



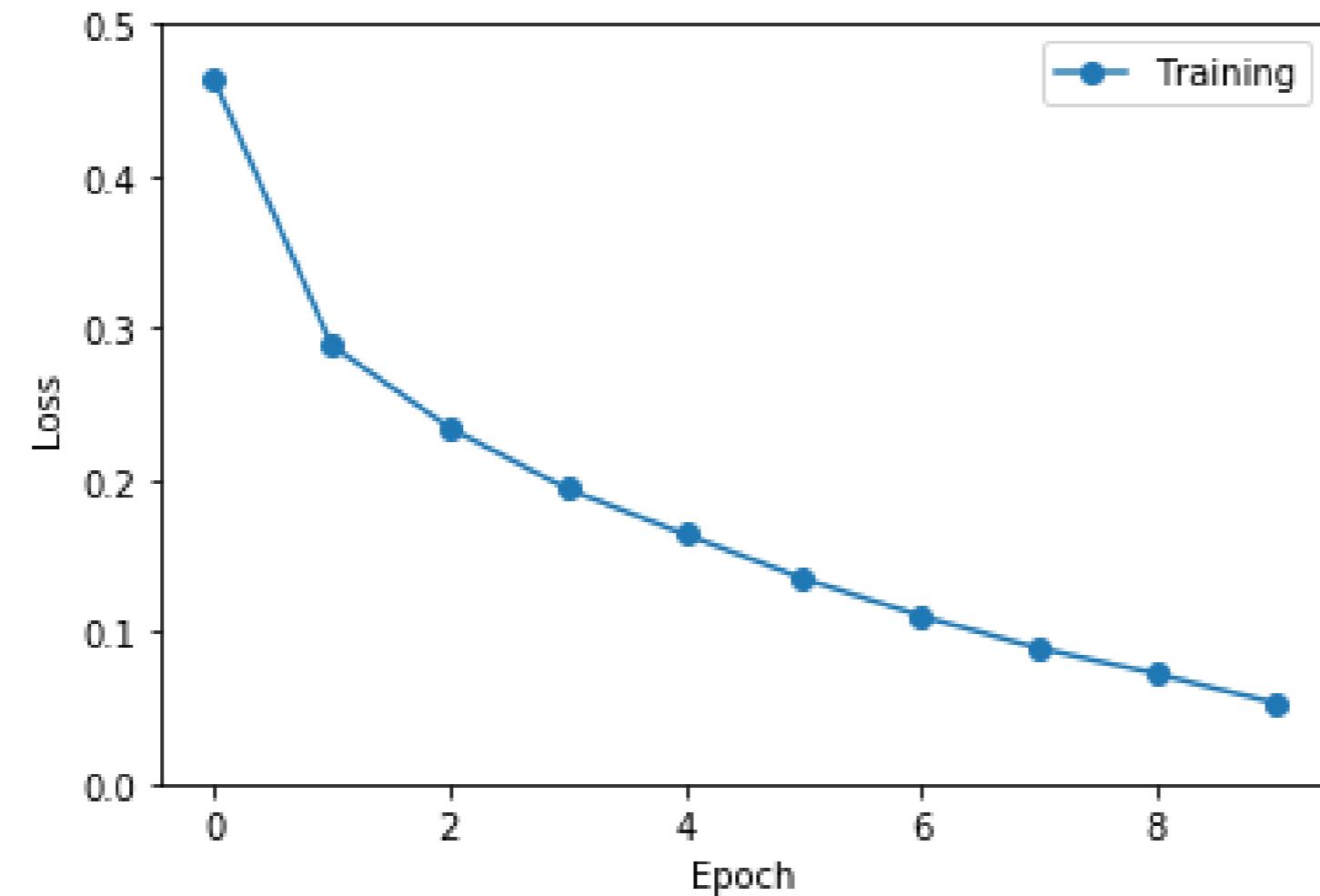
## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

