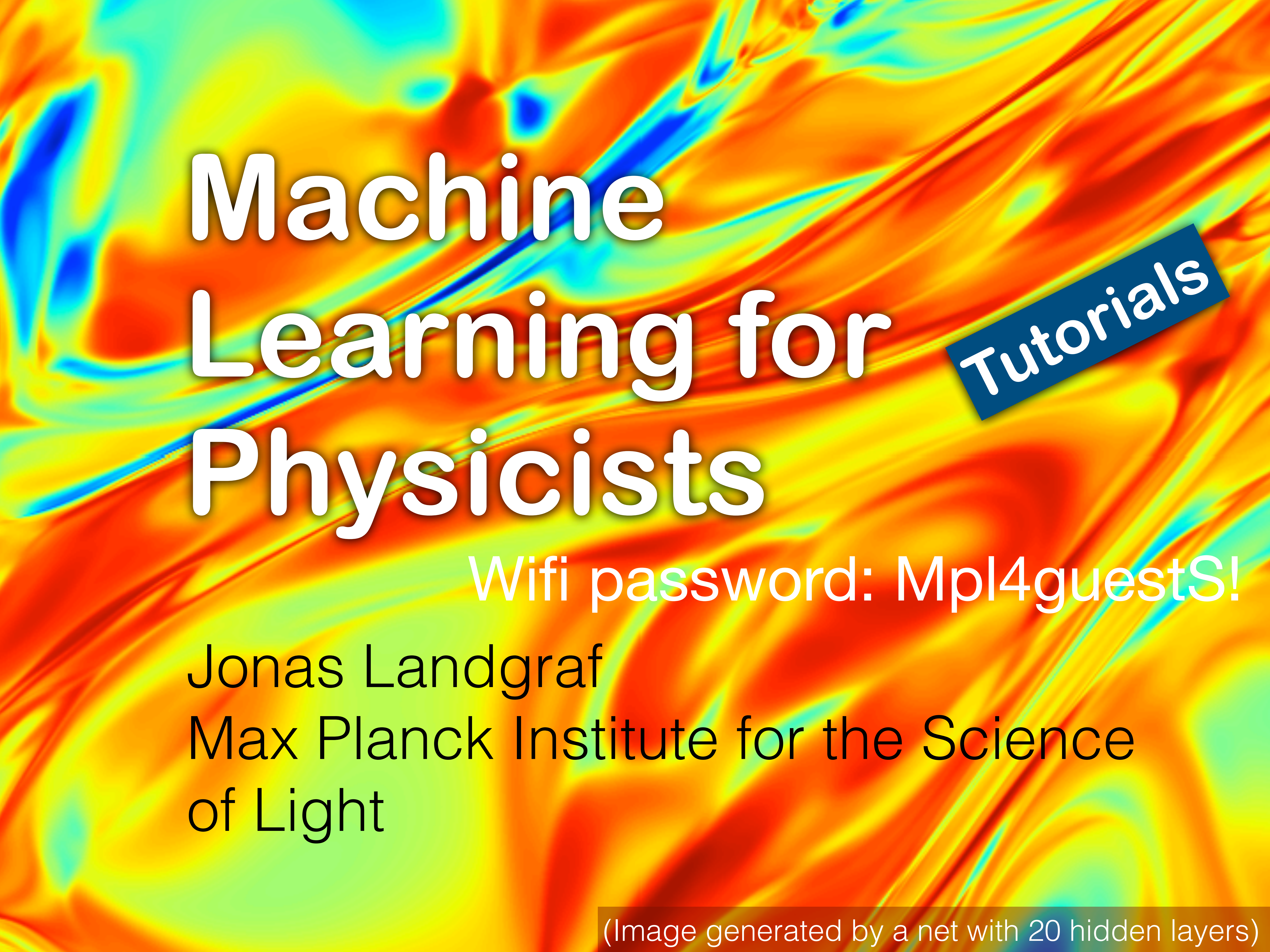
Machine Learning for Physicists

Tutorials

Wifi password: Mpl4guestS!

