MLDL-I
Machine Learning and Deep Learning - I

tmax	tmin	rain	tmax_tomorrow
60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0
50.0	38.0	0.00	52.0
52.0	43.0	0.00	56.0
56.0	49.0	0.24	54.0
54.0	50.0	0.40	57.0
57.0	50.0	0.00	57.0
57.0	50.0	0.31	58.0
58.0	52.0	0.05	59.0
59.0	54.0	0.25	58.0
58.0	53.0	1.94	56.0
56.0	51.0	0.63	61.0
61.0	56.0	0.62	59.0
59.0	54.0	0.00	58.0
58.0	53.0	0.00	60.0
60.0	53.0	0.14	60.0
60.0	53.0	1.21	61.0

Batch_size = 5

Epoch 2, Step 1:

60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0

```
print(len(df))
13509
```

Dataset length, m = 13509

Calculate Prediction

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

Estimate Error

$$J(w,b) = (\hat{y} - y_{actual})^2$$

Update Parameters

$$w = w - \frac{\partial J}{\partial w} \qquad b = b - \frac{\partial J}{\partial b}$$

tmax	tmin	rain	tmax_tomorrow
60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0
50.0	38.0	0.00	52.0
52.0	43.0	0.00	56.0
56.0	49.0	0.24	54.0
54.0	50.0	0.40	57.0
57.0	50.0	0.00	57.0
57.0	50.0	0.31	58.0
58.0	52.0	0.05	59.0
59.0	54.0	0.25	58.0
58.0	53.0	1.94	56.0
56.0	51.0	0.63	61.0
61.0	56.0	0.62	59.0
59.0	54.0	0.00	58.0
58.0	53.0	0.00	60.0
60.0	53.0	0.14	60.0
60.0	53.0	1.21	61.0

Batch_size = 5

Epoch 2, Step 1:

60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0

```
print(len(df))
```

```
13509
```

Dataset length, m = 13509

Calculate Prediction

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

Estimate Error

$$J(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_{i_{actual}})^2$$

Update Parameters

$$\boldsymbol{w} = \boldsymbol{w} - \frac{\partial J}{\partial \boldsymbol{w}} \qquad \boldsymbol{b} = \boldsymbol{b} - \frac{\partial J}{\partial \boldsymbol{b}}$$

tmax	tmin	rain	tmax_tomorrow
60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0
50.0	38.0	0.00	52.0
52.0	43.0	0.00	56.0
56.0	49.0	0.24	54.0
54.0	50.0	0.40	57.0
57.0	50.0	0.00	57.0
57.0	50.0	0.31	58.0
58.0	52.0	0.05	59.0
59.0	54.0	0.25	58.0
58.0	53.0	1.94	56.0
56.0	51.0	0.63	61.0
61.0	56.0	0.62	59.0
59.0	54.0	0.00	58.0
58.0	53.0	0.00	60.0
60.0	53.0	0.14	60.0
60.0	53.0	1.21	61.0

Batch_size = 5

Epoch 2, Step 1:

60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0

```
print(len(df))
13509
```

Dataset length, m = 13509

Calculate Prediction

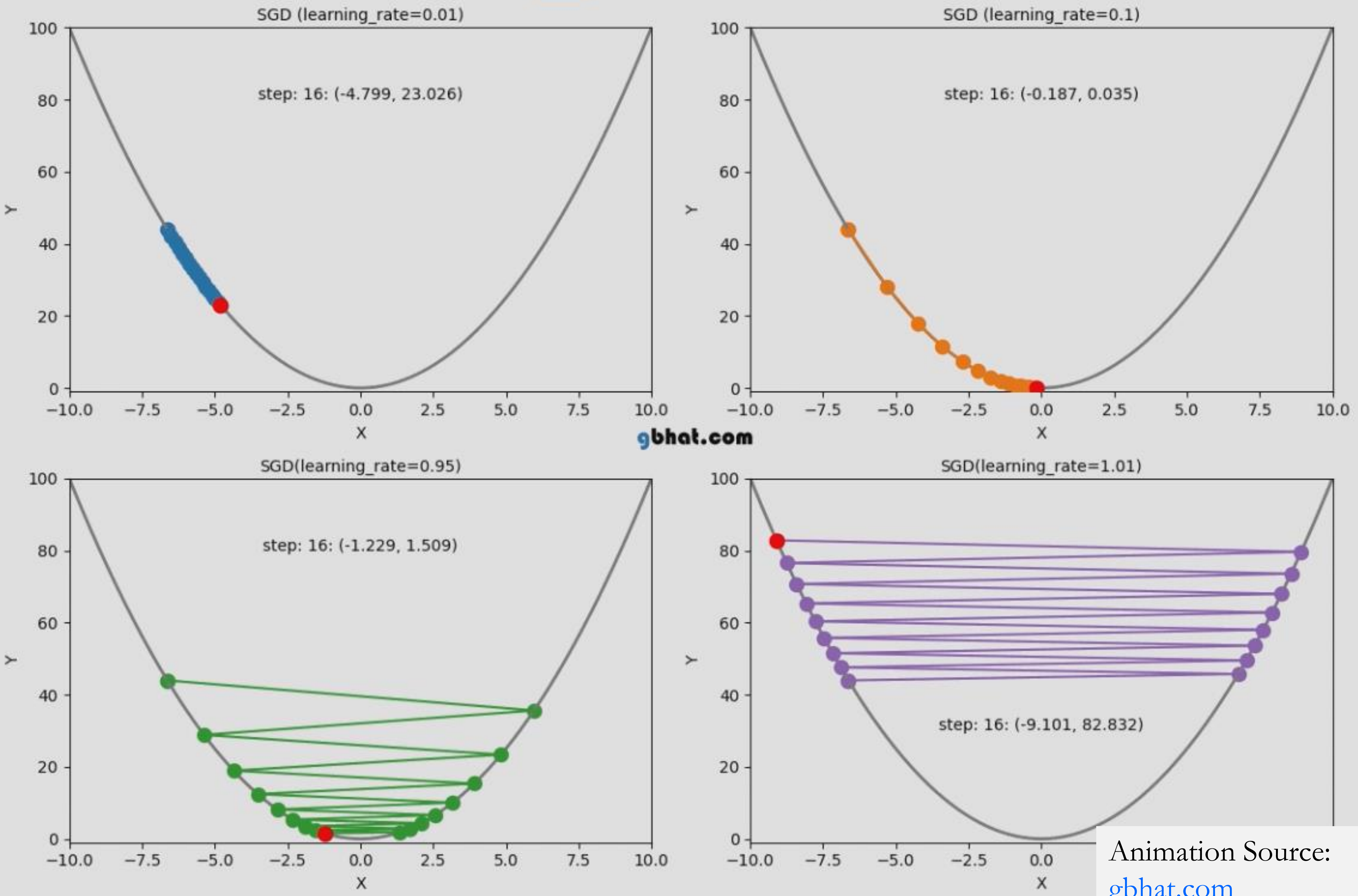
$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

Estimate Error

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} (\hat{y}_i - y_{i_{actual}})^2$$

Update Parameters

$$w = w - \alpha \frac{\partial J}{\partial w} \quad b = b - \alpha \frac{\partial J}{\partial b}$$



Animation Source: gbhat.com

tmax	tmin	rain	tmax_tomorrow
60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0
50.0	38.0	0.00	52.0
52.0	43.0	0.00	56.0
56.0	49.0	0.24	54.0
54.0	50.0	0.40	57.0
57.0	50.0	0.00	57.0
57.0	50.0	0.31	58.0
58.0	52.0	0.05	59.0
59.0	54.0	0.25	58.0
58.0	53.0	1.94	56.0
56.0	51.0	0.63	61.0
61.0	56.0	0.62	59.0
59.0	54.0	0.00	58.0
58.0	53.0	0.00	60.0
60.0	53.0	0.14	60.0
60.0	53.0	1.21	61.0

Batch_size = 5

Epoch 2, Step 1:

60.0	35.0	0.00	52.0
52.0	39.0	0.00	52.0
52.0	35.0	0.00	53.0
53.0	36.0	0.00	52.0
52.0	35.0	0.00	50.0

```
print(len(df))
```

```
13509
```

Dataset length, m = 13509

Calculate Prediction

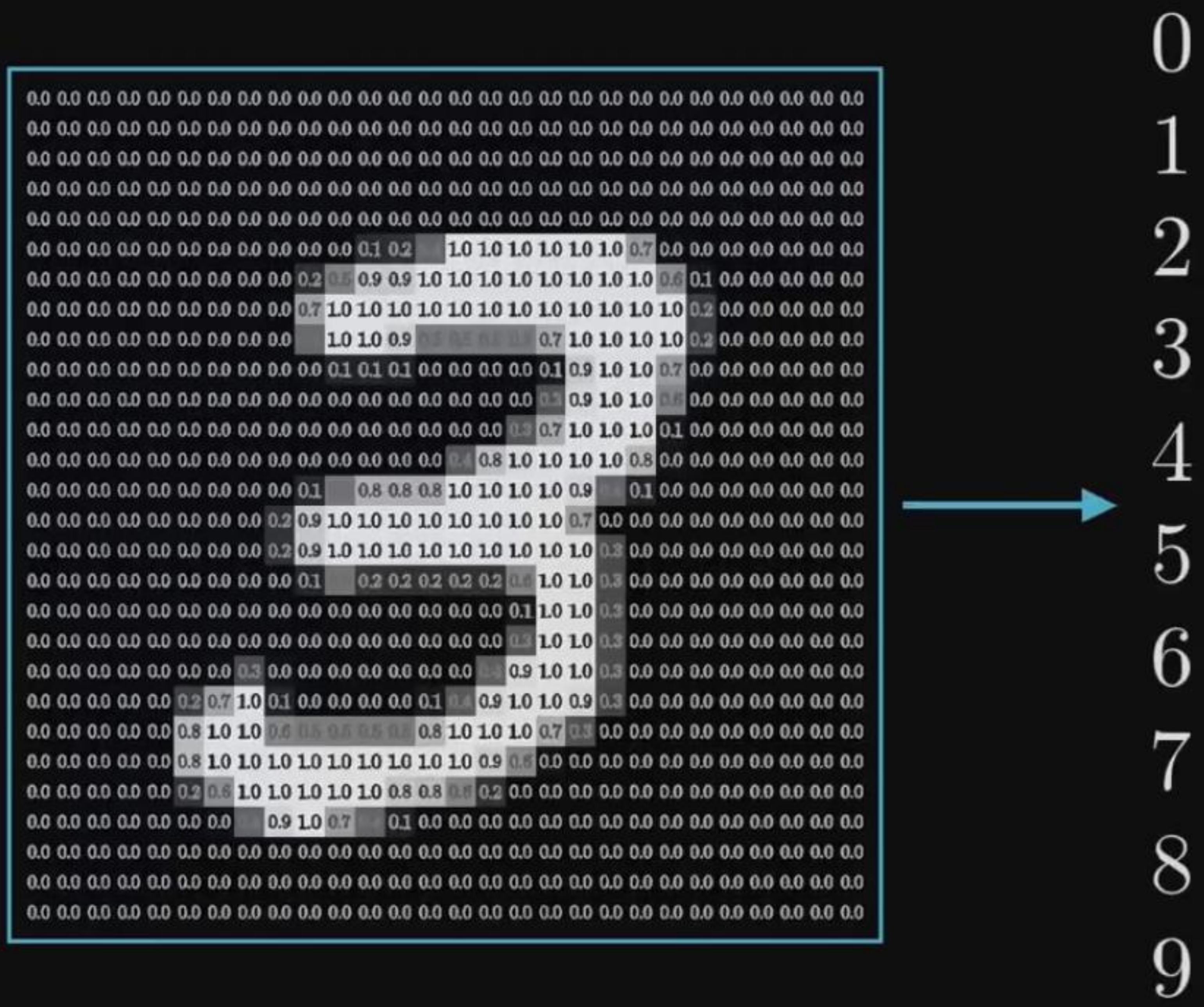
$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

