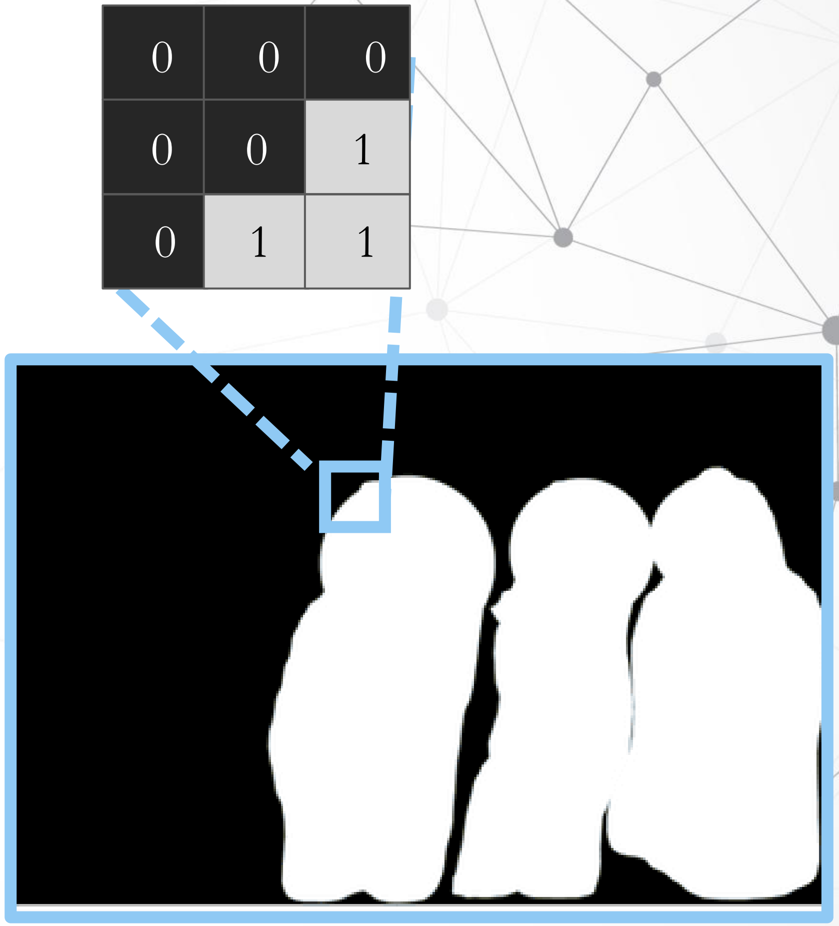
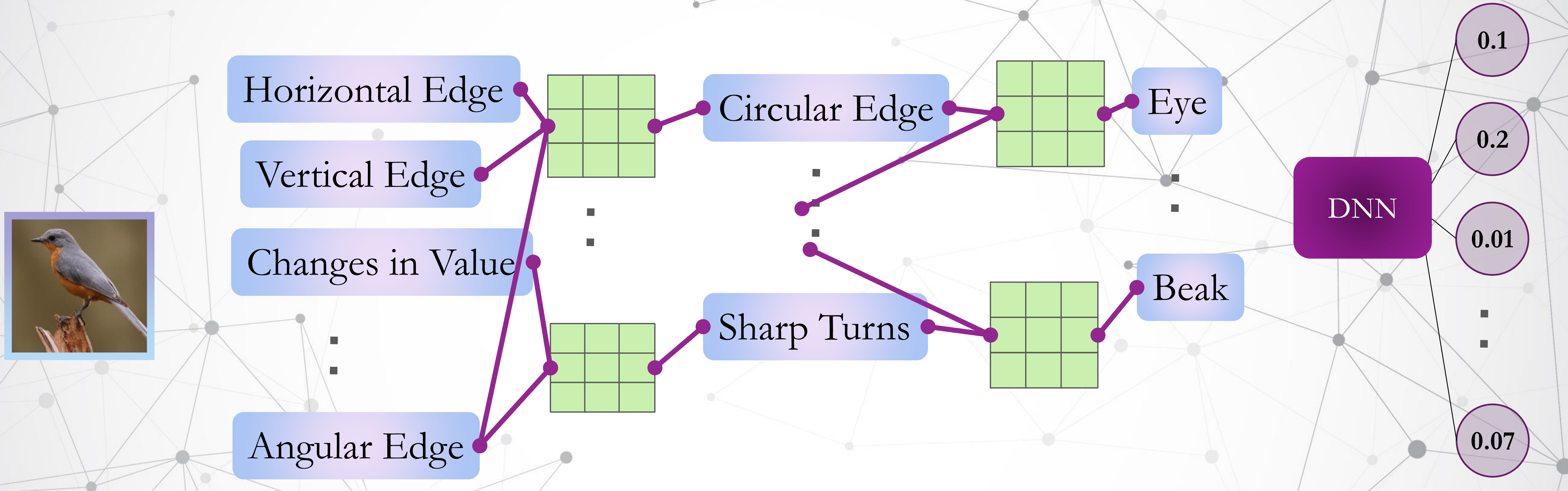
MLDL-I
Machine Learning and Deep Learning - I