# Tracking learning

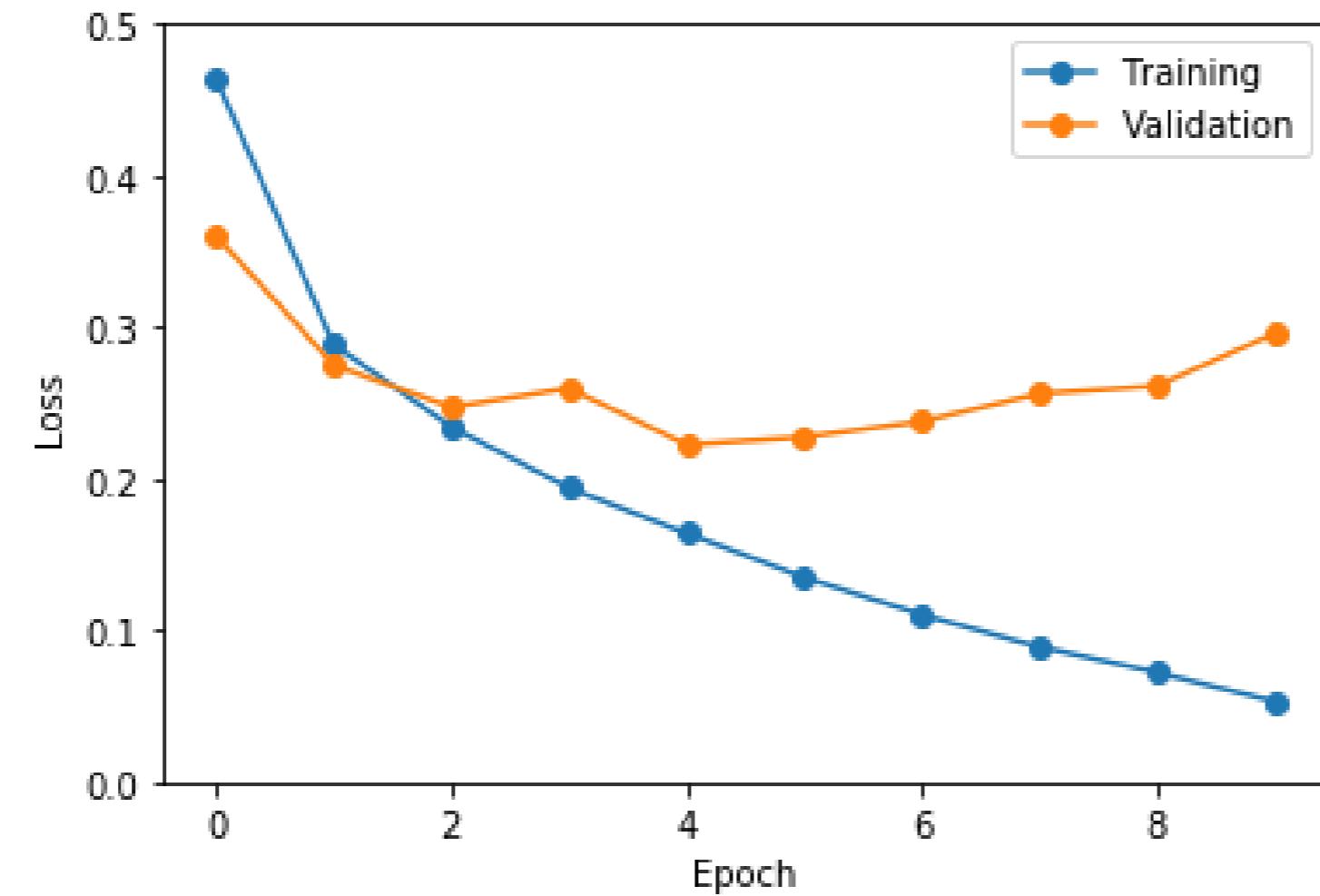
Ariel Rokem

Senior Data Scientist, University of Washington

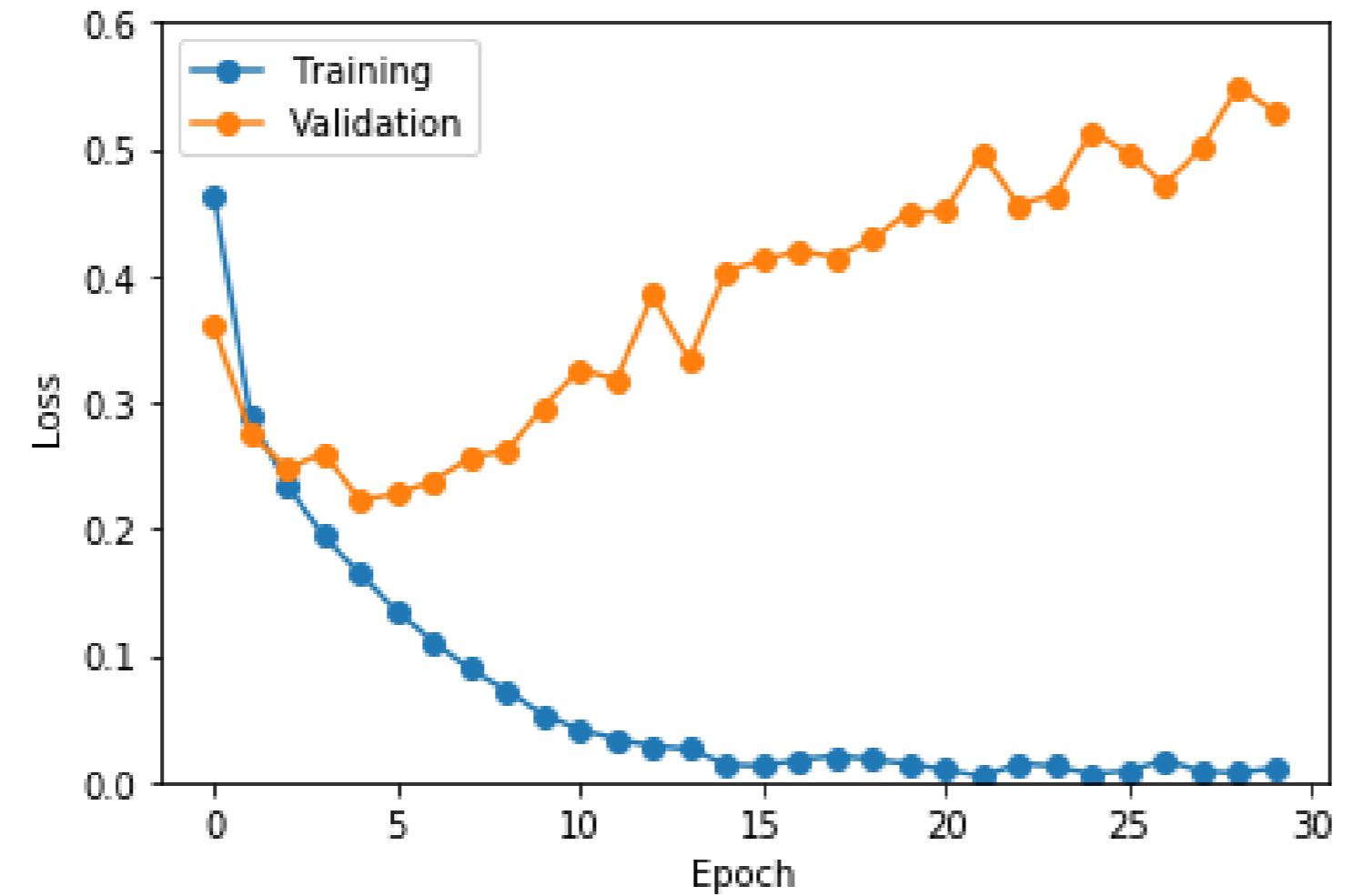
# Learning curves: training



# Learning curves: validation

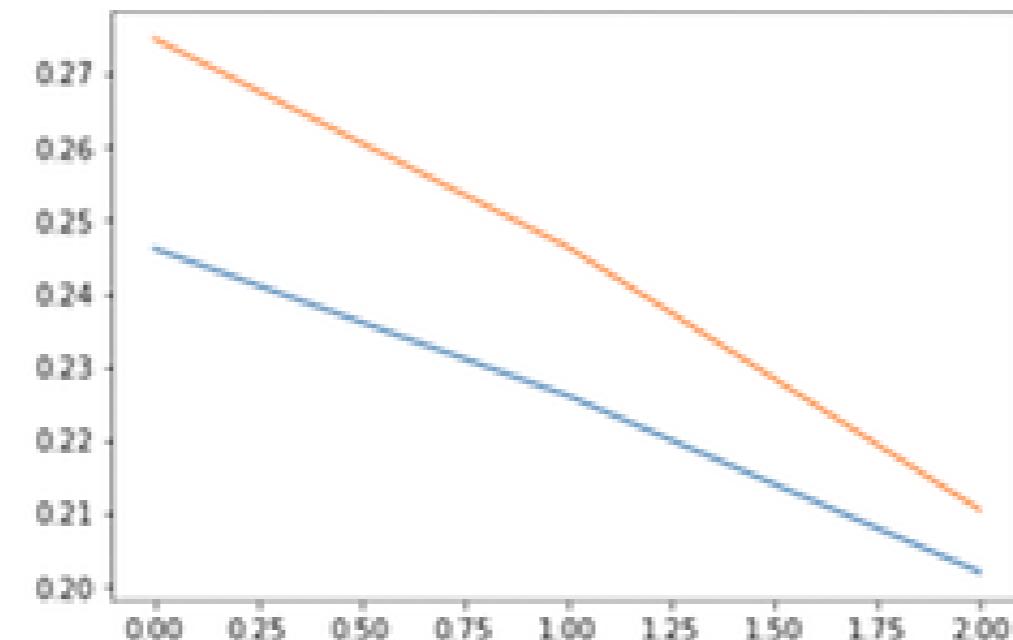


# Learning curves: overfitting



# Plotting training curves

```
training = model.fit(train_data, train_labels,  
                      epochs=3, validation_split=0.2)  
  
import matplotlib.pyplot as plt  
plt.plot(training.history['loss'])  
  
plt.plot(training.history['val_loss'])  
  
plt.show()
```



# Storing the optimal parameters

```
from keras.callbacks import ModelCheckpoint

# This checkpoint object will store the model parameters
# in the file "weights.hdf5"
checkpoint = ModelCheckpoint('weights.hdf5', monitor='val_loss',
                             save_best_only=True)

# Store in a list to be used during training
callbacks_list = [checkpoint]

# Fit the model on a training set, using the checkpoint as a callback
model.fit(train_data, train_labels, validation_split=0.2, epochs=3,
           callbacks=callbacks_list)
```

# Loading stored parameters

```
model.load_weights('weights.hdf5')  
model.predict_classes(test_data)  
array([2, 2, 1, 2, 0, 1, 0, 1, 2, 0])
```



## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

**Let's practice!**



## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

# Neural network regularization

Ariel Rokem

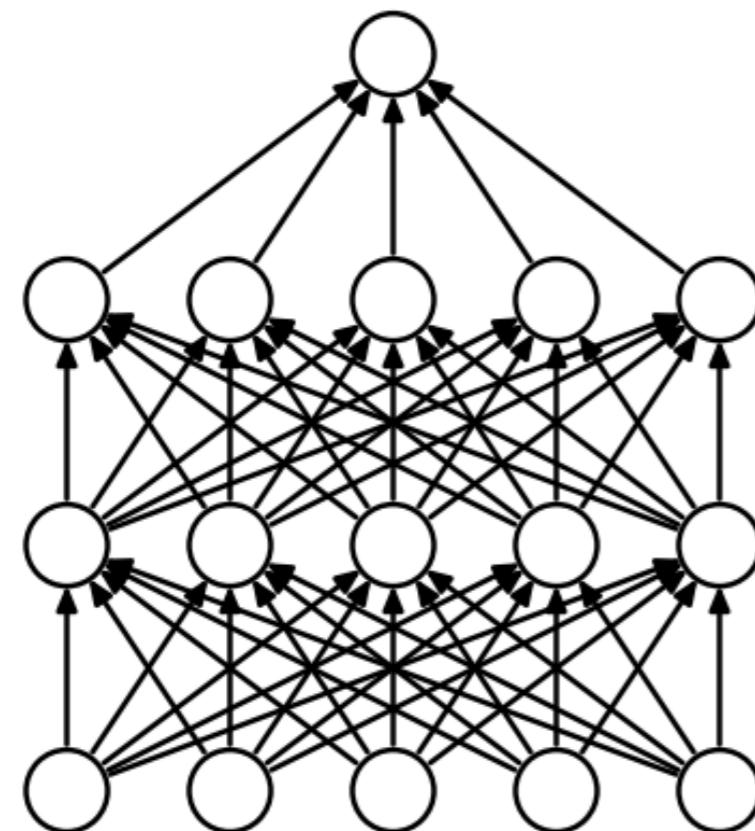
Senior Data Scientist, University of Washington

# Dropout

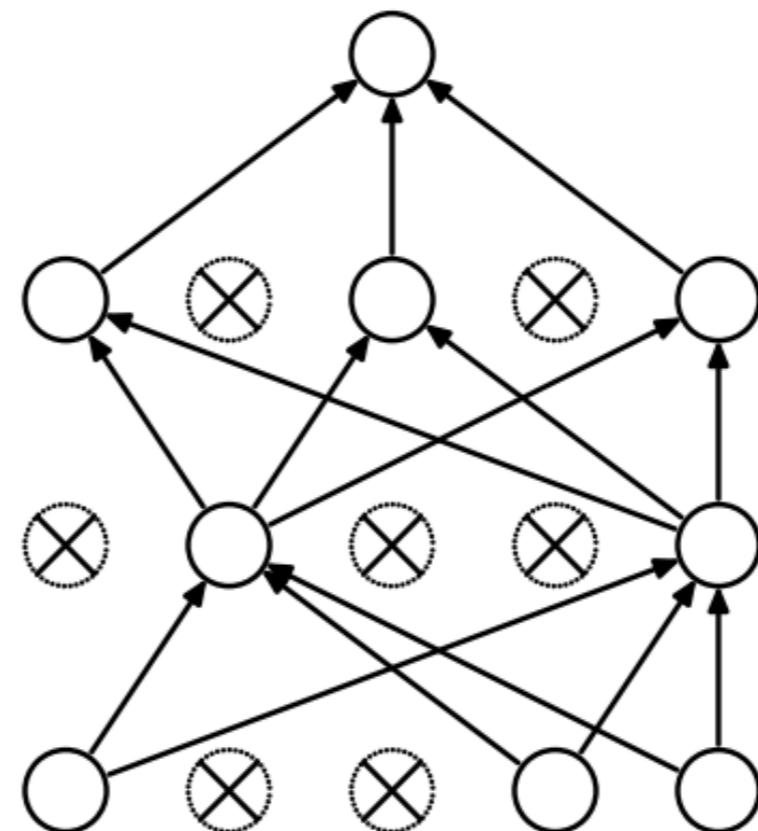
In each learning step:

- Select a subset of the units
- Ignore it in the forward pass
- And in the back-propagation of error

# Dropout



(a) Standard Neural Net



(b) After applying dropout.

# Dropout in Keras

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, Dropout

model = Sequential()
model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))

model.add(Dropout(0.25))

model.add(Conv2D(15, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

# Batch normalization

- Rescale the outputs

# Batch Normalization in Keras

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, BatchNormalization

model = Sequential()
model.add(Conv2D(5, kernel_size=3, activation='relu',
                input_shape=(img_rows, img_cols, 1)))

model.add(BatchNormalization())

model.add(Conv2D(15, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

# Be careful when using them together!

The disharmony between dropout and batch normalization



## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

**Let's practice!**



## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

# Interpreting the model

Ariel Rokem

Senior Data Scientist, University of Washington

# Selecting layers

```
model.layers  
[<keras.layers.convolutional.Conv2D at 0x109f10c18>,  
<keras.layers.convolutional.Conv2D at 0x109ec5ba8>,  
<keras.layers.core.Flatten at 0x1221ffcc0>,  
<keras.layers.core.Dense at 0x1221ffef0>]
```