Estimate Error

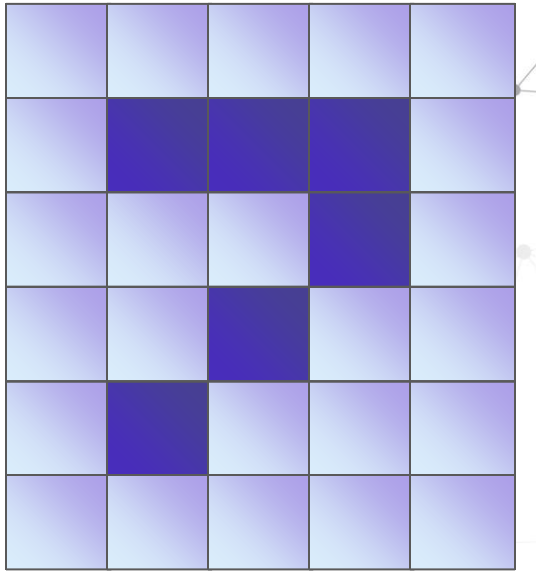
$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_{i_{actual}})^2$$

Update Parameters

$$w = w - \alpha \frac{\partial J}{\partial w} \quad b = b - \alpha \frac{\partial J}{\partial b}$$

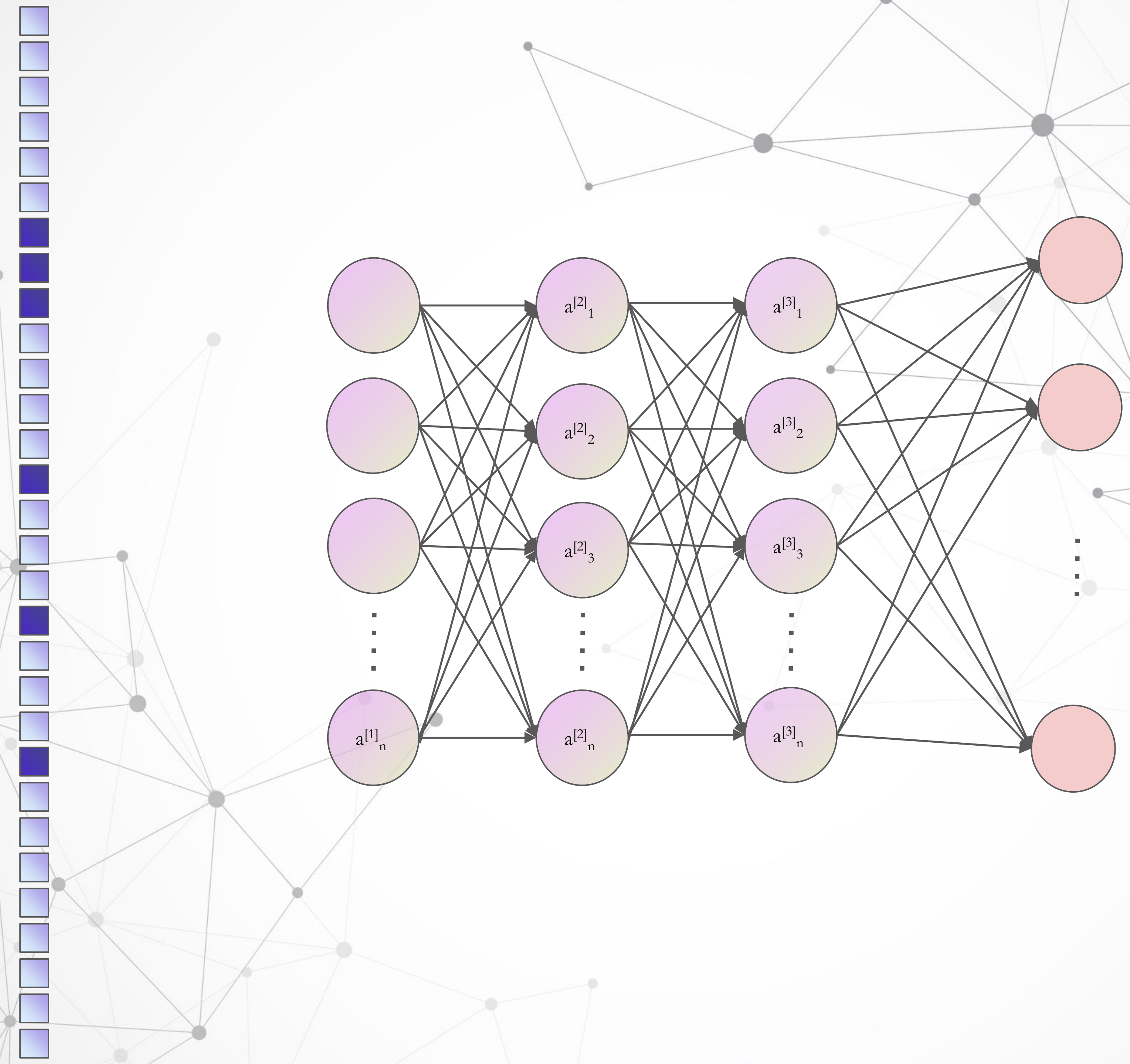


Source: [3Blue1Brown]