Intro to Segmentation

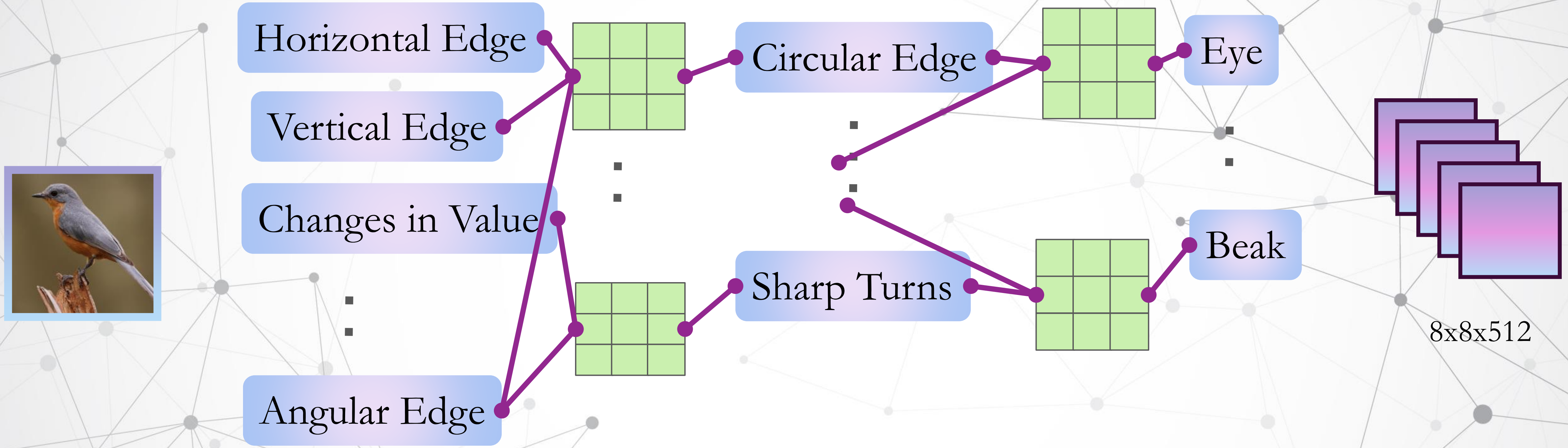


[Link]