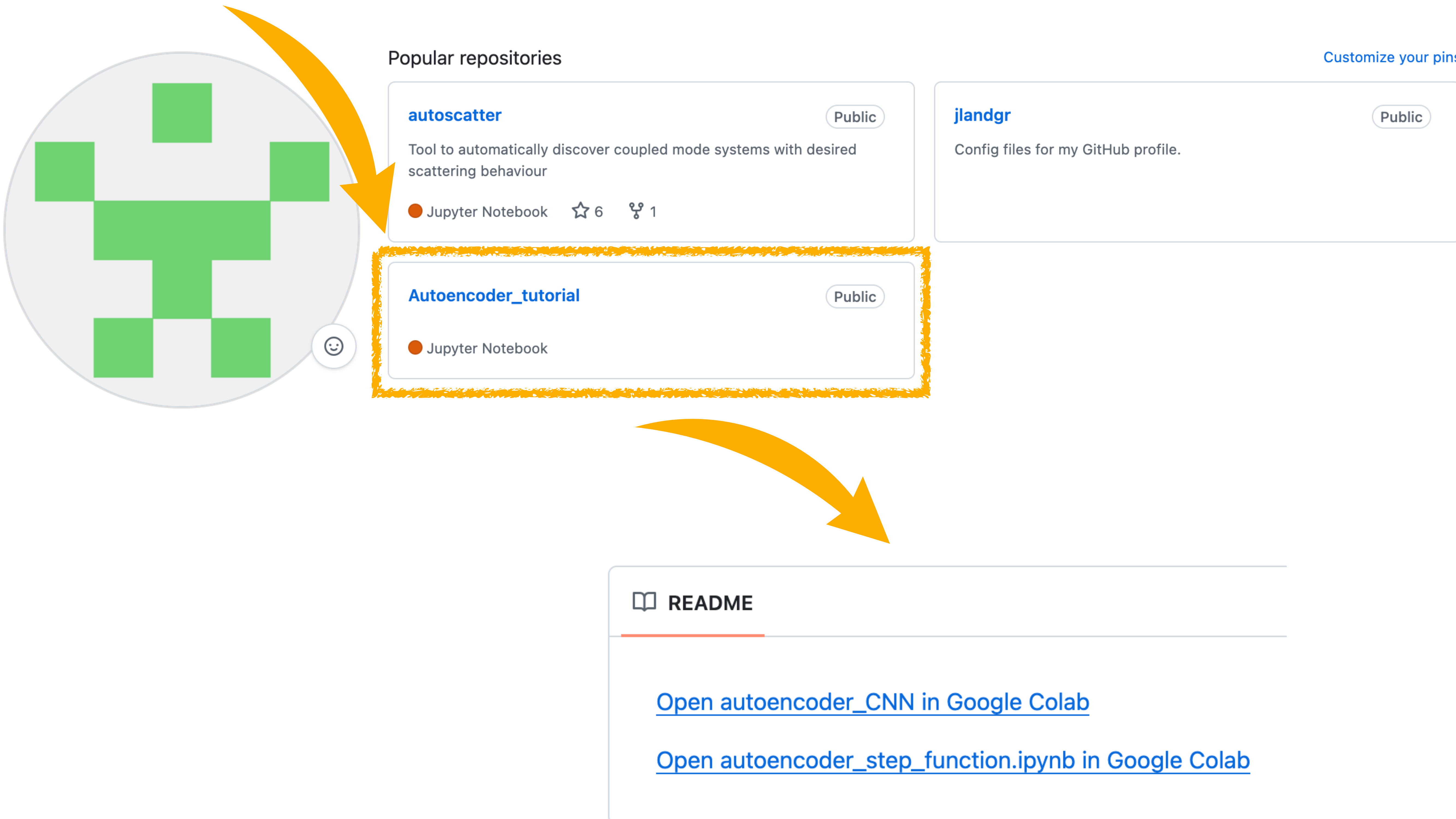
Jonas Landgraf
Max Planck Institute for the Science of Light



(Image generated by a net with 20 hidden layers)