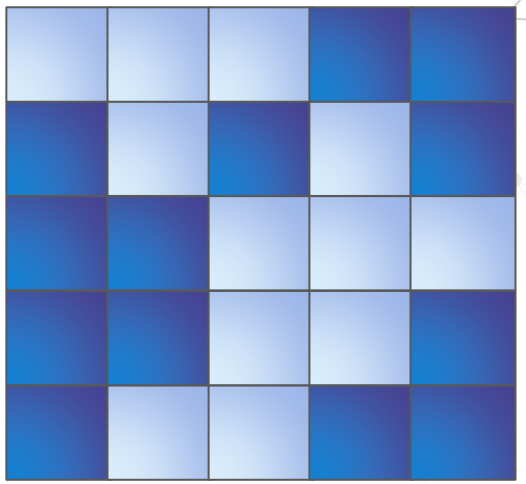


Input Image

```
tf.keras.layers.Flatten()
```



a[2]1
a[2]2
a[2]3
a[1]n
a[2]n
a[3]1
a[3]2
a[3]3
a[3]n



Input Image

1	1	1	0	0
0	1	0	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input Image

1	0	1
0	1	0
1	0	1

Convolutional Kernel

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

1	0	1
0	1	0
1	0	1

Animation Source:
Link

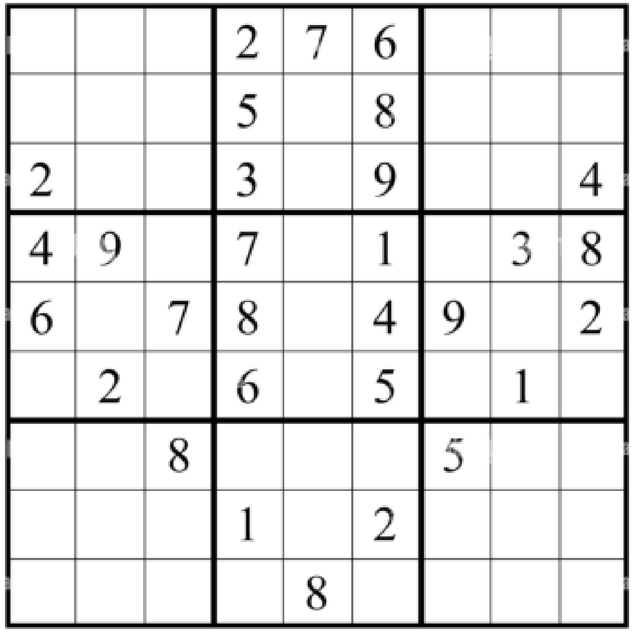
Convolutional Filter

400 x 600

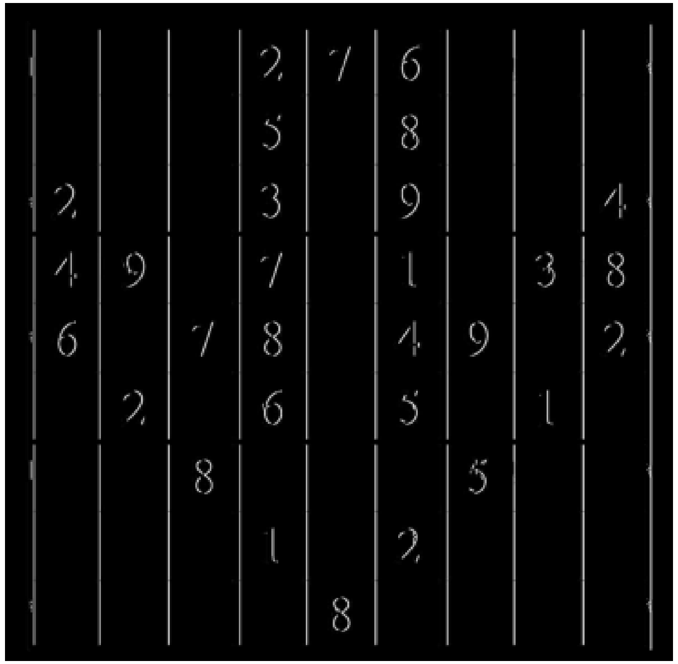
-1	0	1
-2	0	2
-1	0	1

3 x 3

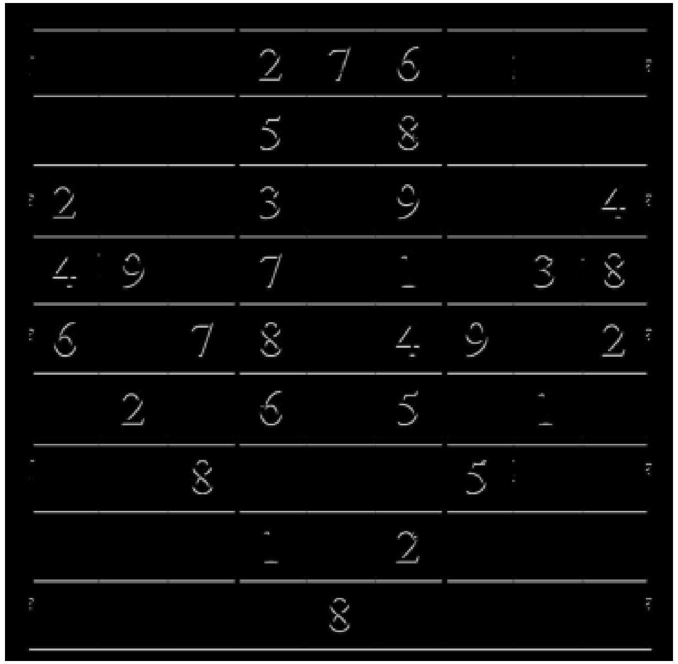
Original Image



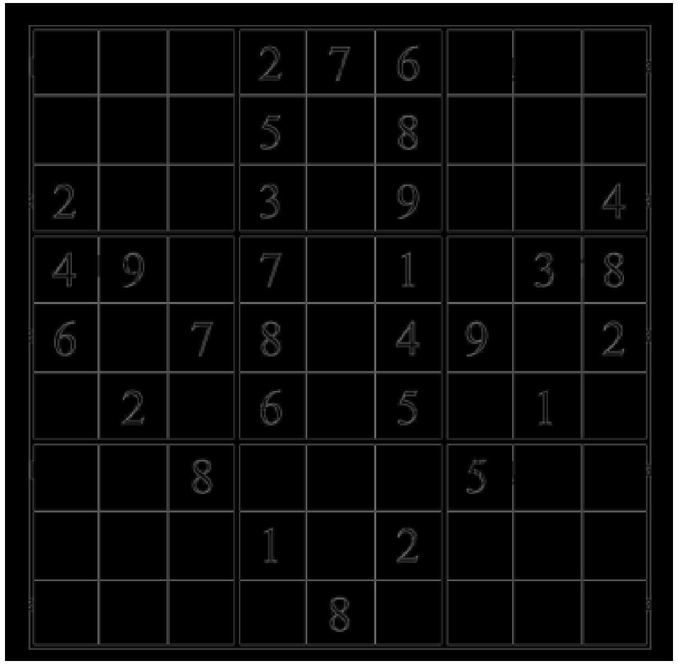
Sobel X



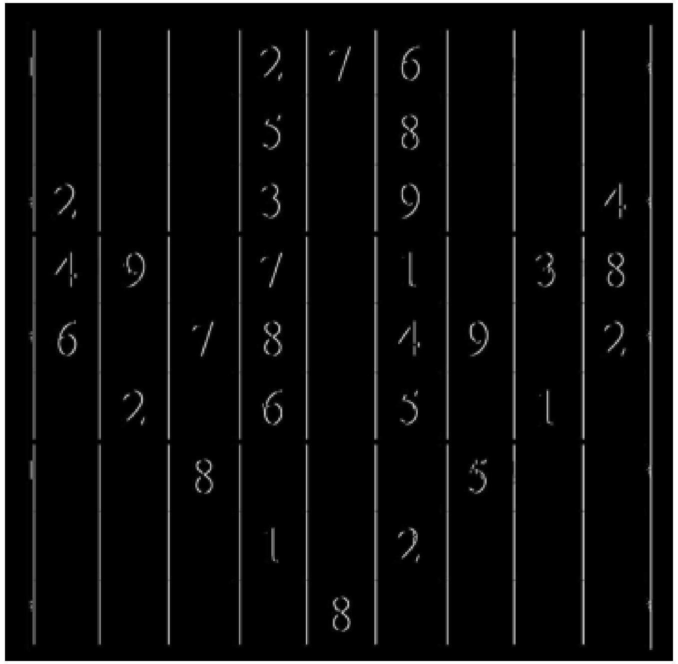
Sobel Y



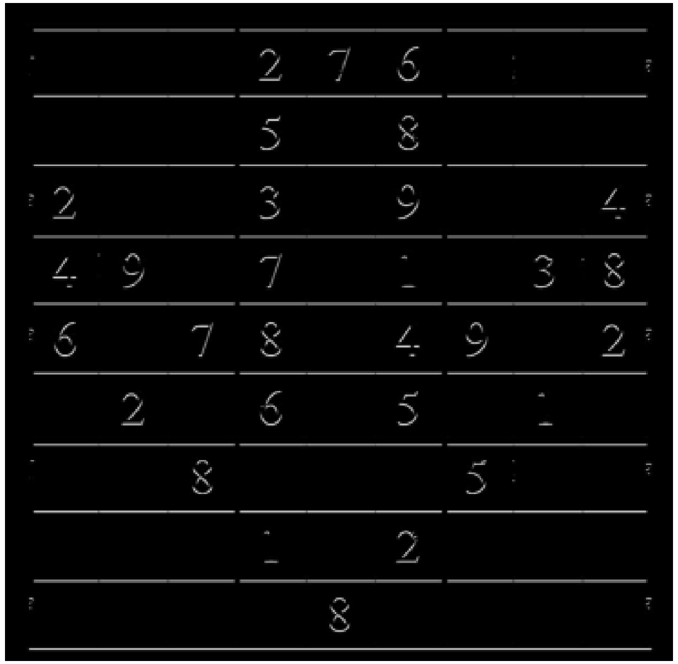
Laplacian



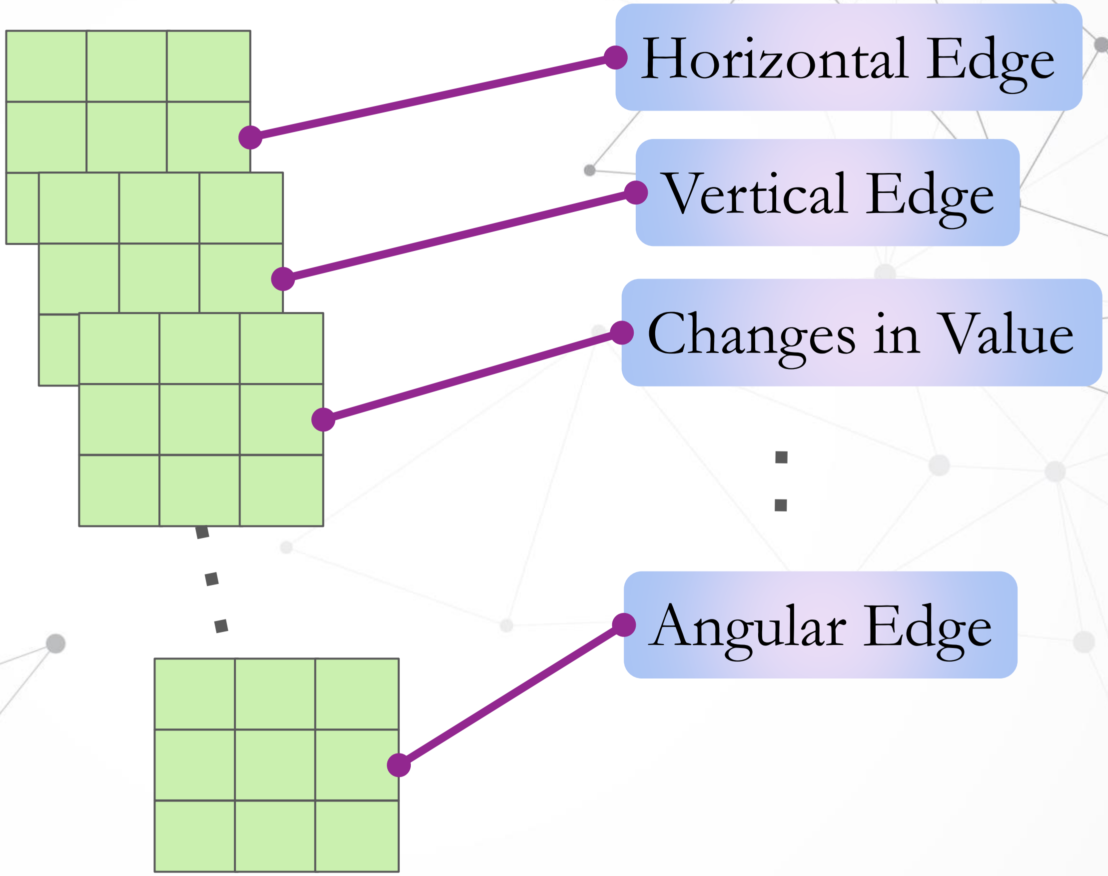
Prewitt X

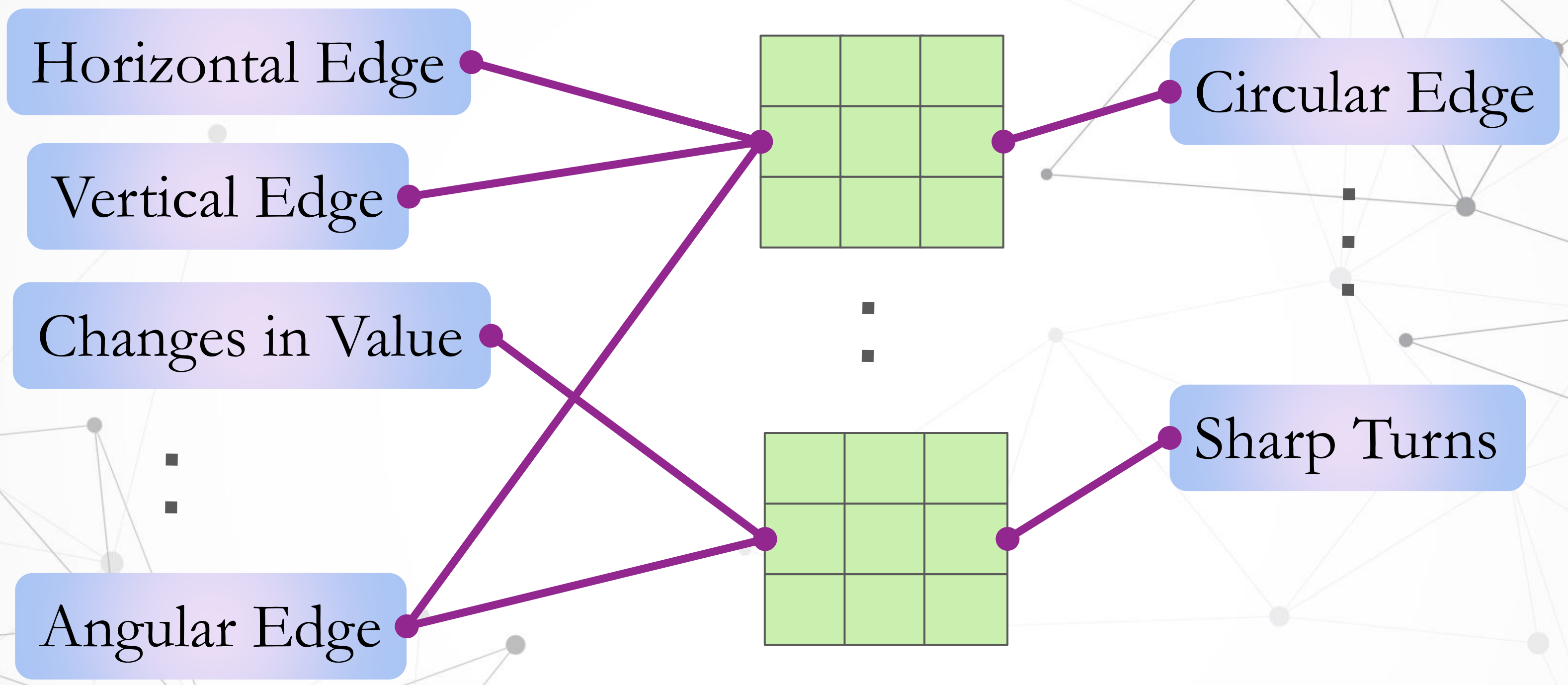


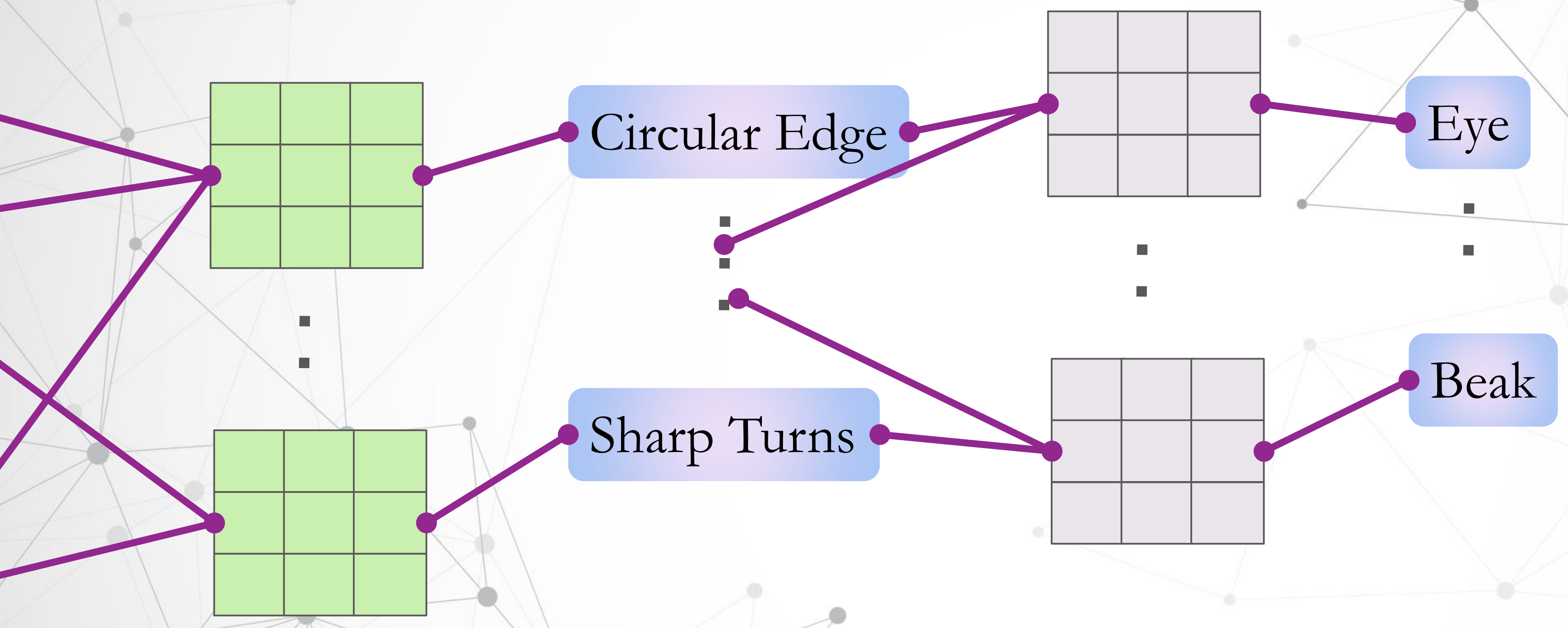
Prewitt Y

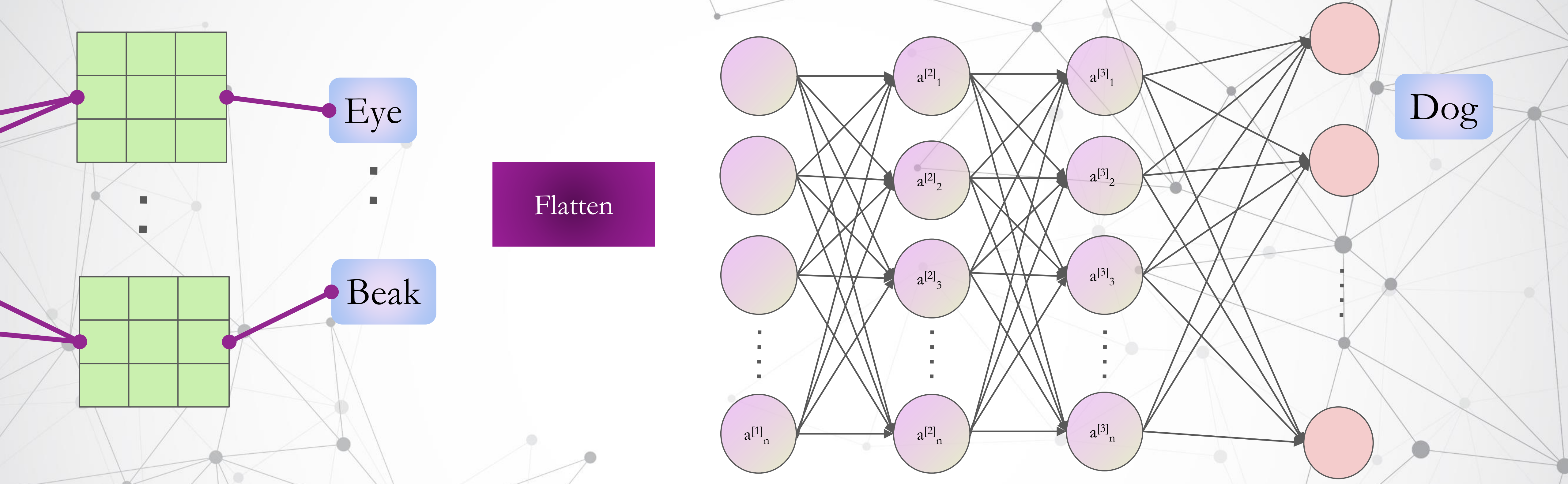