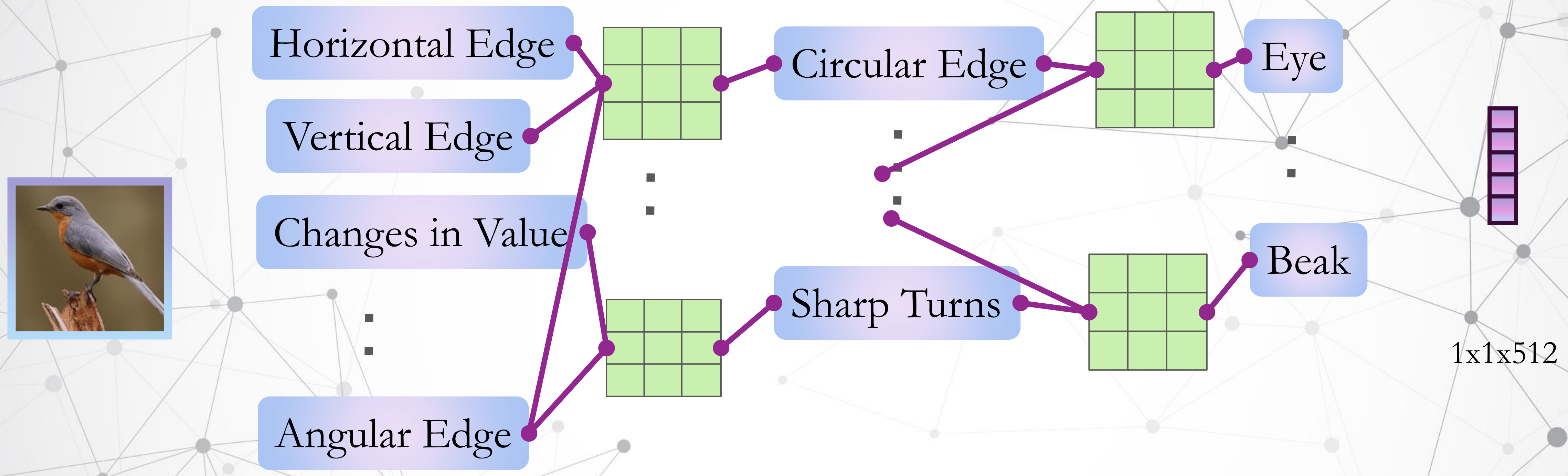
Semantic vs Instance Segmentation



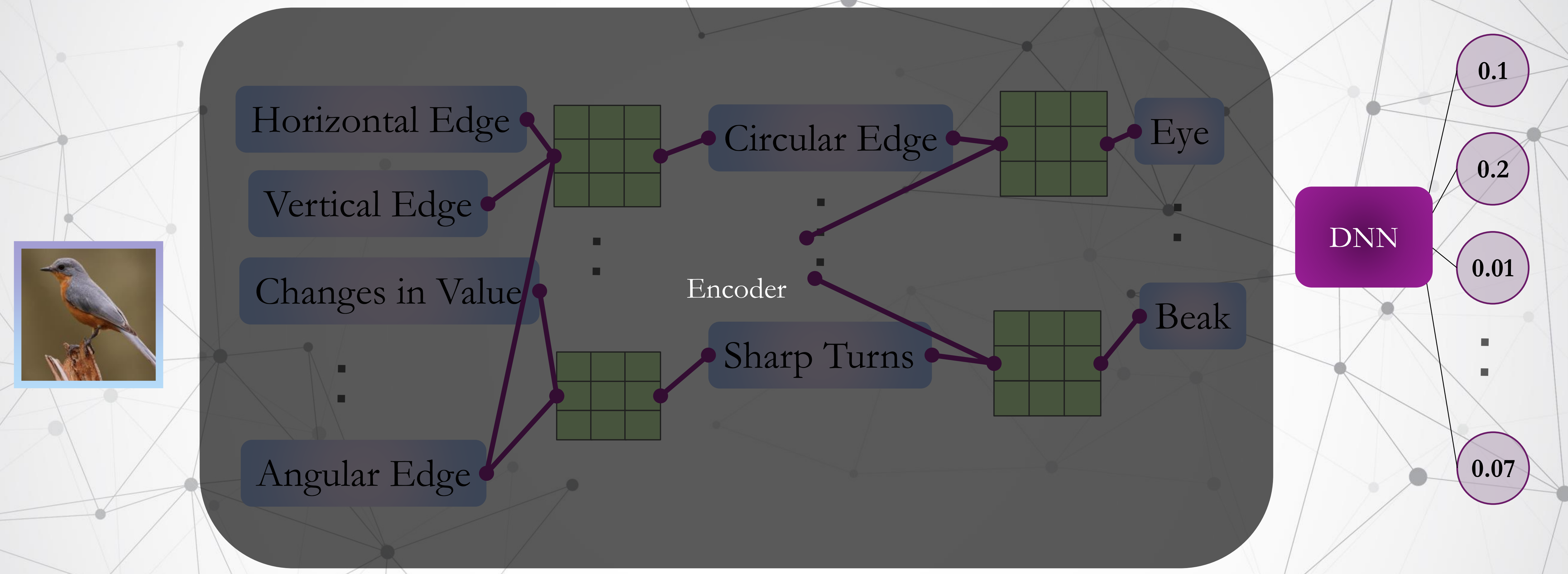

```
x = base_model(inp)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
out = Dense(num_classes, activation='sigmoid')(x)
```

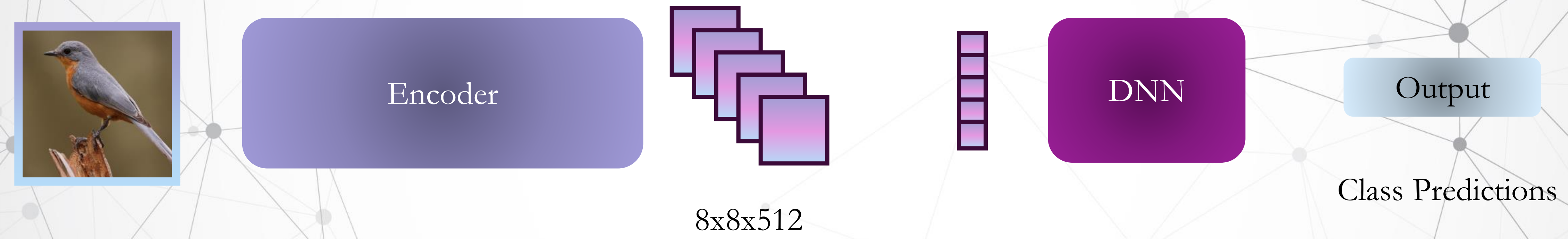


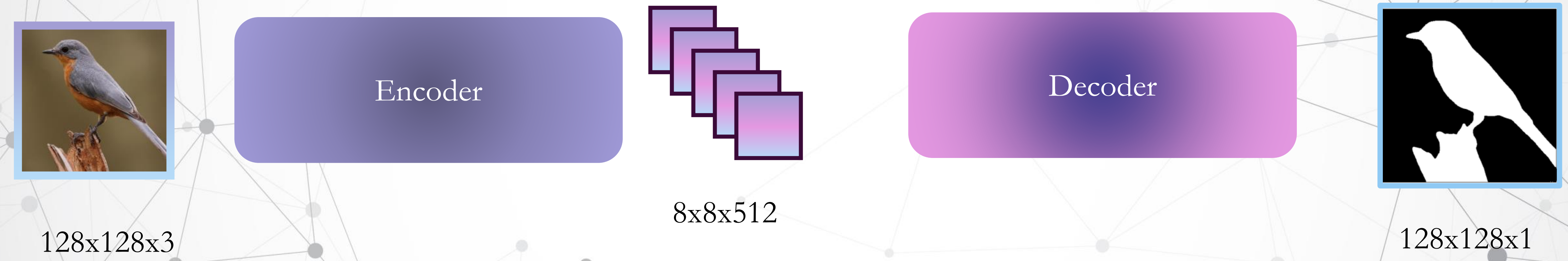

```
x = base_model(inp)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
out = Dense(num_classes, activation='sigmoid')(x)
```



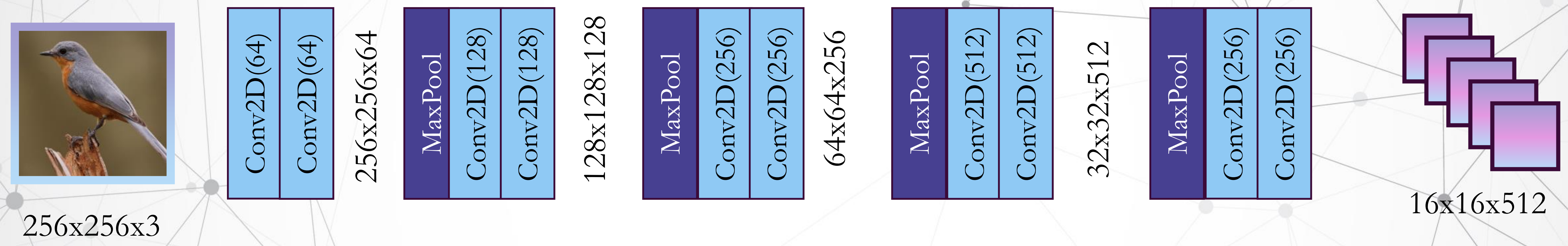

```
x = base_model(inp)
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
out = Dense(num_classes, activation='sigmoid')(x)
```