Practice session: Autoencoders

see: github.com/jlandgr

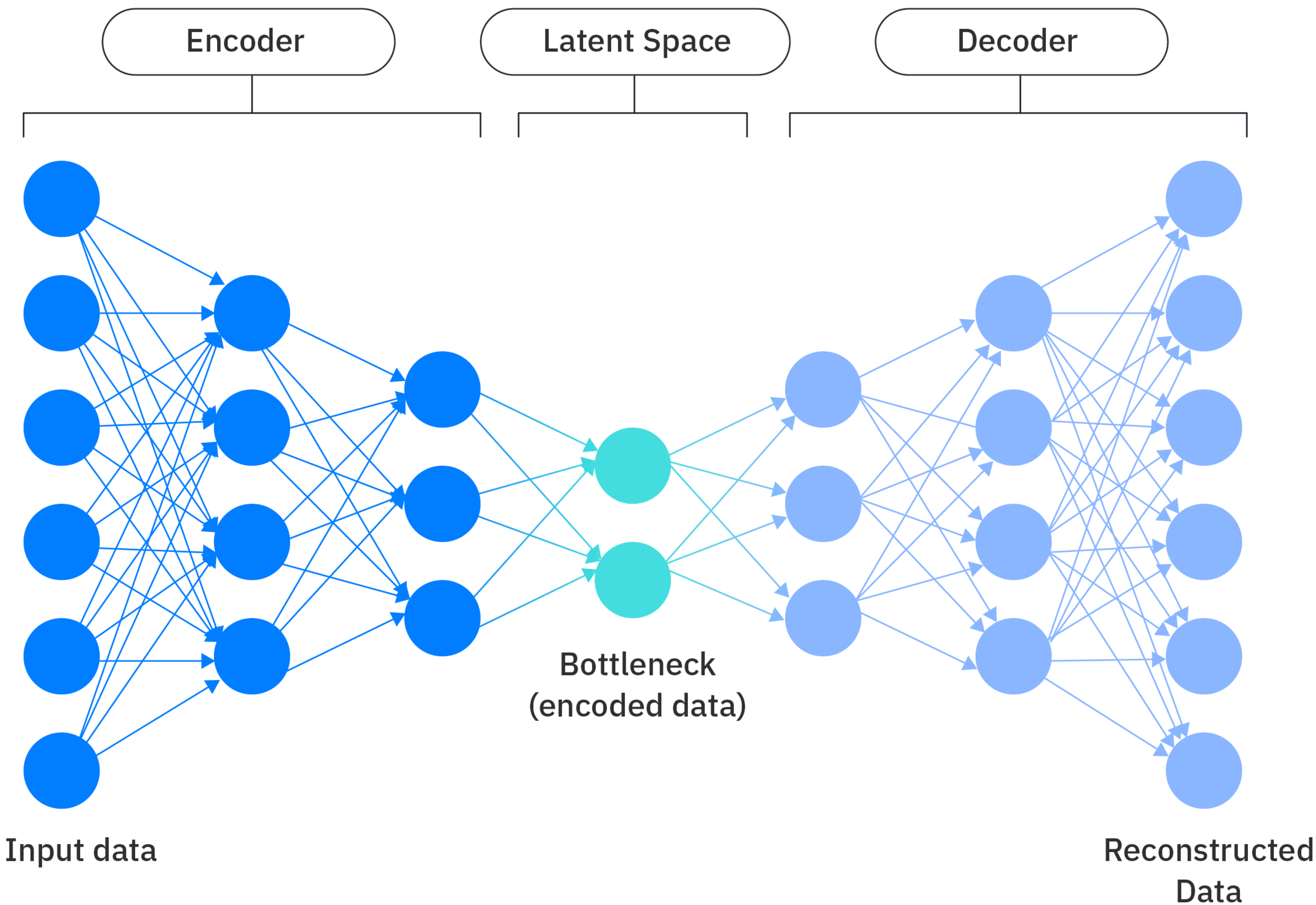


Alternative: Download the notebook and run it lokally on your laptop

Reminder: What is an autoencoder?