Intro to CNN


Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge


Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns

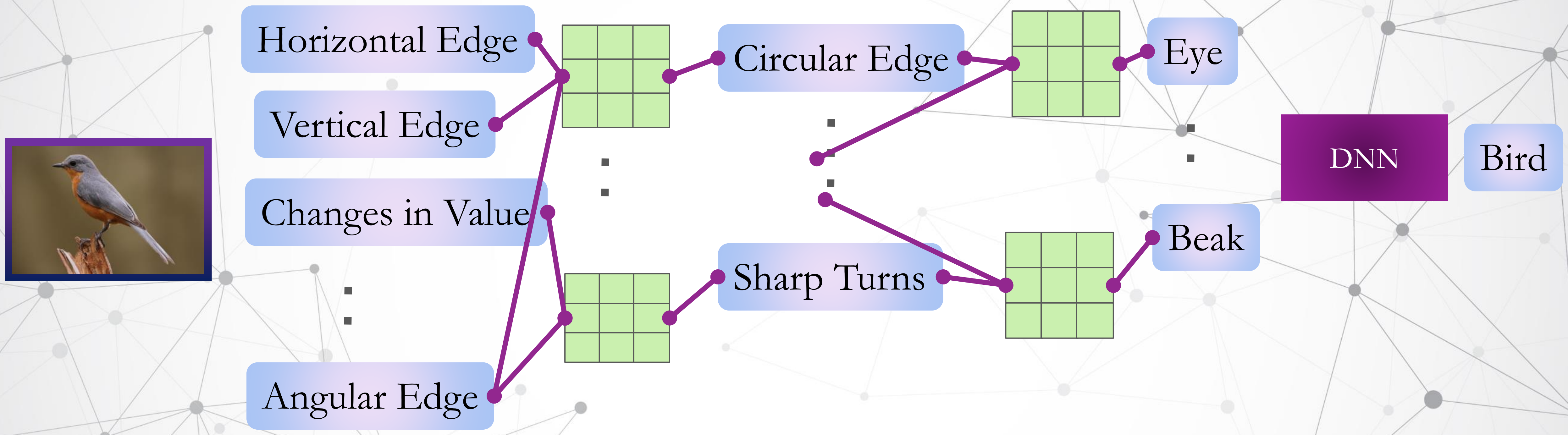

Circular Edge
Sharp Turns
Eye
Beak



Eye
Beak
Flatten
a[1]n
a[2]1
a[2]2
a[2]3
a[2]n
a[3]1
a[3]2
a[3]3
a[3]n
Bird
Dog

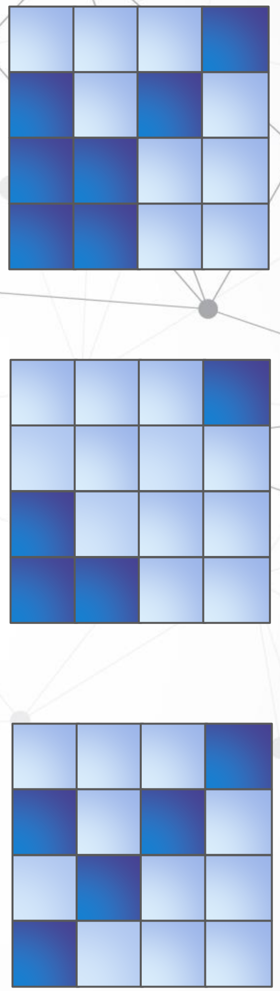
Has_Tail	Number_of_Legs	Has_Fur	Lives_in_Water	Can_Fly	Lays_Eggs	Is_Mammal
True	4	True	False	False	False	True
True	4	True	False	False	False	True
False	8	False	False	False	True	False
False	0	False	False	False	True	False
False	2	False	False	True	True	False
True	2	True	False	False	False	True
True	4	False	False	False	False	True
False	0	False	True	False	True	False
False	2	False	False	False	True	False
True	4	False	False	False	False	True

Animal
Cat
Dog
Spider
Snake
Parrot
Kangaroo
Elephant
Fish
Chicken
Cow


Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Eye
Beak
DNN
Bird

Intro to CNN

?	?	?
?	?	?
?	?	?