# Getting model weights

```
conv1 = model.layers[0]

weights1 = conv1.get_weights()

len(weights1)

2

kernels1 = weights1[0]

kernels1.shape

(3, 3, 1, 5)

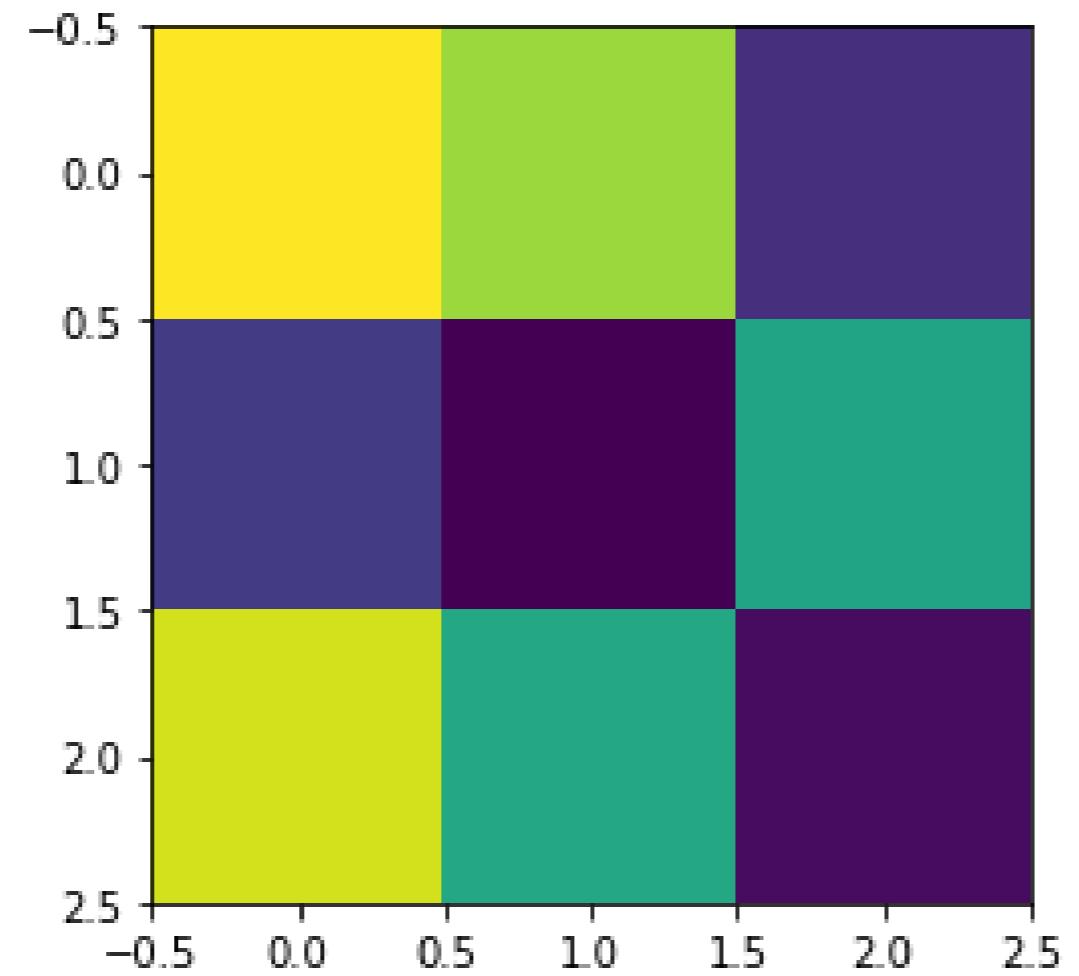
kernels1_1 = kernels1[:, :, 0, 0]

kernels1_1.shape

(3, 3)
```