Main idea:

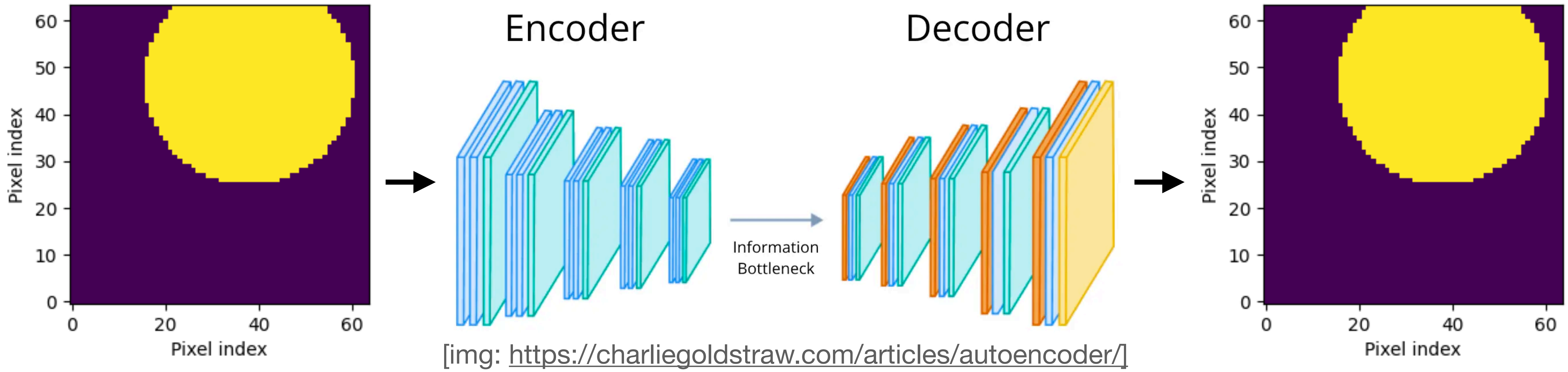
An autoencoder replicates its input data, so $\text{autoencoder}(x) \approx x$



[img: https://www.ibm.com/de-de/think/topics/variational-autoencoder]

Simple example:

Encode pictures of circles with random center and radius



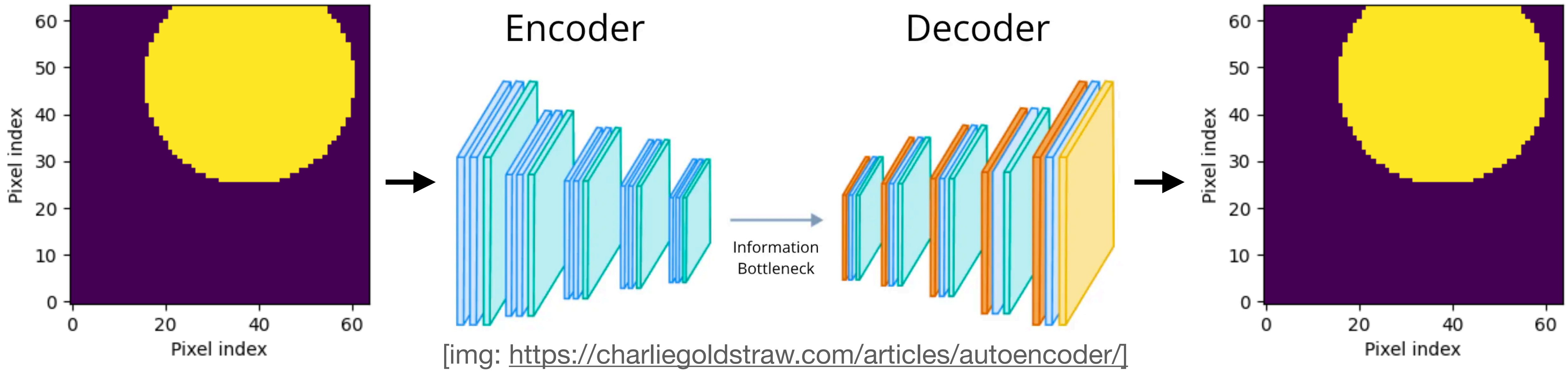
[img: https://charliegoldstraw.com/articles/autoencoder/]

```python
class Encoder(nn.Module):
    @nn.compact
    def __call__(self, x):
        x = nn.Conv(4, (5, 5), padding='same')(x)
        x = nn.sigmoid(x)
        x = nn.pooling.avg_pool(x, (4, 4), strides=(4, 4))

        x = nn.Conv(4, (5, 5), padding='same')(x)
        x = nn.sigmoid(x)
        x = nn.pooling.avg_pool(x, (2, 2), strides=(2, 2))

        x = nn.Conv(1, (3, 3), padding='same')(x)
        x = nn.sigmoid(x)

        return x
```

```python
class Decoder(nn.Module):
    @nn.compact
    def __call__(self, x):
        x = up_sample_2d(x, (2, 2))
        x = nn.Conv(4, (5, 5), padding='same')(x)
        x = nn.sigmoid(x)

        x = up_sample_2d(x, (4, 4))
        x = nn.Conv(4, (5, 5), padding='same')(x)
        x = nn.sigmoid(x)

        x = nn.Conv(1, (3, 3), padding='same')(x)

        return x
```