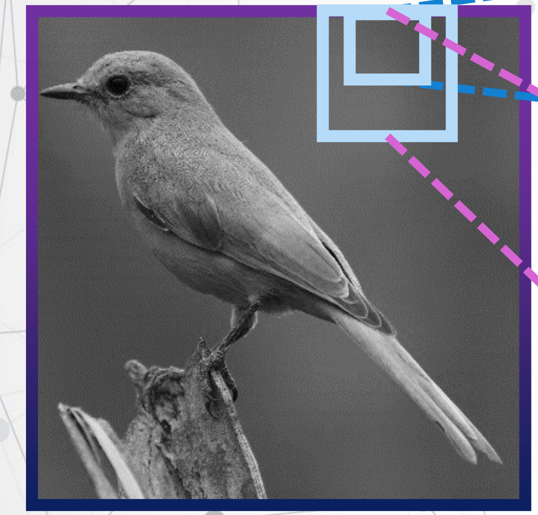


```
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

- Filter
- Kernel Size
- Padding
- Strides

```python
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

Kernel Size



256 x 256 x 1

1	0	1
0	1	0
1	0	1

1	1	1	0	0
0	1	0	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

```python
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

Stride

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (1,1)

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (2,2)

```python
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

Stride

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (1,1)

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (2,2)

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

1	0	1
0	1	0
1	0	1

$$Output\ Size = \frac{Input\ Size\ - Kernel\ Size}{Stride} + 1$$

Stride

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

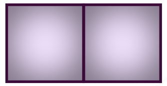
Stride (2,2)

$$Output\ Size = \frac{Input\ Size\ - Kernel\ Size}{Stride} + 1$$

Stride

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (2,2)

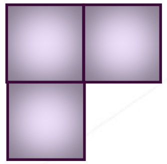


$$Output\ Size = \frac{Input\ Size\ - Kernel\ Size}{Stride} + 1$$

Stride

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (2,2)

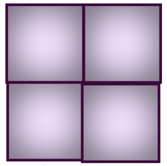


$$Output\ Size = \frac{Input\ Size\ - Kernel\ Size}{Stride} + 1$$

Stride

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Stride (2,2)



$$Output\ Size = \frac{Input\ Size\ - Kernel\ Size}{Stride} + 1$$

Padding

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

$$Output\ Size$$

$$= \frac{Input\ Size\ - Kernel\ Size}{Stride} + 1$$

```
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

Padding

0	1	0	0	0	0	1
0	0	1	0	0	1	0
0	0	0	1	1	0	0
0	1	0	0	0	1	0
0	0	1	0	0	1	0
0	0	0	1	0	1	0

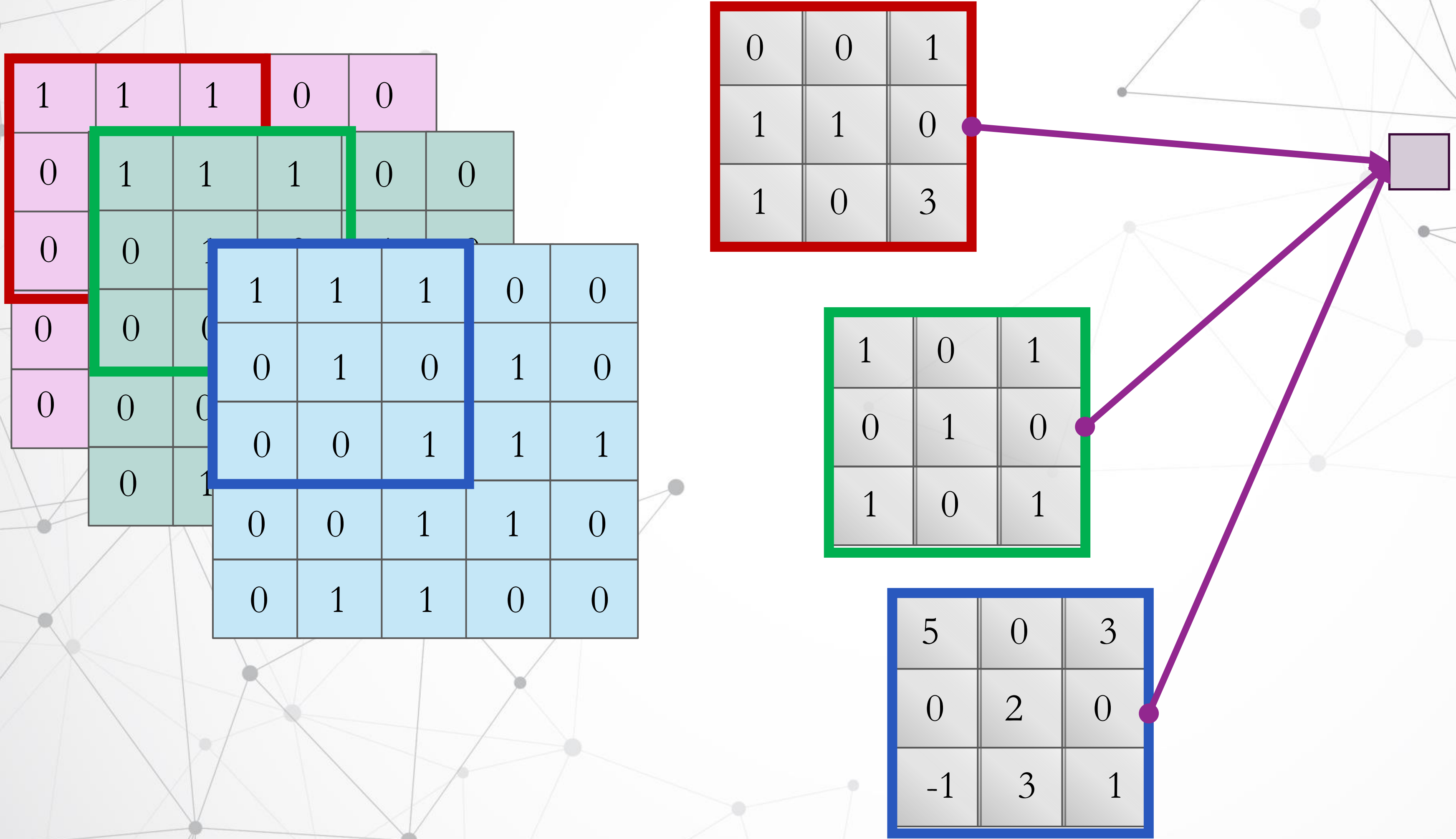
$$\mathit{Output\ Size} = \frac{\mathit{Input\ Size} + \mathit{Padding} - \mathit{Kernel\ Size}}{\mathit{Stride}} + 1$$

'valid'
'same'

```
tf.keras.layers.ZeroPadding2D(padding=((1, 0),(0,0))
```

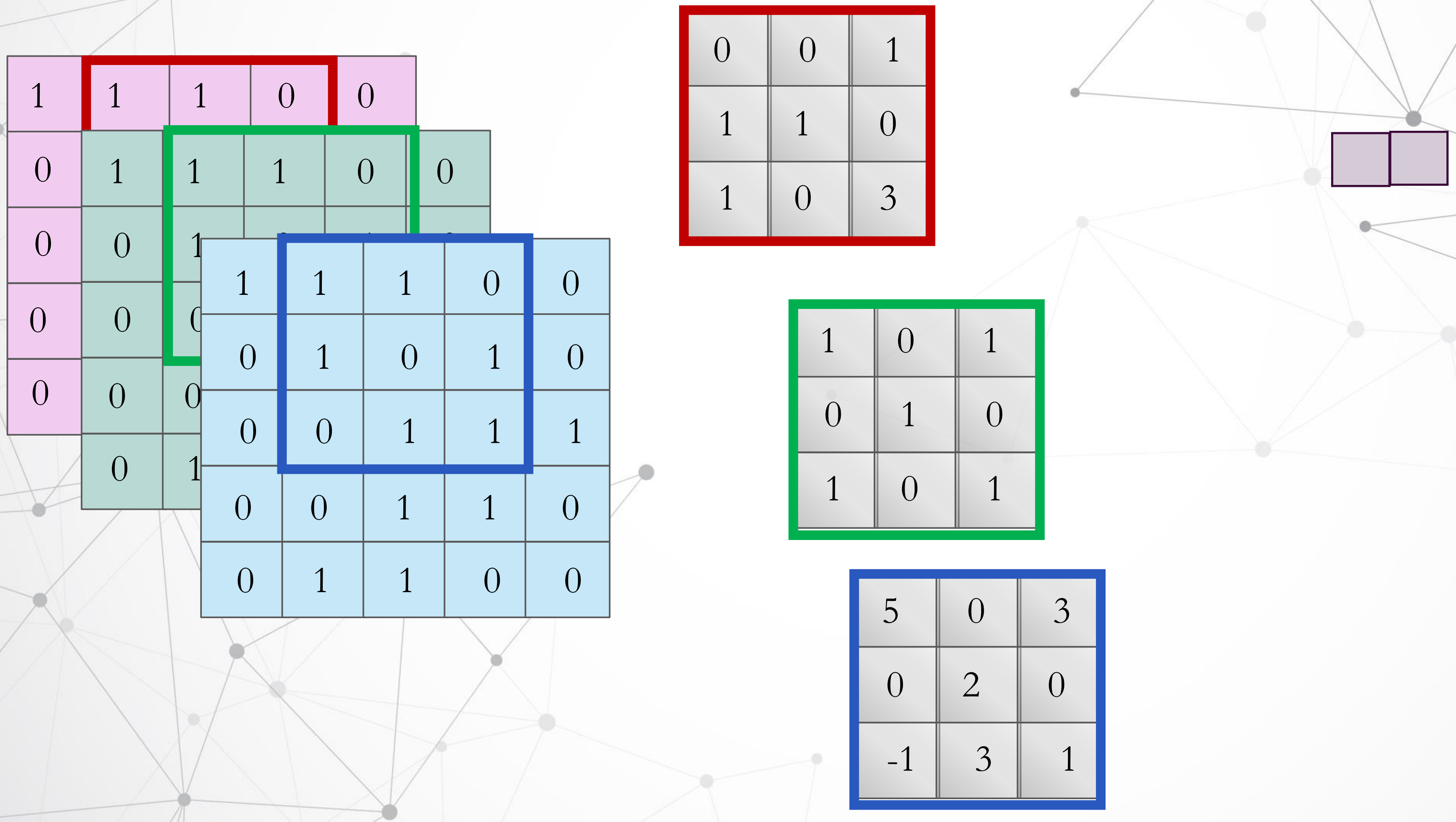
```
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

