Practical Machine Learning

Additional Layers

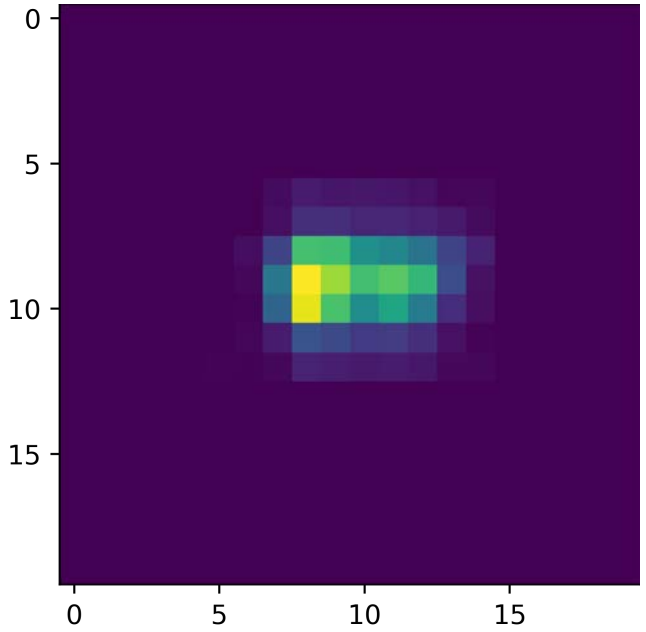
Sven Mayer

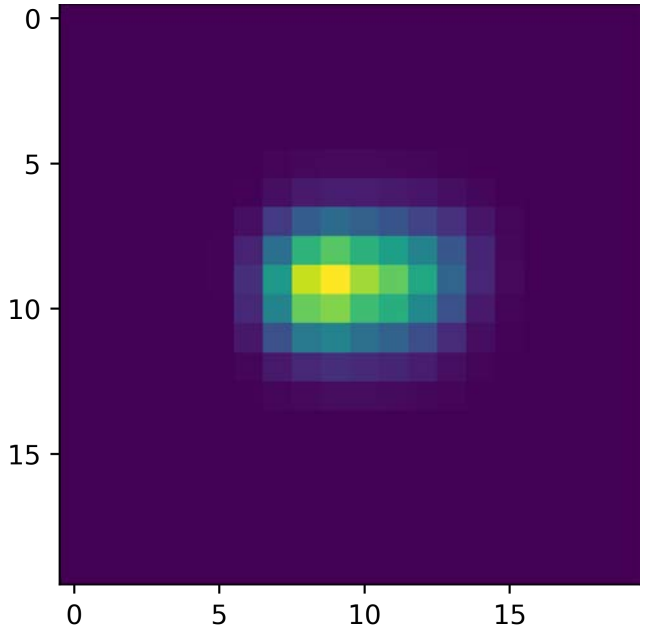
Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN)