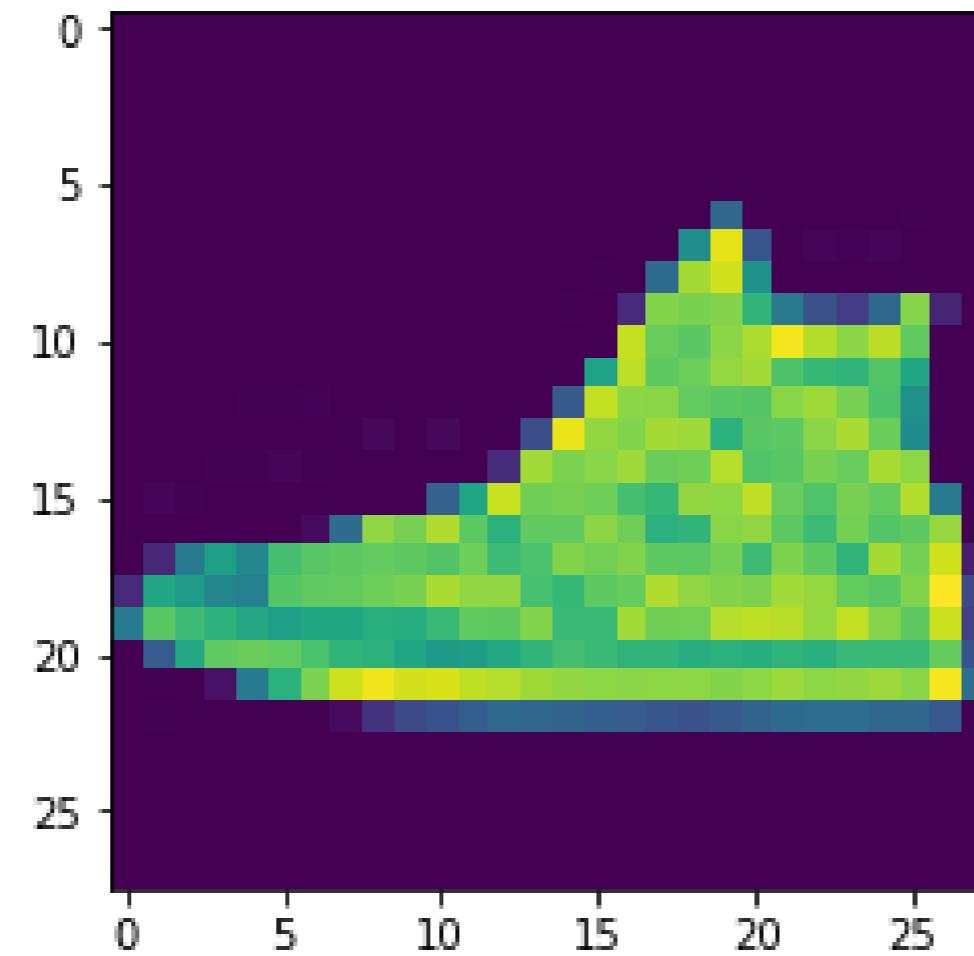
# Visualizing the kernel

```
plt.imshow(kernell_1)
```



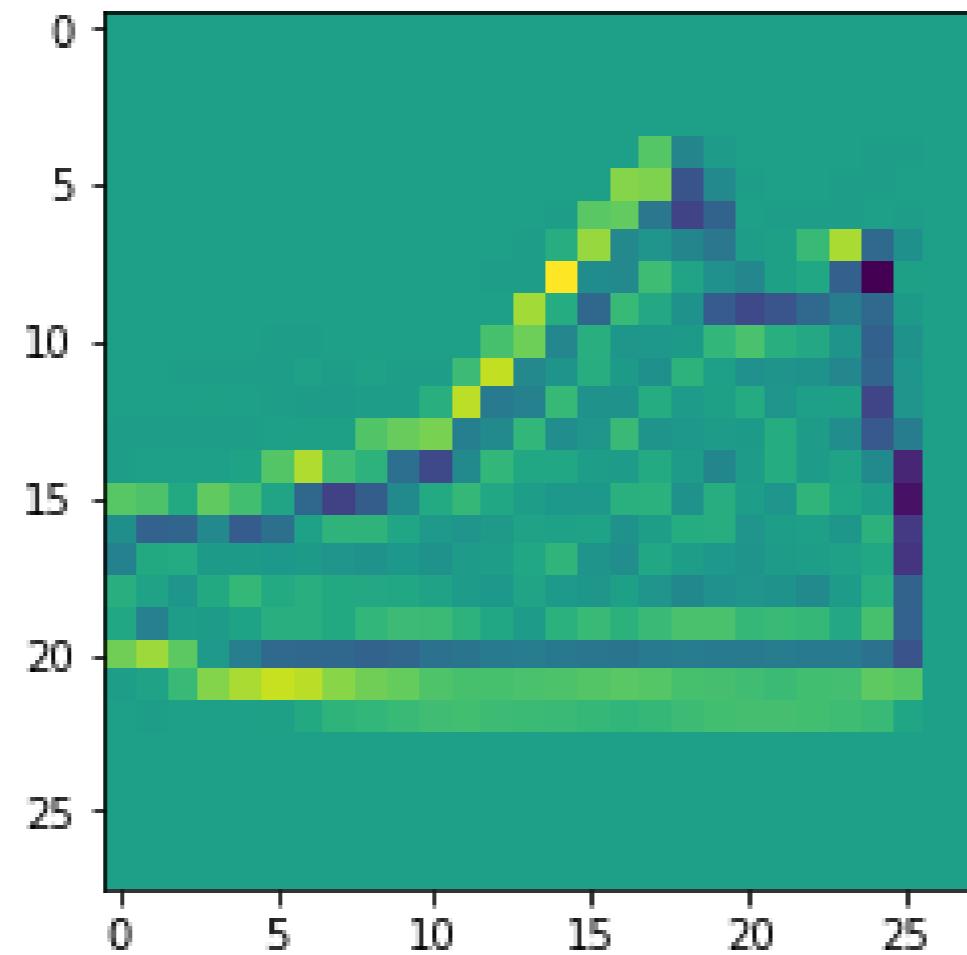
# Visualizing the kernel responses

```
test_image = test_data[3, :, :, 0]  
plt.imshow(test_image)
```