Simple example:

Encode pictures of circles with random center and radius



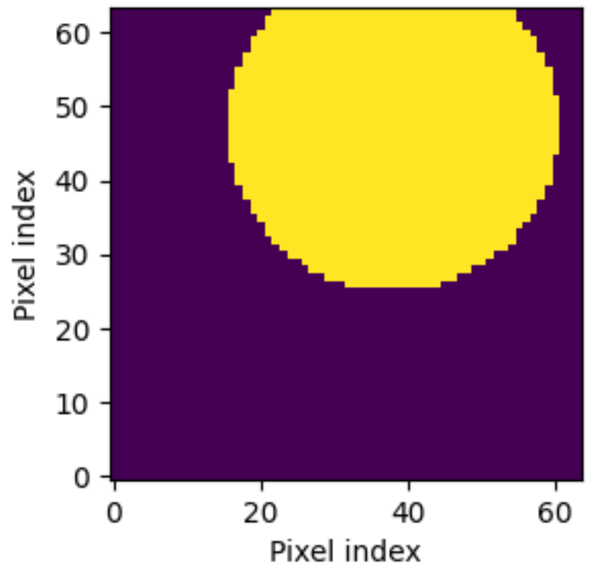
[img: https://charliegoldstraw.com/articles/autoencoder/]

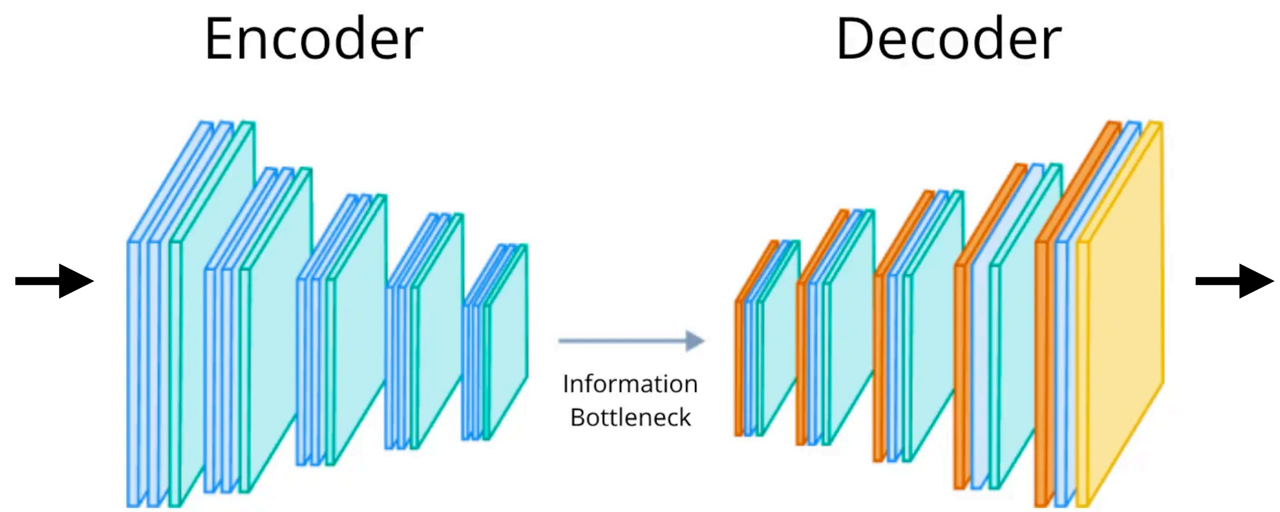
```
class ConvAutoenc(nn.Module):
    @nn.compact
    def __call__(self, x):
        x = Encoder()(x)
        x = Decoder()(x)
        return x
```