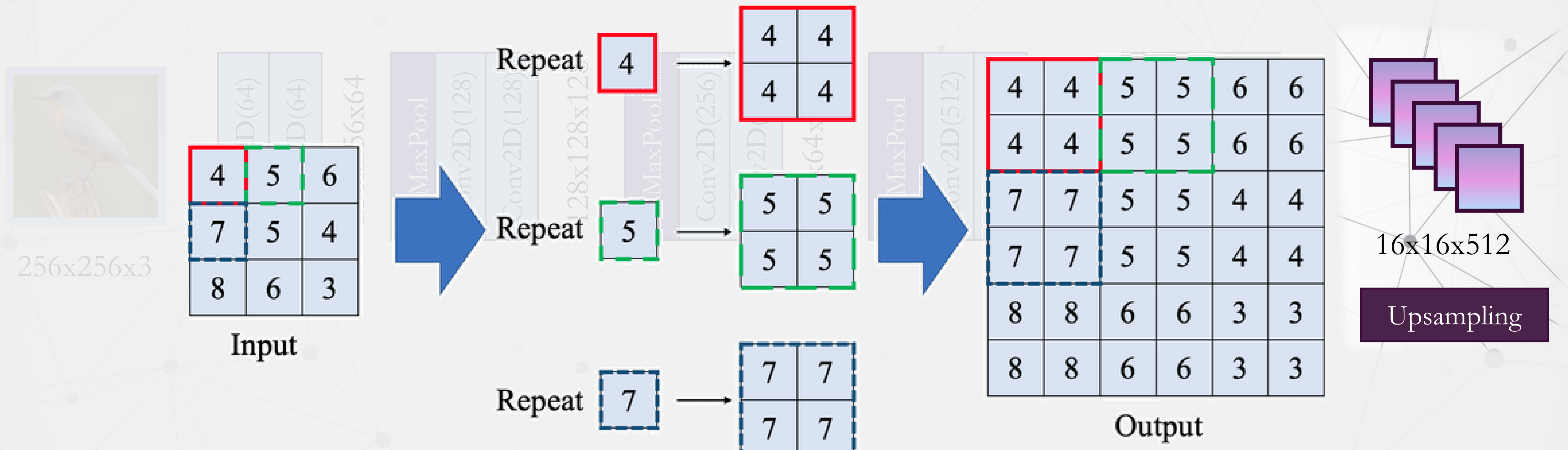

Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Encoder
Eye
Beak
DNN
0.1
0.2
0.01
0.07


Encoder
8x8x512
DNN
Output
Class Predictions

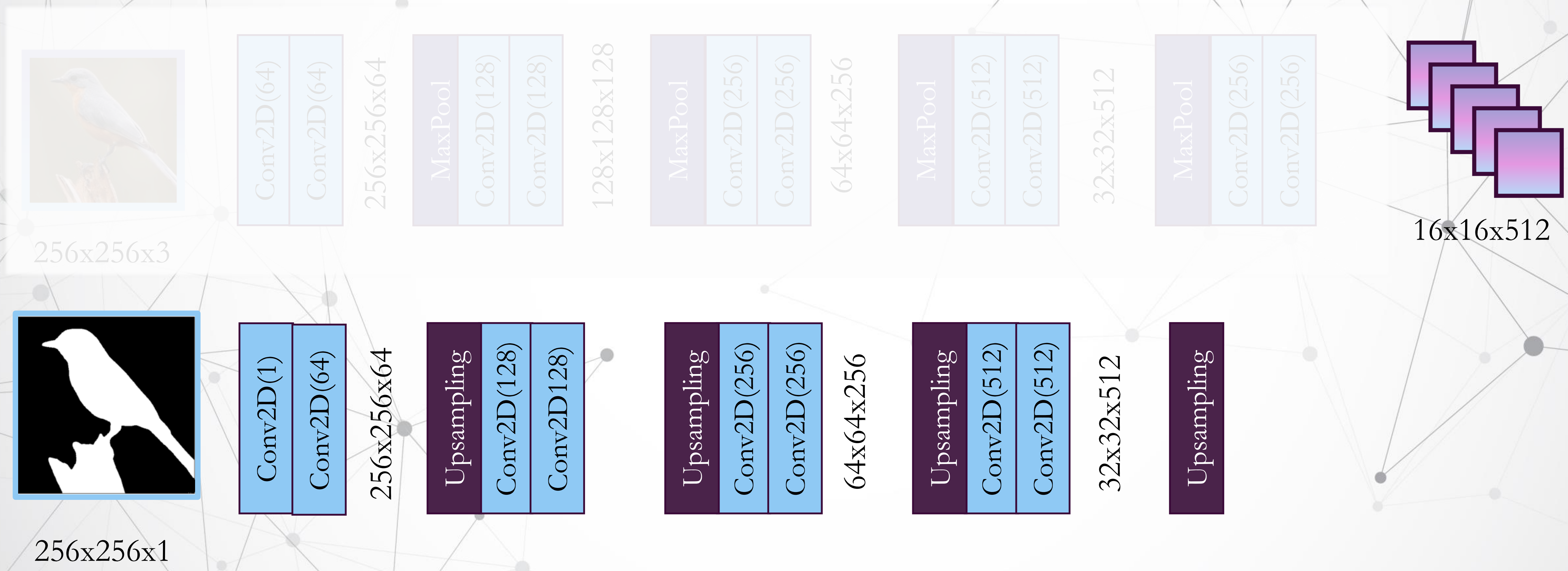


- Final Shape 8x8 to 128x128
- Build a new image based on image
- Correctly predict the pixels