Filter

$$filter = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$$

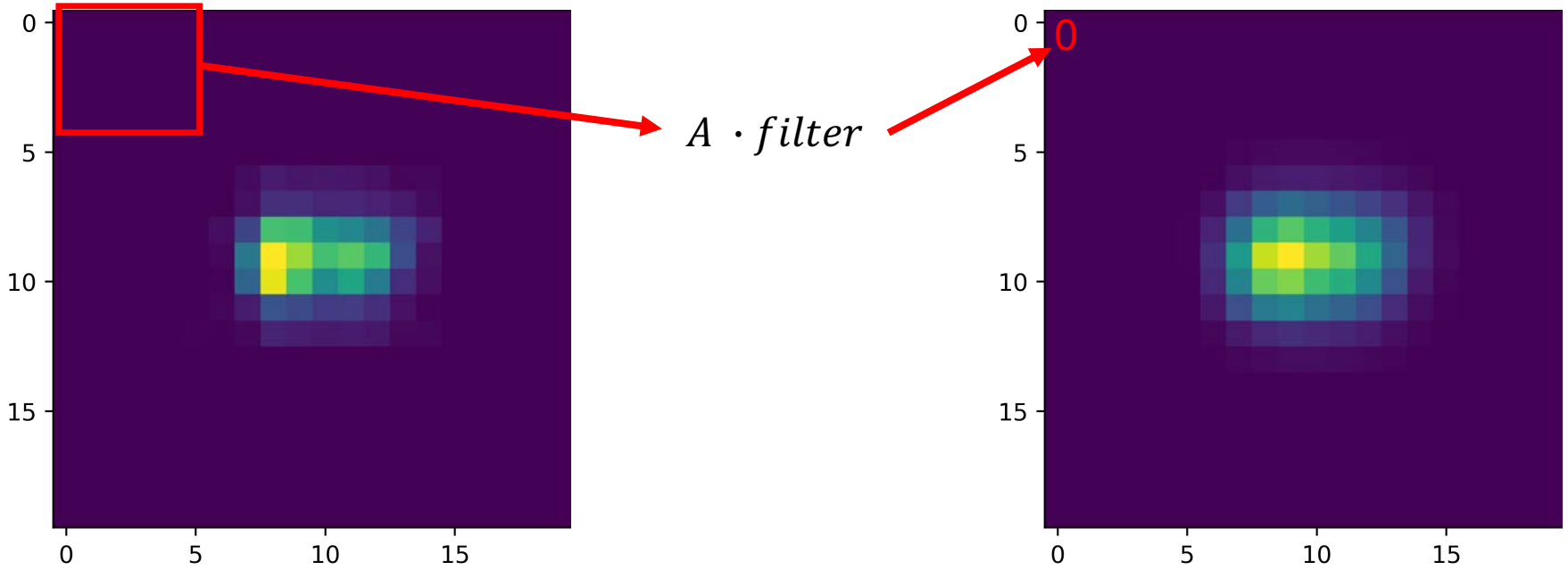




Convolutional Neural Network (CNN)

Process

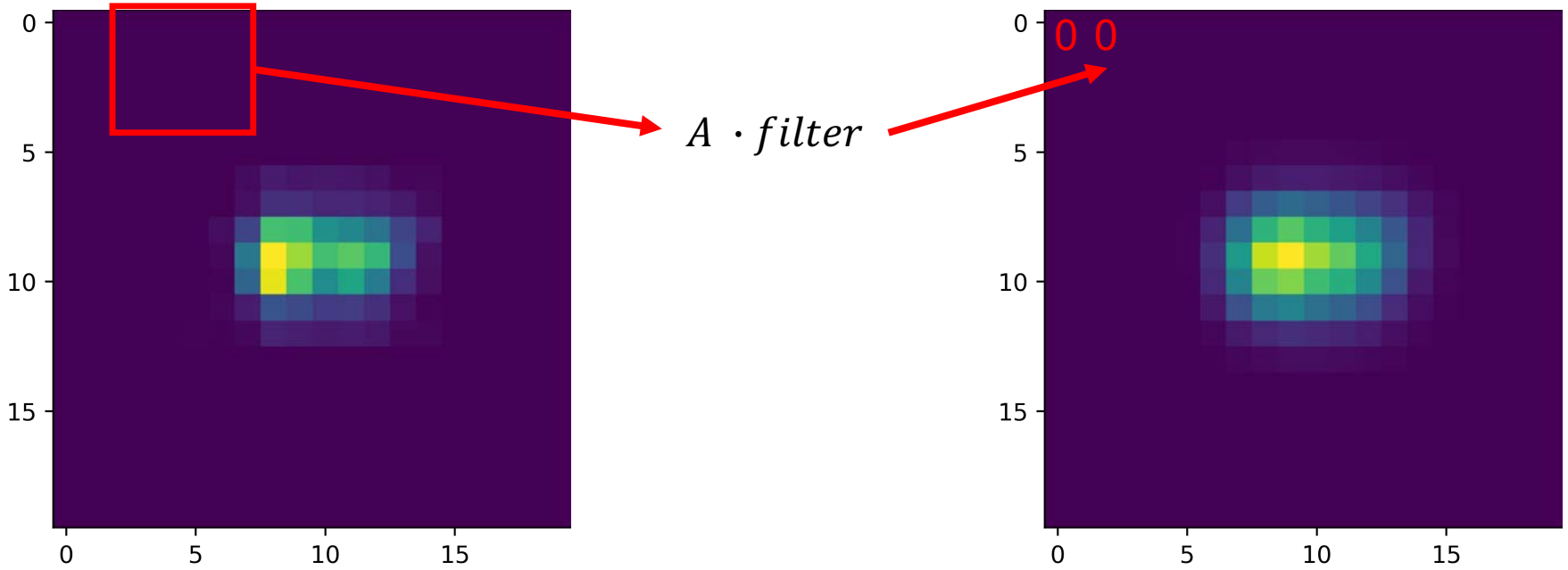
$$filter = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$$