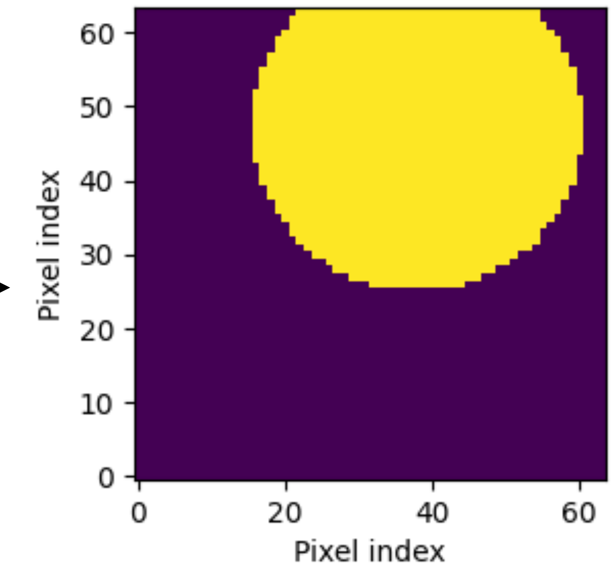
```
def loss_fn(params):
    # evaluate autoencoder for input X
    y_pred = state.apply_fn({'params': params}, X)
    # calculate mean squared deviation between output and X
    sq_dev = (y_pred - X)**2
    mean_sq_dev = sq_dev.mean()
    return mean_sq_dev
```

Simple example:

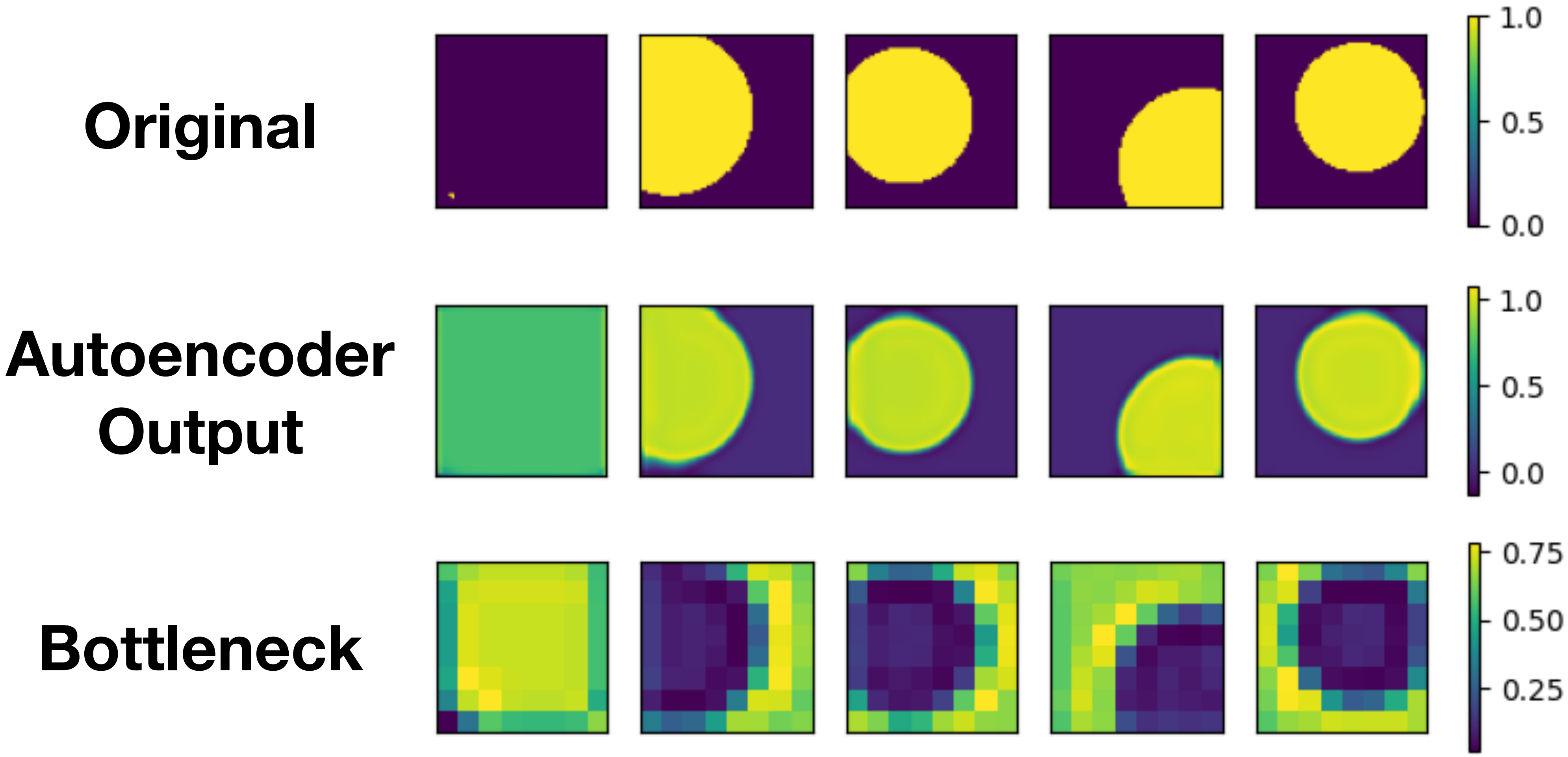
Encode pictures of circles with random center and radius





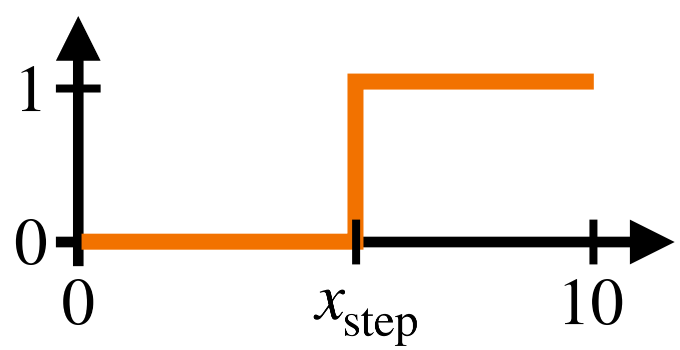


[img: https://charliegoldstraw.com/articles/autoencoder/]



(1) Go through the random-circles example and understand how an autoencoder works!

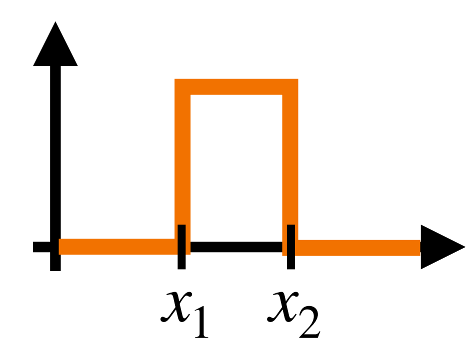
(2) Write an autoencoder, that encodes step functions where jump position is random! The autoencoder should only have on bottleneck neuron!



Todo: finish the autencoder class in autoencoder_step_function.ipynb and train the autoencoder

(3) Understand the meaning of the bottleneck values!

(4) Extend the code, such that it is able to cover one jump up, and one jump down!



(5) Bonus: What is the simplest network for (2) and (4)? Can you find an analytical solution for the weights?