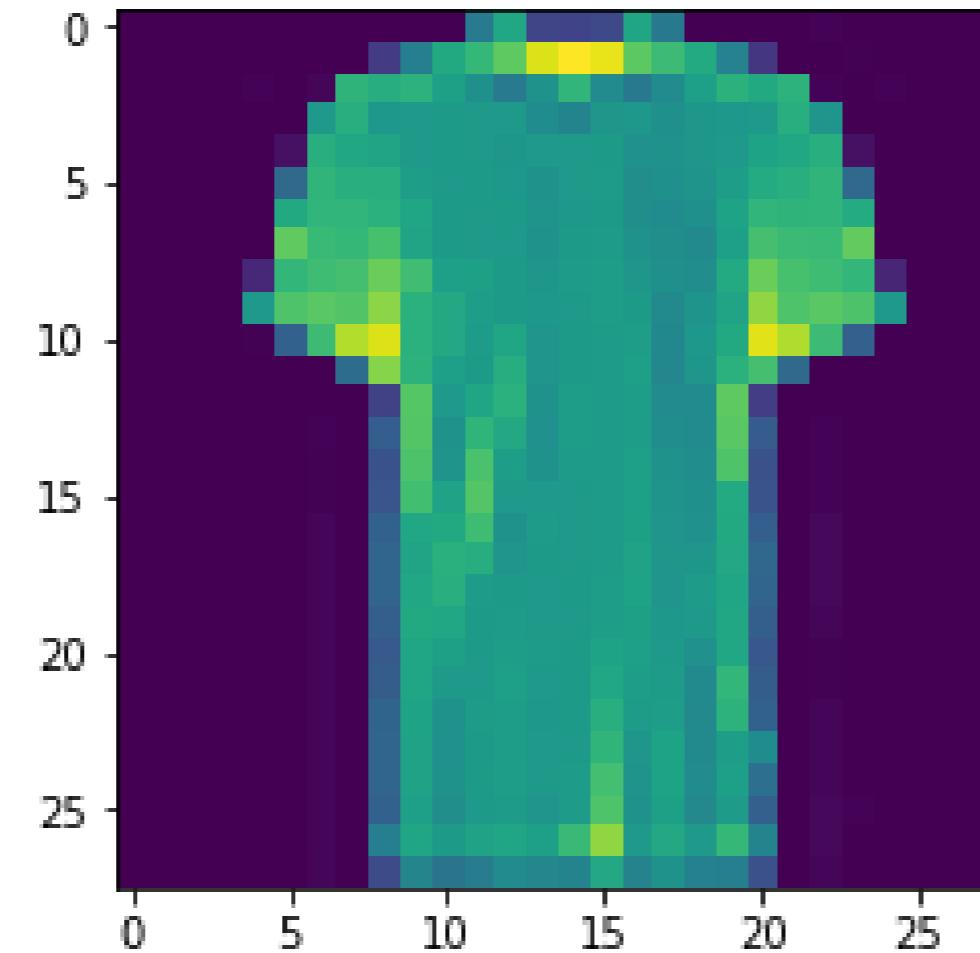
# Visualizing the kernel responses

```
filtered_image = convolution(test_image, kernell_1)  
plt.imshow(filtered_image)
```



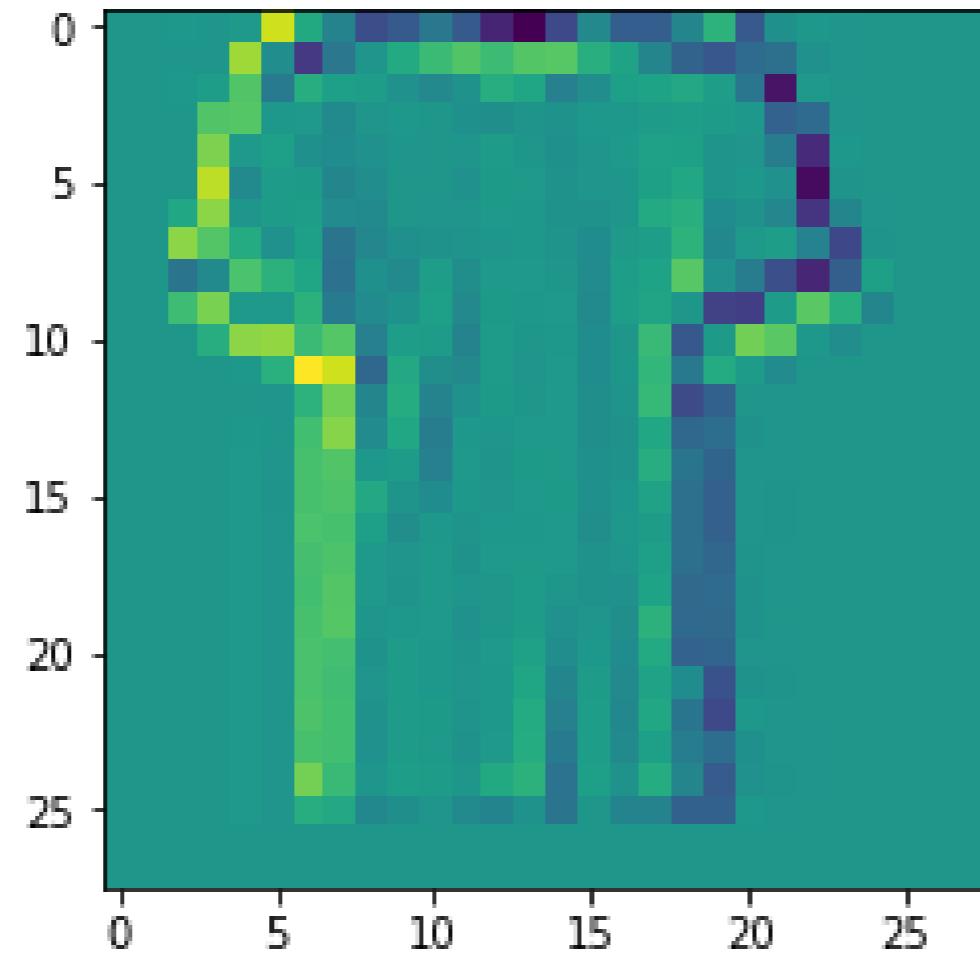
# Visualizing the kernel responses

```
test_image = test_data[4, :, :, 1]  
plt.imshow(test_image)
```