Conv2D(64)
Conv2D(64)
256x256x64
MaxPool
Conv2D(128)
Conv2D(128)
128x128x128
MaxPool
Conv2D(256)
Conv2D(256)
64x64x256
MaxPool
Conv2D(512)
Conv2D(512)
32x32x512
MaxPool
Conv2D(256)
Conv2D(256)
256x256x3
16x16x512




```
tf.keras.layers.UpSampling2D(size=(2, 2))
```


256x256x3
Conv2D(64)
Conv2D(64)
256x256x64
MaxPool
Conv2D(128)
Conv2D(128)
128x128x128
MaxPool
Conv2D(256)
Conv2D(256)
64x64x256
MaxPool
Conv2D(512)
Conv2D(512)
32x32x512
MaxPool
Conv2D(256)
Conv2D(256)
16x16x512
256x256x1
Conv2D(1)
Conv2D(64)
256x256x64
Upsampling
Conv2D(128)
Conv2D128)
Upsampling
Conv2D(256)
Conv2D(256)
64x64x256
Upsampling
Conv2D(512)
Conv2D(512)
32x32x512
Upsampling

Pooling

```
tf.keras.layers.MaxPooling2D(pool_size = (2, 2))
```

512,512,3

256,256,3

128,128,3

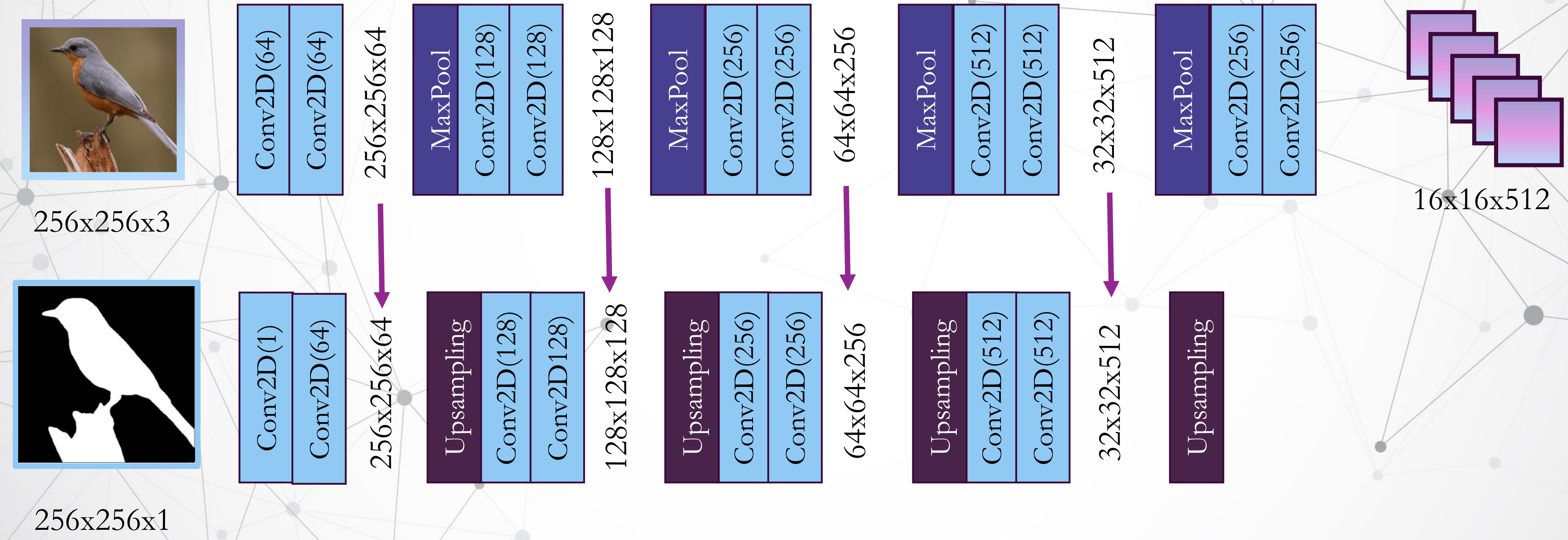
64,64,3

32,32,3

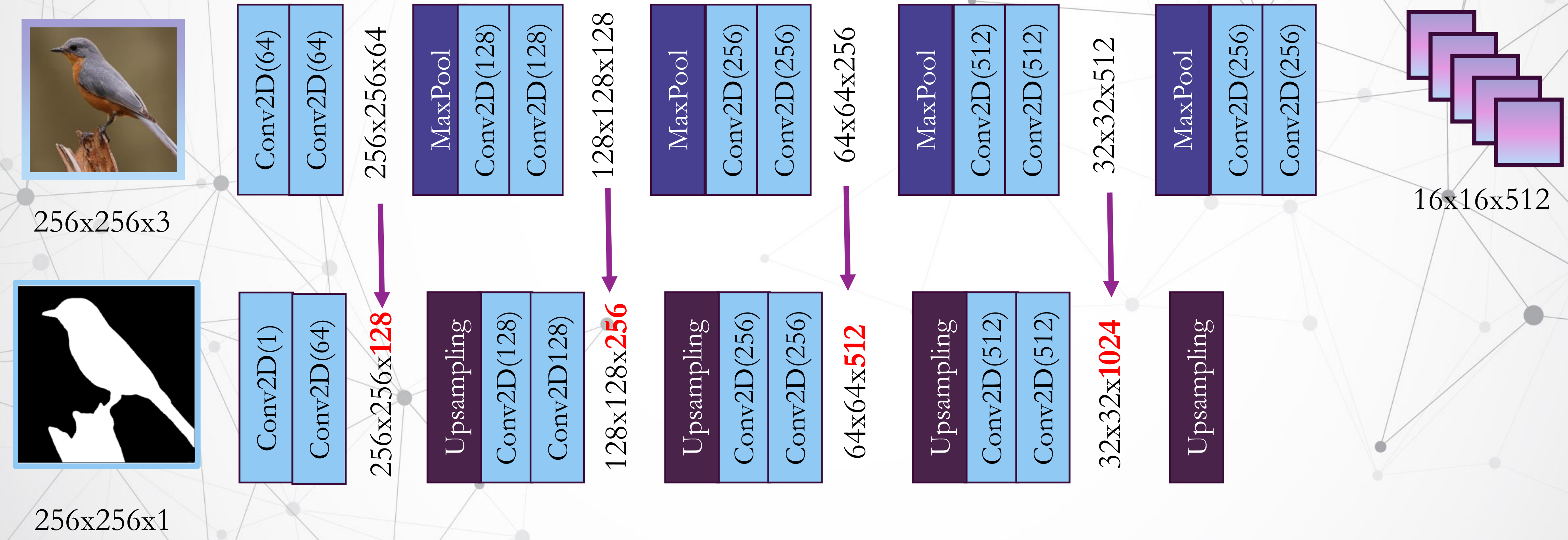
Upsampling

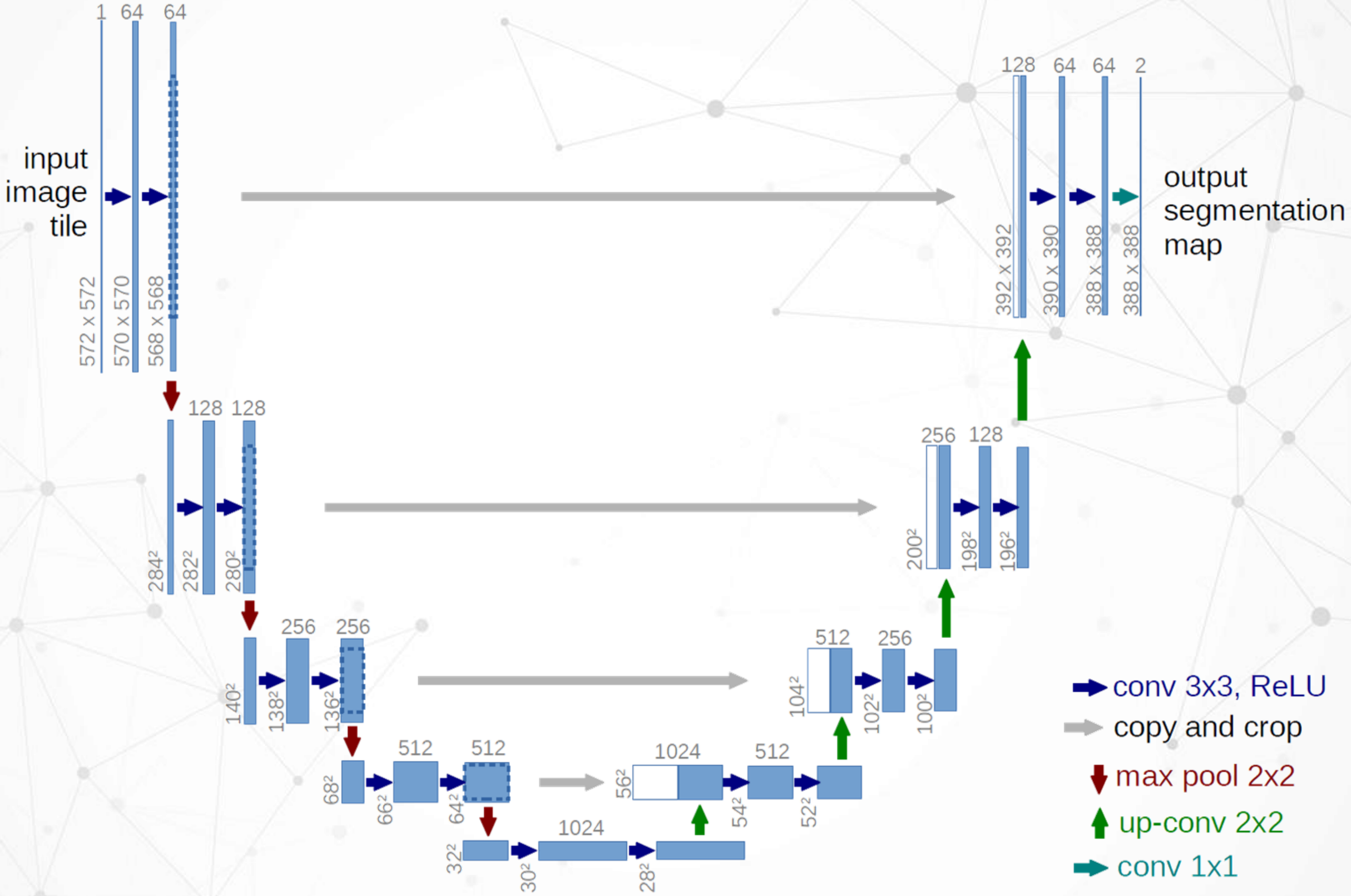
```
tf.keras.layers.UpSampling2D(size=(2, 2))
```