Convolutional Neural Network (CNN)

Process

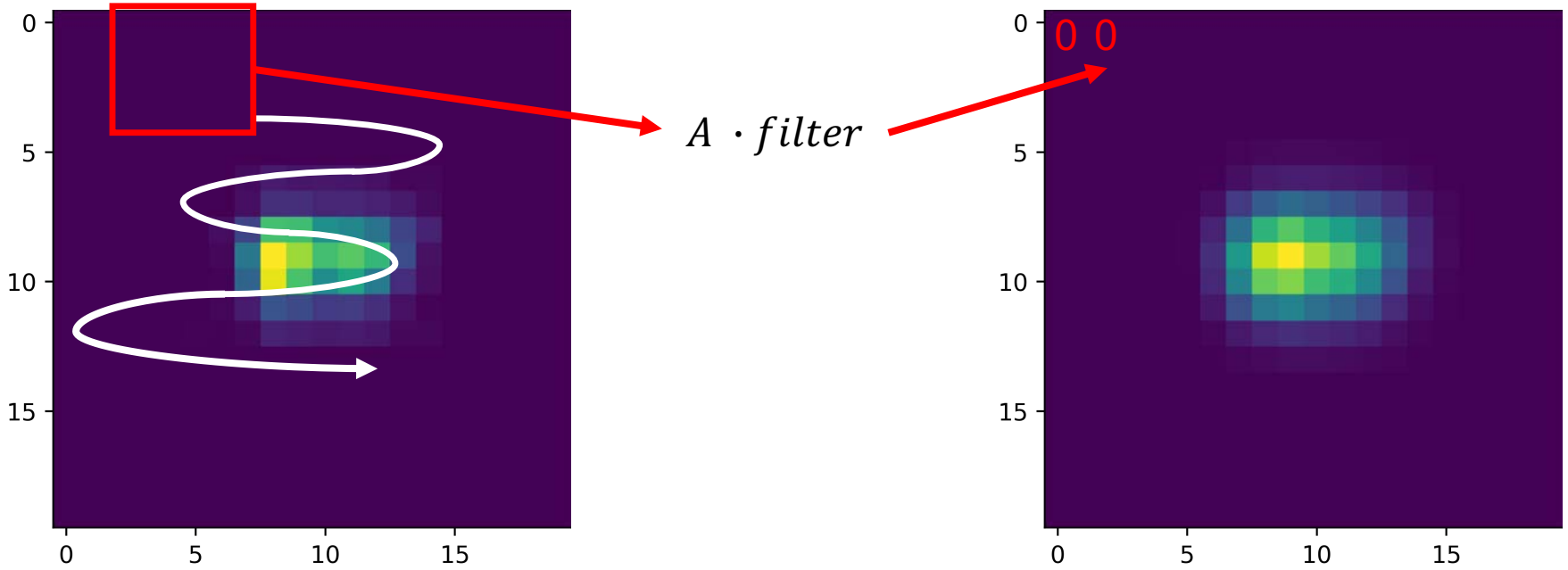
$$filter = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$$



Convolutional Neural Network (CNN)