Filter



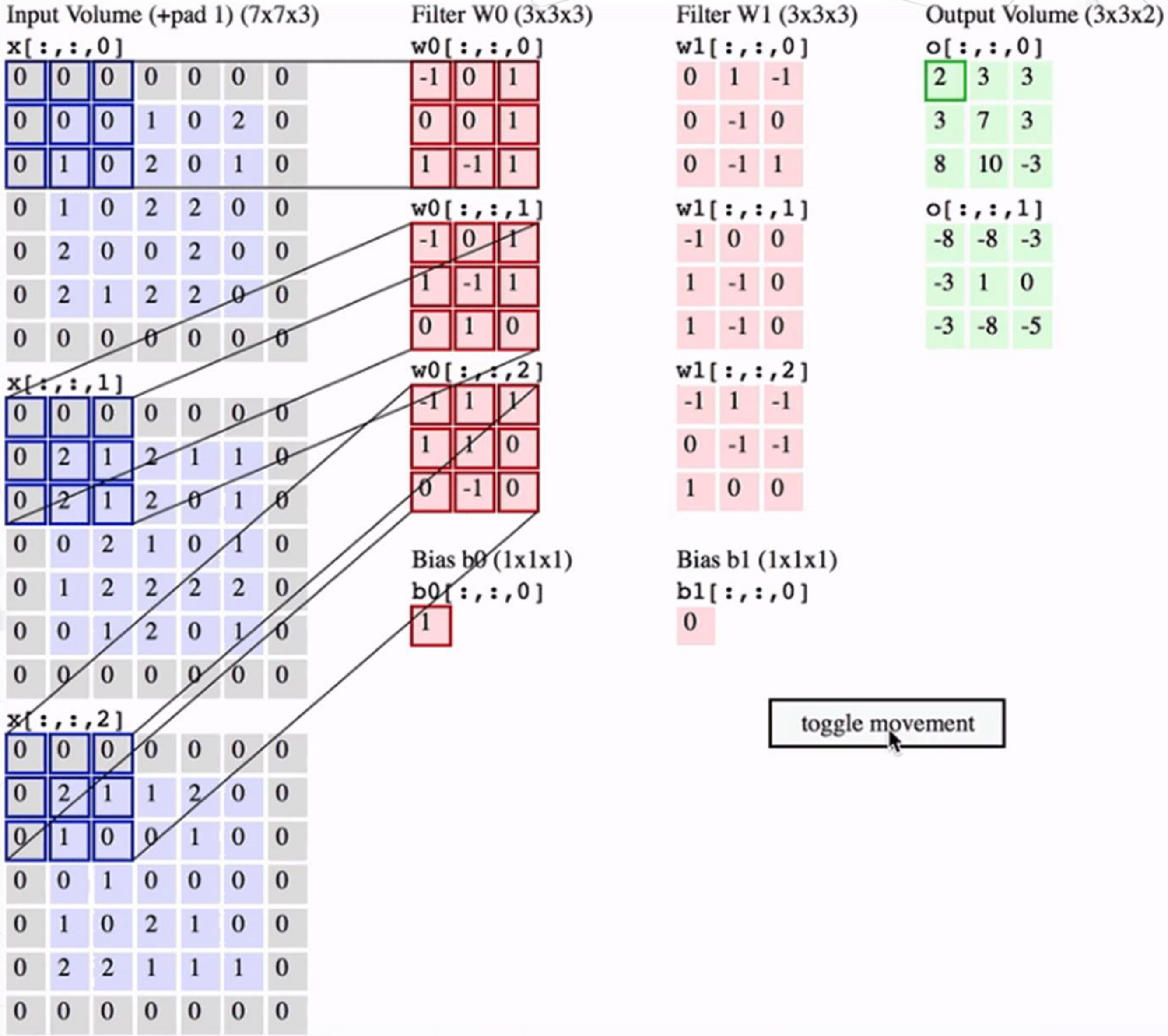
```
tf.keras.layers.Conv2D(filters = 64, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

Filter



```python
tf.keras.layers.ZeroPadding2D(padding=(1, 1))
```

```python
tf.keras.layers.Conv2D(filters = 2, kernel_size = (3, 3), padding='valid', strides=(2, 2))
```



Animation Source:
[Stanford CS231n]()

256 x 256 x 3

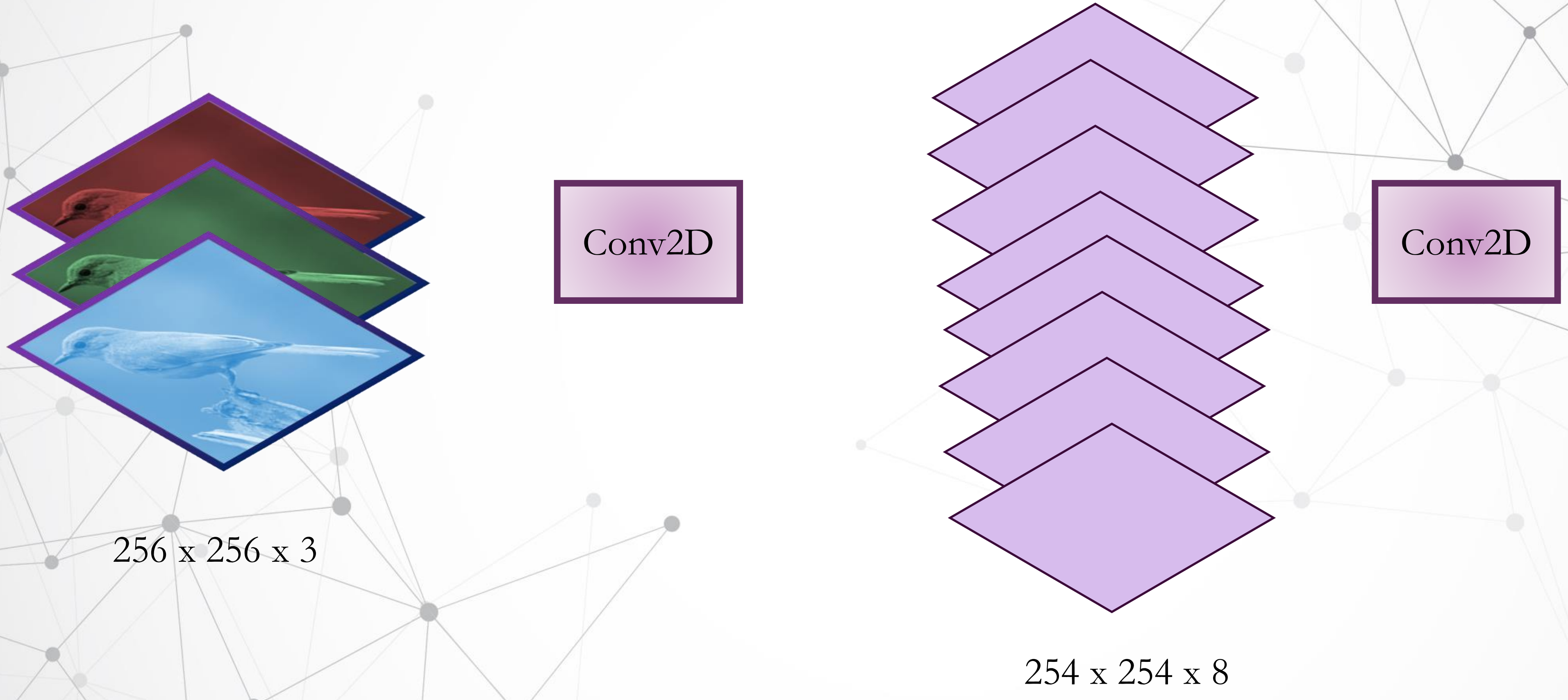
Conv2D

Conv2D

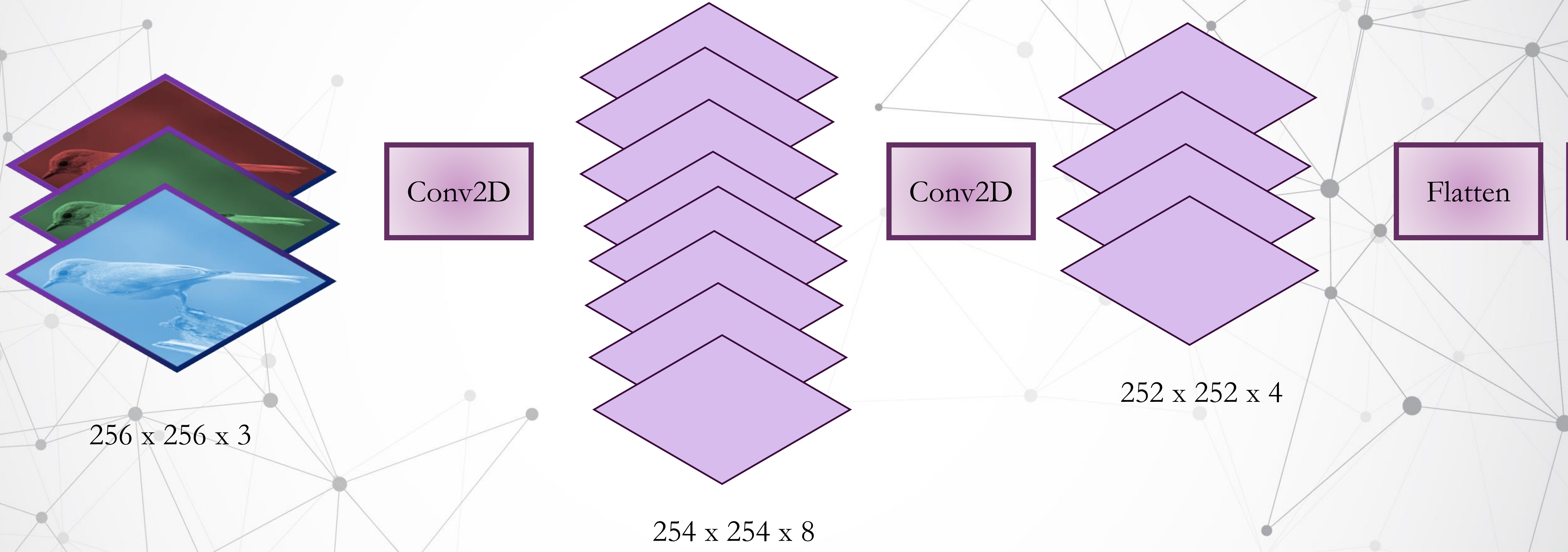
Flatten

DNN

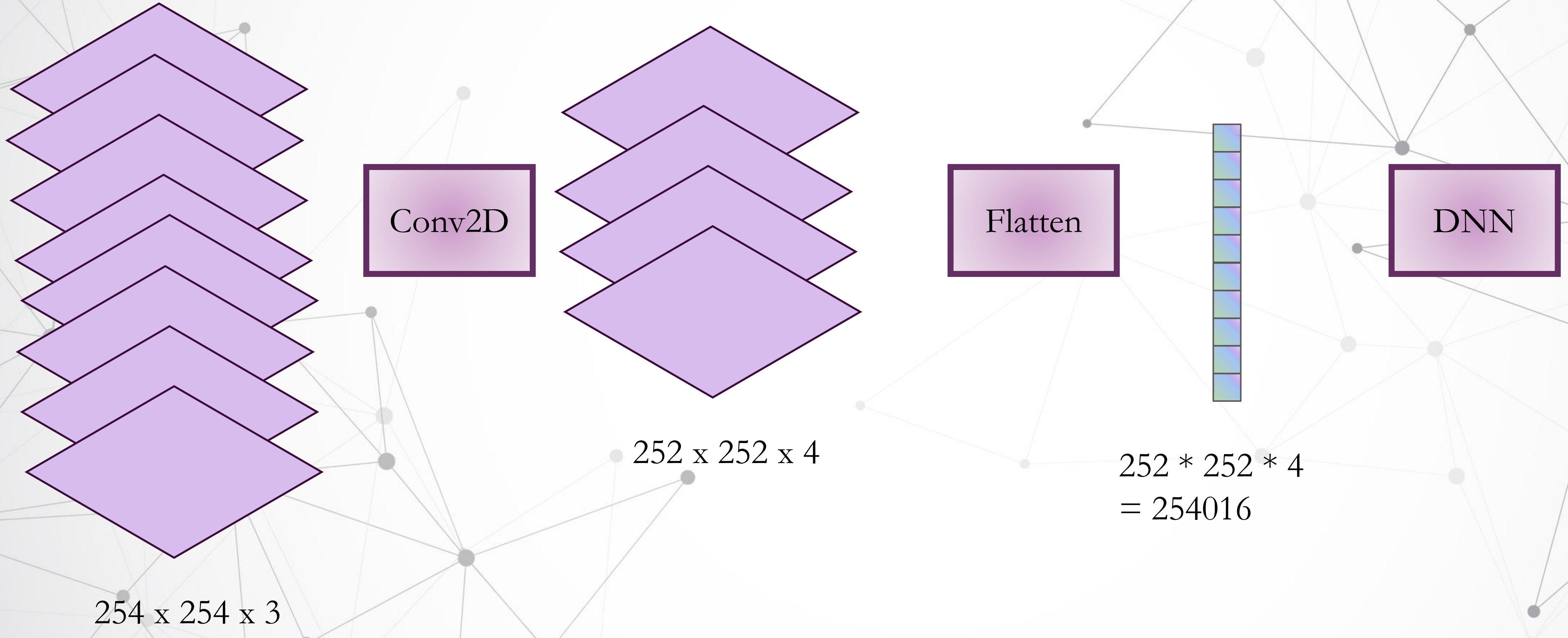
```
tf.keras.layers.Conv2D(filters = 8, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```



```
tf.keras.layers.Conv2D(filters = 4, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```

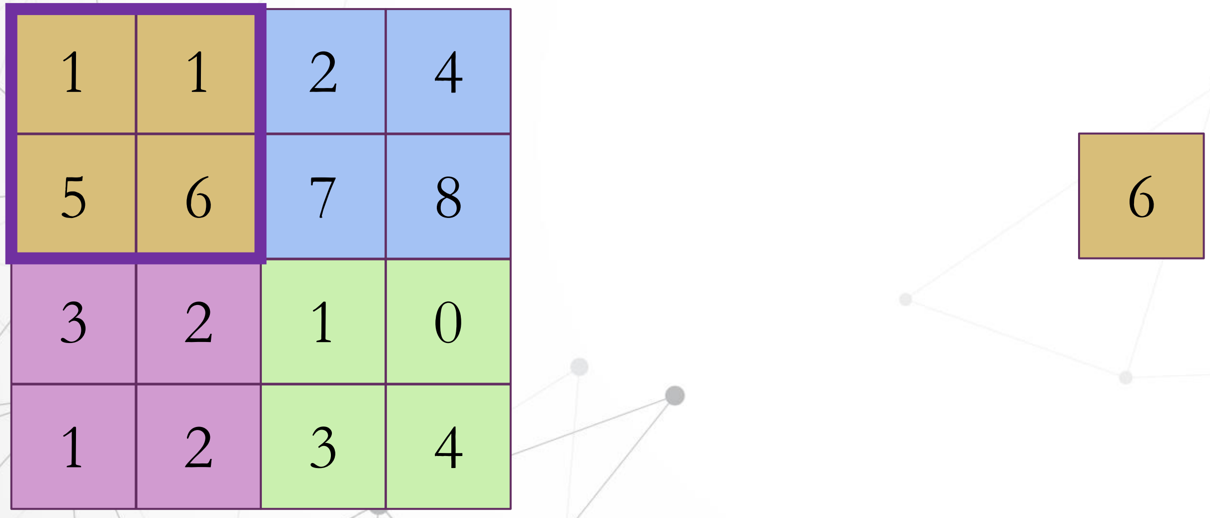