32,32,3 64,64,3 128,128,3 256,256,3 512,512,3



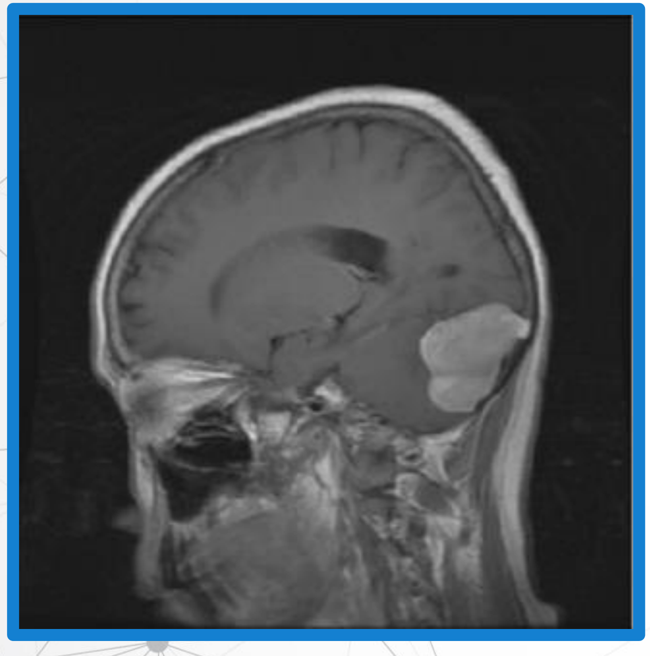
256x256x3
Conv2D(64)
Conv2D(64)
256x256x64
MaxPool
Conv2D(128)
Conv2D(128)
128x128x128
MaxPool
Conv2D(256)
Conv2D(256)
64x64x256
MaxPool
Conv2D(512)
Conv2D(512)
32x32x512
MaxPool
Conv2D(256)
Conv2D(256)
16x16x512
256x256x1
Conv2D(1)
Conv2D(64)
256x256x64
Upsampling
Conv2D(128)
Conv2D128)
128x128x128
Upsampling
Conv2D(256)
Conv2D(256)
64x64x256
Upsampling
Conv2D(512)
Conv2D(512)
32x32x512
Upsampling


256x256x3
Conv2D(64)
Conv2D(64)
256x256x64
MaxPool
Conv2D(128)
Conv2D(128)
128x128x128
MaxPool
Conv2D(256)
Conv2D(256)
64x64x256
MaxPool
Conv2D(512)
Conv2D(512)
32x32x512
MaxPool
Conv2D(256)
Conv2D(256)
16x16x512
256x256x1
Conv2D(1)
Conv2D(64)
256x256x128
Upsampling
Conv2D(128)
Conv2D128)
128x128x256
Upsampling
Conv2D(256)
Conv2D(256)
64x64x512
Upsampling
Conv2D(512)
Conv2D(512)
32x32x1024
Upsampling

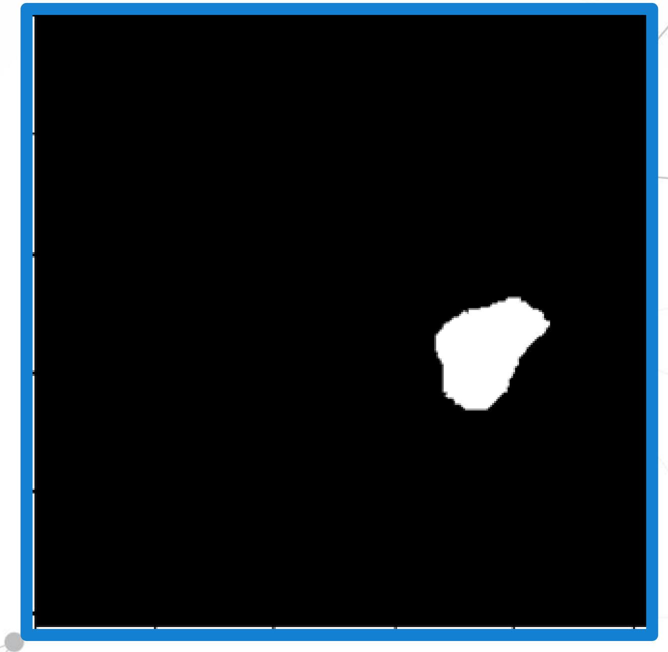

1
64
64
input image tile
572 x 572
570 x 570
568 x 568
128
128
284^2
282^2
280^2
256
256
140^2
138^2
136^2
512
512
68^2
66^2
64^2
1024
32^2
30^2
28^2
1024
512
56^2
54^2
52^2
512
256
104^2
102^2
100^2
256
128
200^2
198^2
196^2
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

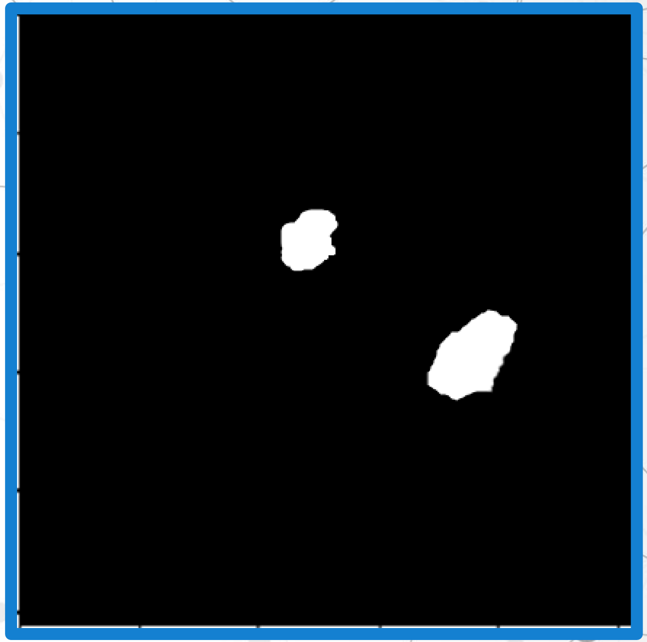
```
loss = tf.keras.losses.BinaryCrossentropy()
```