Process

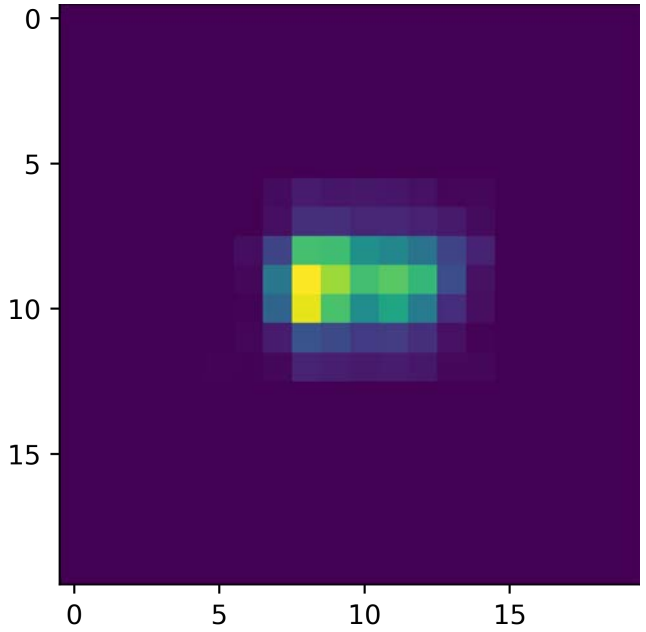
$$filter = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$$

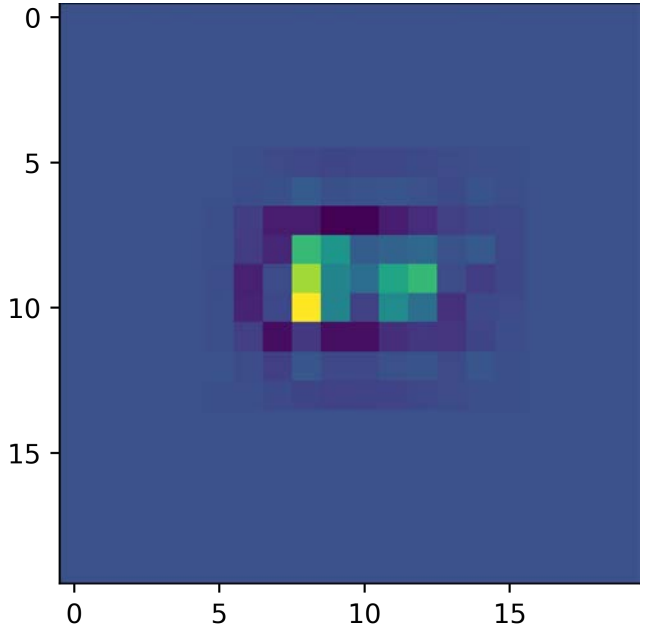


Convolutional Neural Network (CNN)

Filter – Edge Detection Example

$$filter = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$





Convolutional Neural Network (CNN)

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

- Filter count
- Kernel size e.g. (3x3)
- Step length (strides) e.g. (1,1)
- Padding

$$filter = \begin{bmatrix} w_{0,0} & \cdots & w_{0,m} \\ \vdots & \ddots & \vdots \\ w_{n,0} & \cdots & w_{n,m} \end{bmatrix}$$

During the learning the model learns the w's of each filter.