```
tf.keras.layers.Conv2D(filters = 4, kernel_size = (3, 3), padding='valid', strides=(1, 1))
```



Pooling

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



Pooling

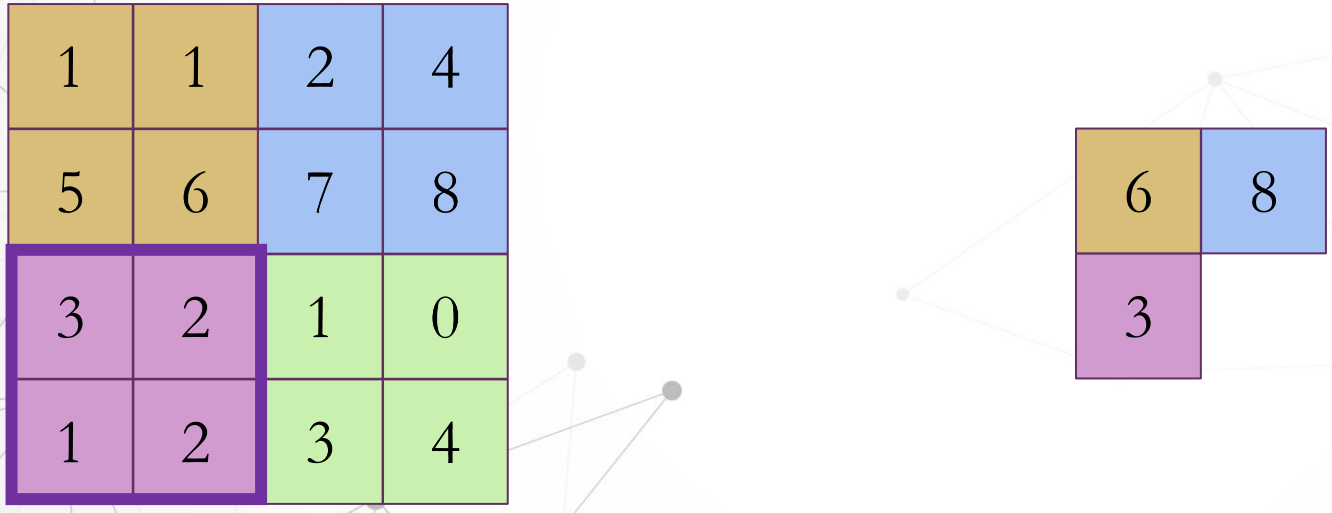
```
tf.keras.layers.MaxPooling2D(pool_size = (2, 2))
```

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8

Pooling

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



Pooling

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

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Pooling

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

512,512,3

256,256,3

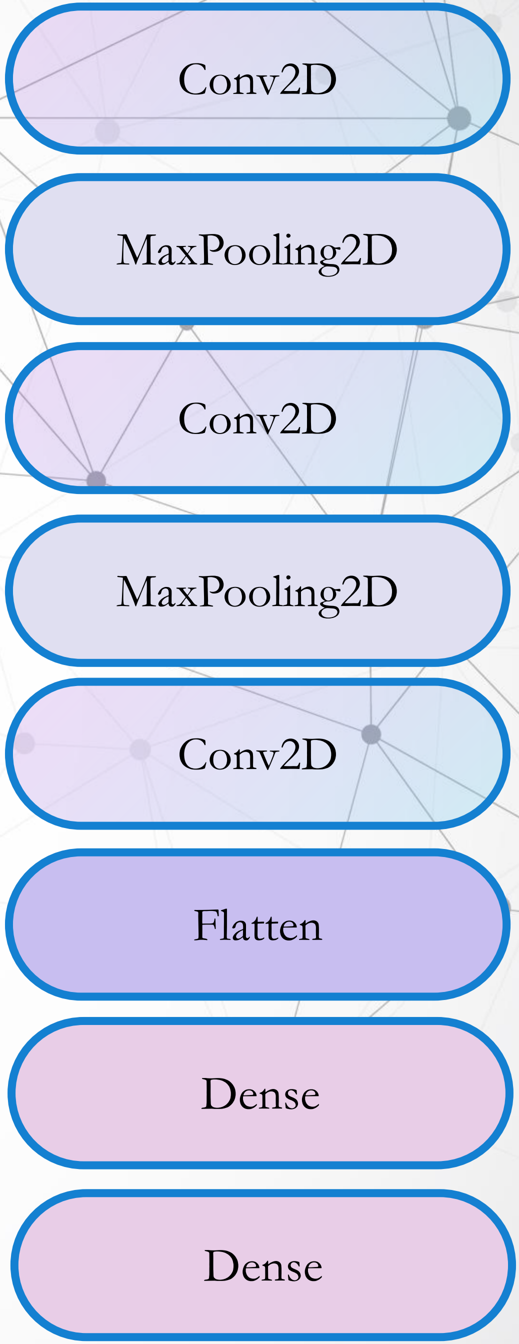
128,128,3

64,64,3

32,32,3

Putting it all together

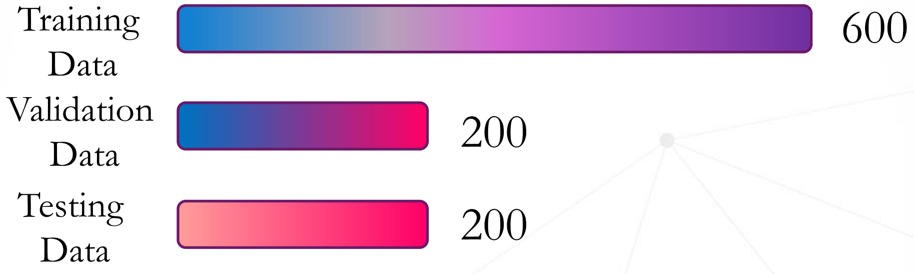
```
tf.keras.layers.Input(shape=(32, 32, 3)),
tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='valid'),