Input



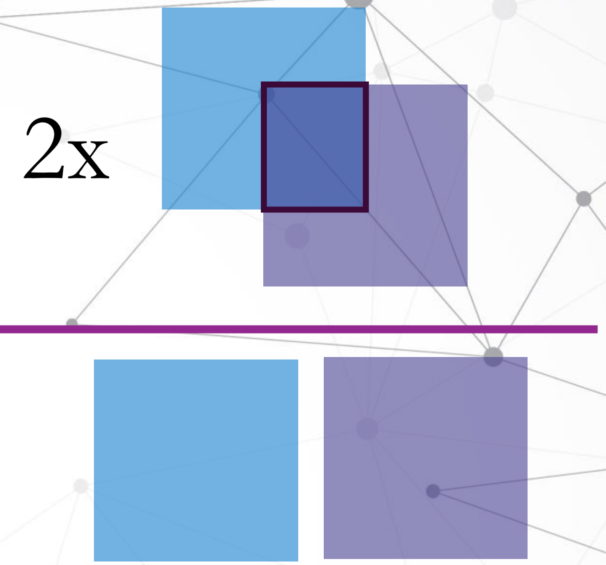
Ground Truth



Prediction

Dice Coefficient

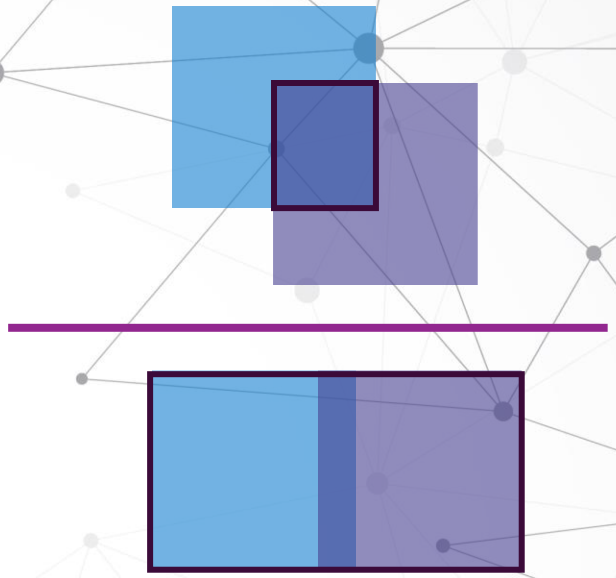
$$= \frac{2 \times \text{area of intersection}}{\text{area of prediction} + \text{area of ground truth}}$$




```
def dice_loss(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f* y_pred_f)
    val = (2. * intersection + K.epsilon()) / (K.sum(y_true_f) + K.sum(y_pred_f) + K.epsilon())
    return 1. - val
```

Intersection over Union

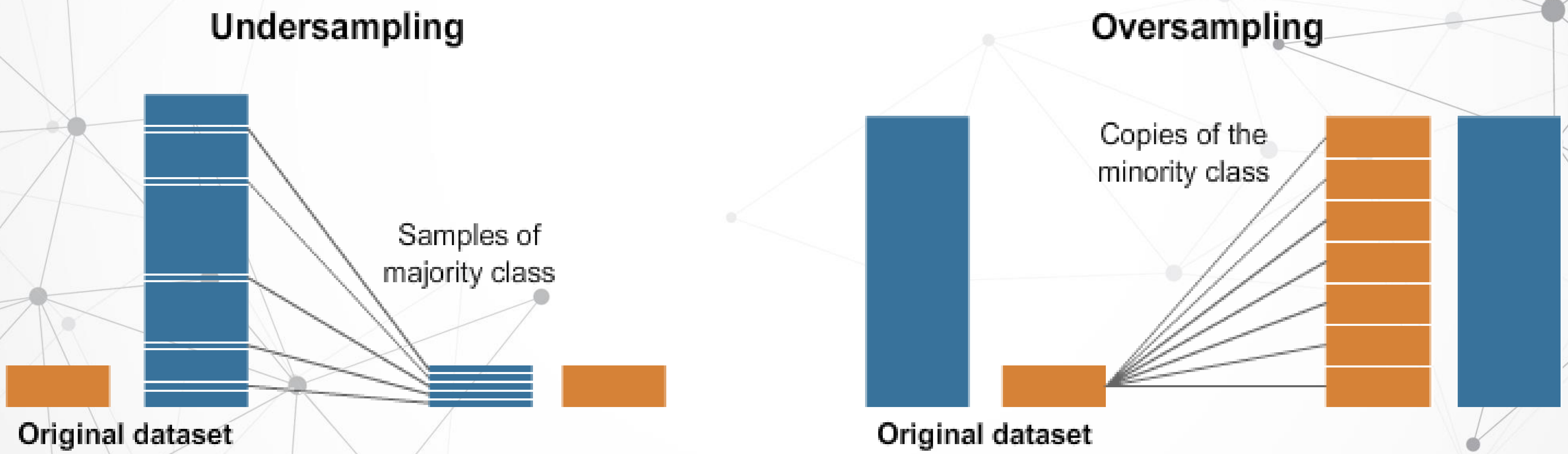
$$= \frac{\text{intersection of area}}{union\ of\ area}$$



```python
def jaccard_distance(y_true, y_pred):
    intersection = K.sum(y_true * y_pred)
    union = K.sum(y_true + y_pred) - intersection
    jac = (intersection) / union
    return 1 - jac
```

Class Imbalance

- Data Sampling



Other Approaches

- Class Weights
- Dedicated Model
- Ensembling

Any Question?