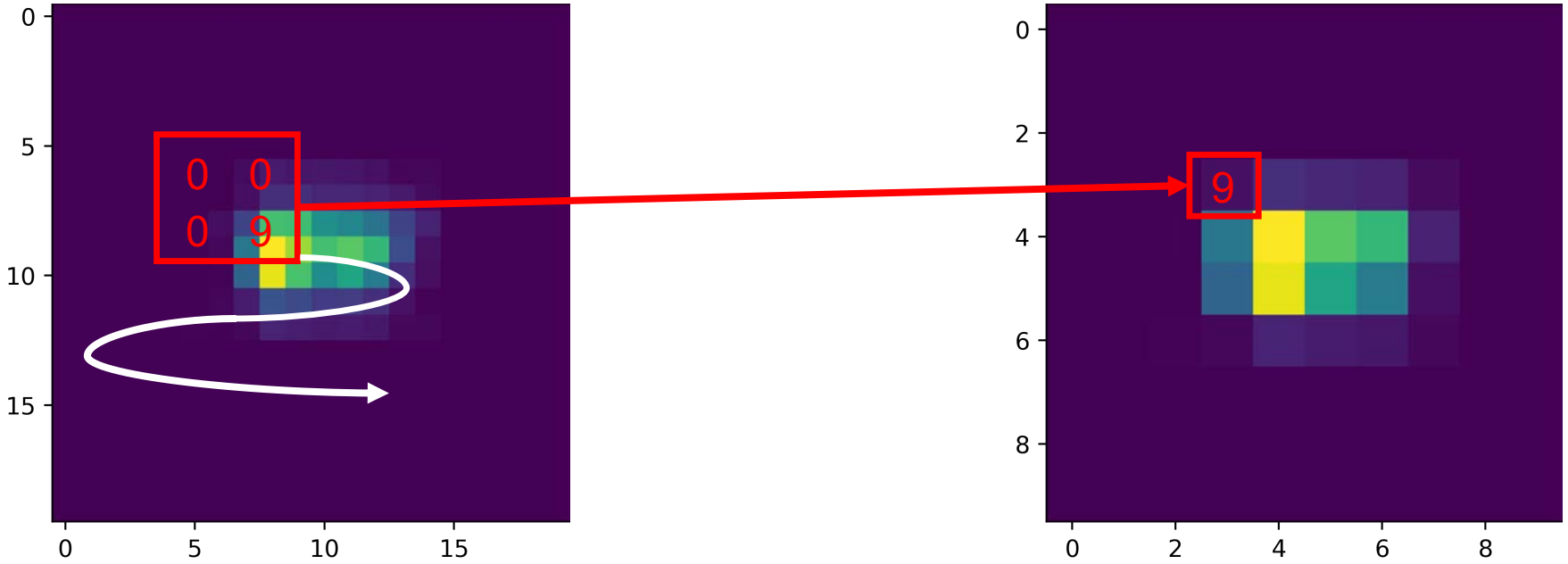
Constraints:

- Shaped Input
- Shaped Output

Pooling Layer

Max Pooling Layer

- Kernel size e.g. (2x2)
- Max value within the filter

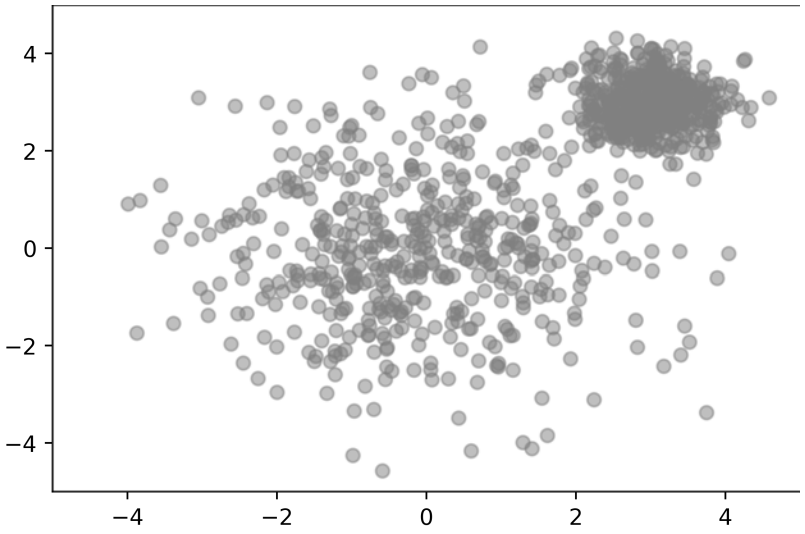


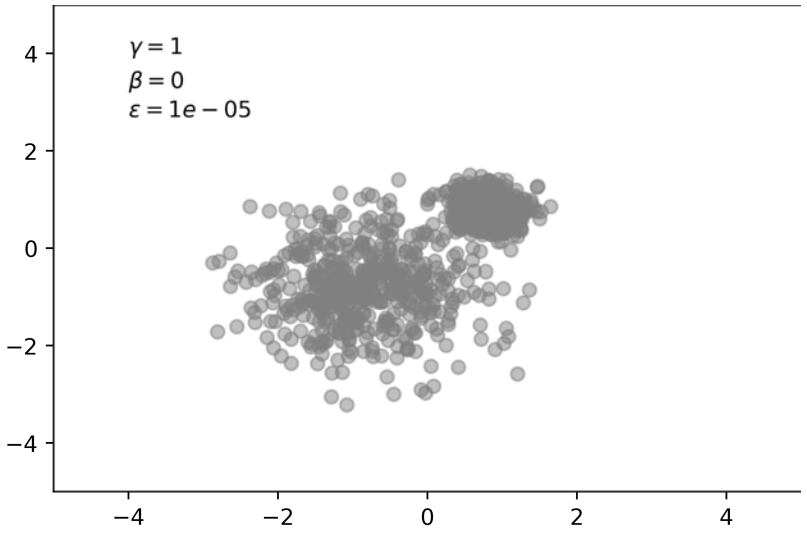
Batch Normalization

Batch Normalization

- tf.keras.layers.BatchNormalization
- Improves training
- Often speeds up training

$$x_{t+1} = \frac{x_t - mean(x_t)}{\sqrt{std(x_t) + \varepsilon}} * \gamma + \beta$$



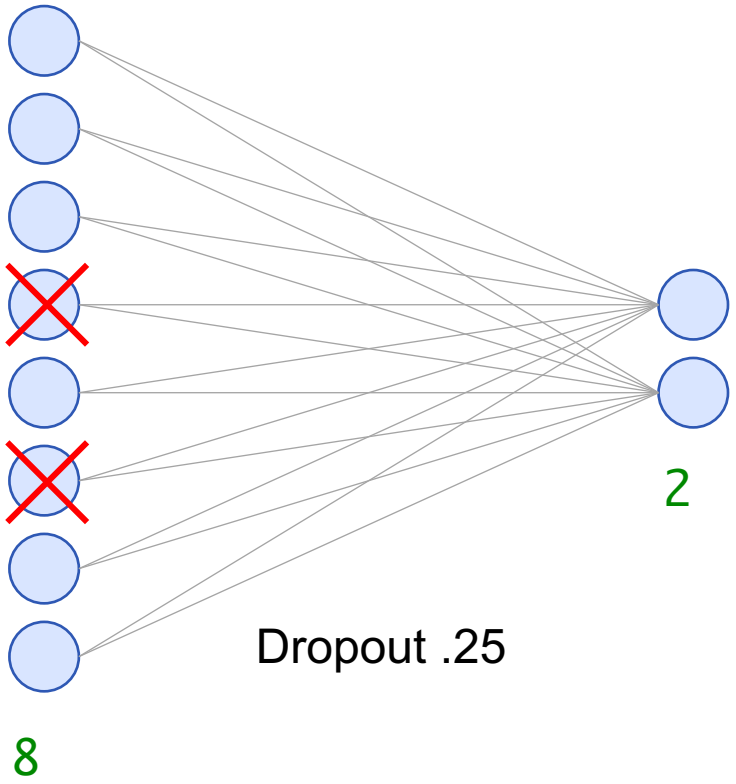


Ioffe and Szegedy 2015 https://arxiv.org/abs/1502.03167

Dropout

Dropout

- Reduces overfitting



Dropout layer reduces the likelihood to relay on one single "feature" as sometimes its not active.

What can we do now?

Example Model

```
model = Sequential()

model.add(InputLayer((28,28,1), name = "InputLayer"))

model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu',
padding='same'))

model.add(MaxPool2D())

model.add(Dropout(rate=0.25))

model.add(Conv2D(64, kernel_size=(3,3), activation='relu', padding='same'))

model.add(MaxPool2D())

model.add(BatchNormalization())

model.add(Flatten())

model.add(Dense(128, name = "HiddenLayer1", activation='relu'))

model.add(Dense(64, name = "HiddenLayer2", activation='relu'))

model.add(Dense(2, name = "OutputLayer", activation=softmax'))
```

new layer `Flatten` needed after `Conv2D` to use a `Dense` next – similar to reshape

Conclusion

Layers

- Convolutional Neural Network (CNN)
- Max Pooling
- Batch Normalization
- Dropout

License