tf.keras.layers.MaxPooling2D((2, 2)),
tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='valid'),
tf.keras.layers.MaxPooling2D((2, 2)),
tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='valid'),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax'),
```



DNN vs CNN

- Less Parameters
- CNNs are more robust to changes in Image

Data Augmentations



Data Augmentations

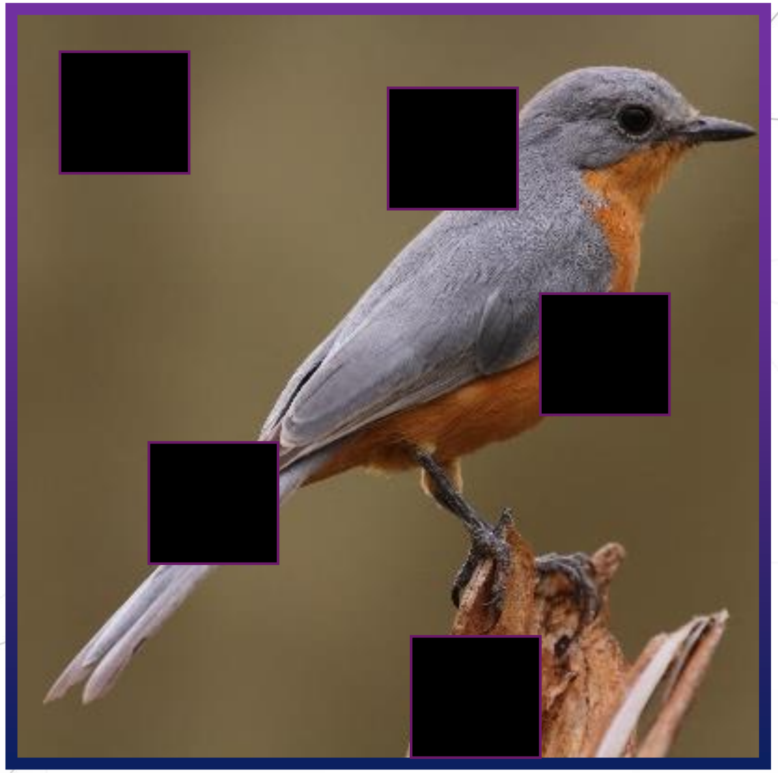




Why Augmentation?

- Increase Data
- More Generalizability
- Challenging cases
- Stopping over-reliance on features

Cutout



Mixup

Mixup

1 Car

0 Bus

0 CNG

0

1

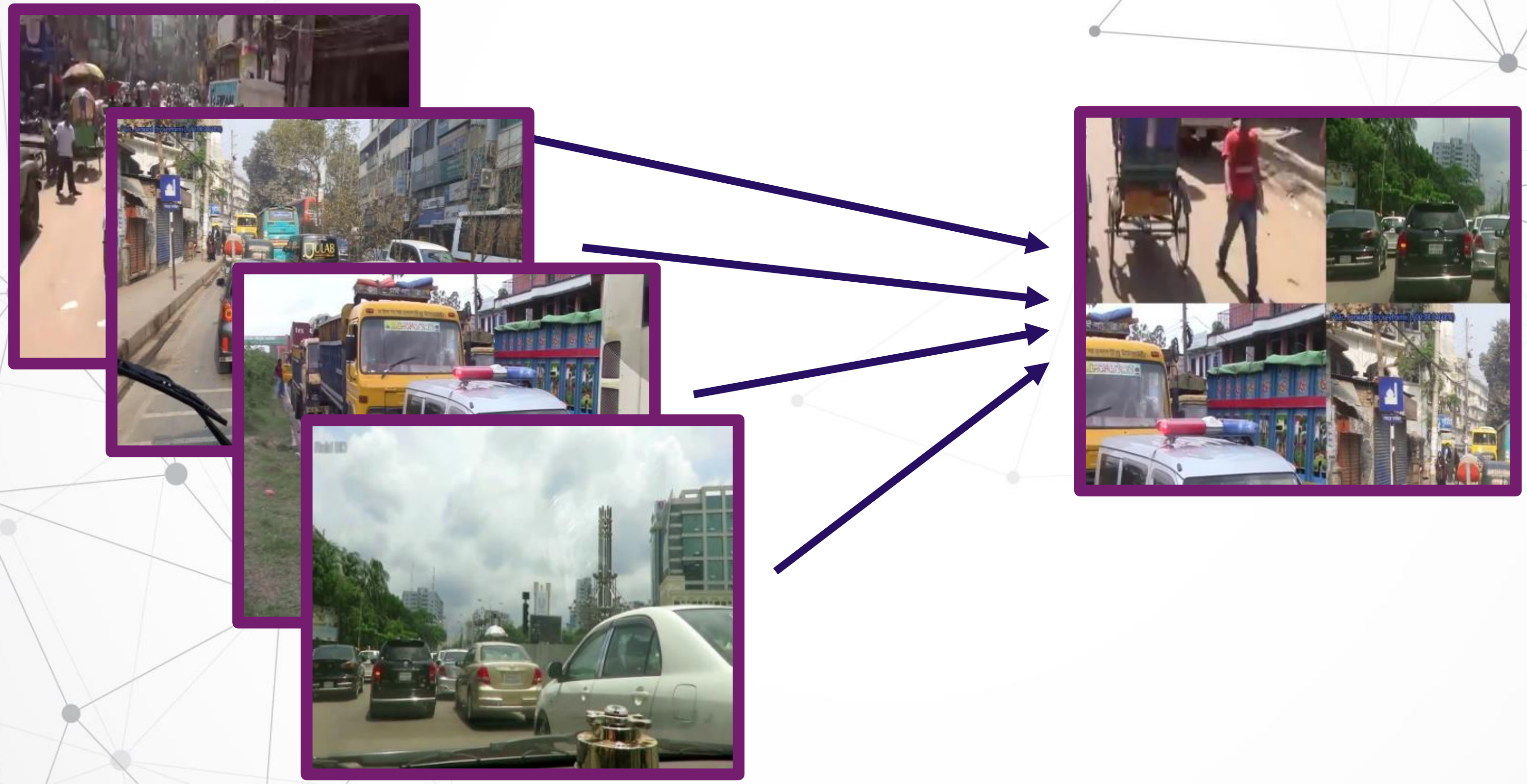
0

0.5

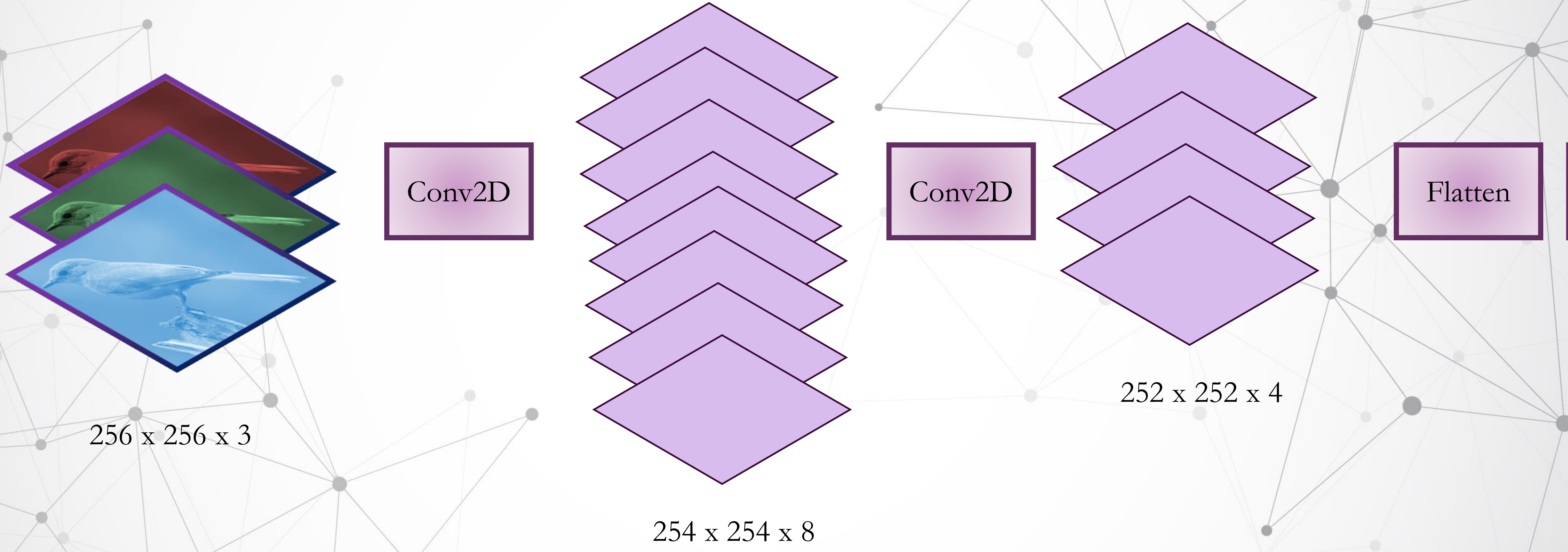
0.5

0

Mosaic

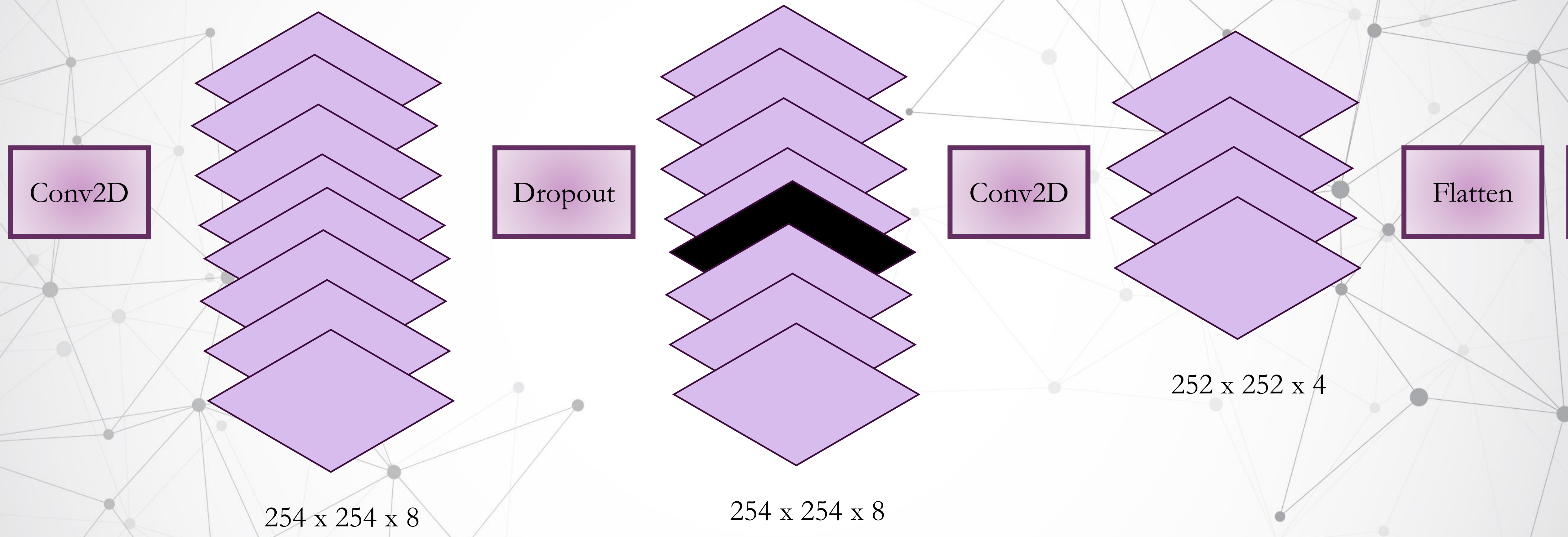


Dropout



Dropout

```
tf.keras.layers.Dropout(rate=.2)
```



Any Question?