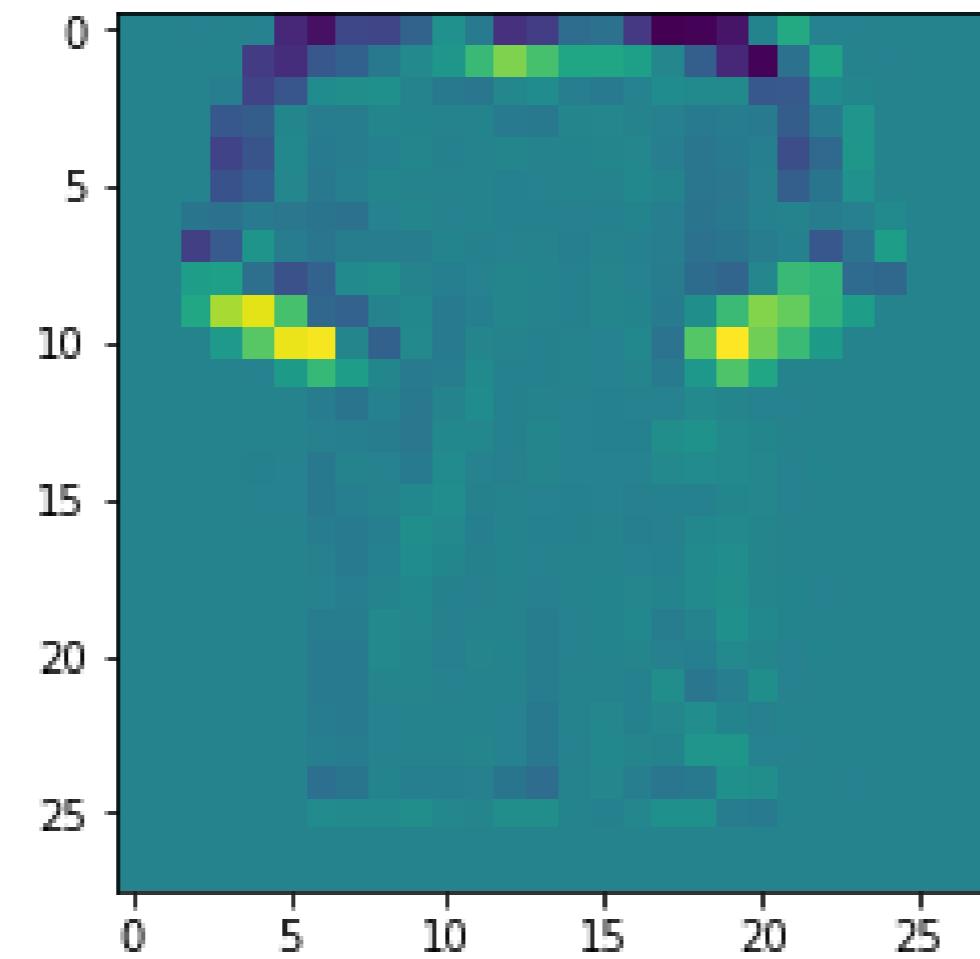
# Visualizing the kernel responses

```
filtered_image = convolution(test_image, kernell_1)  
plt.imshow(filtered_img)
```



# Visualizing the kernel responses

```
kernel1_2 = kernels[:, :, 0, 1]
filtered_image = convolution(test_image, kernel1_2)
plt.imshow(filtered_img)
```





## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

**Let's practice!**



## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

# Wrapping up

Ariel Rokem

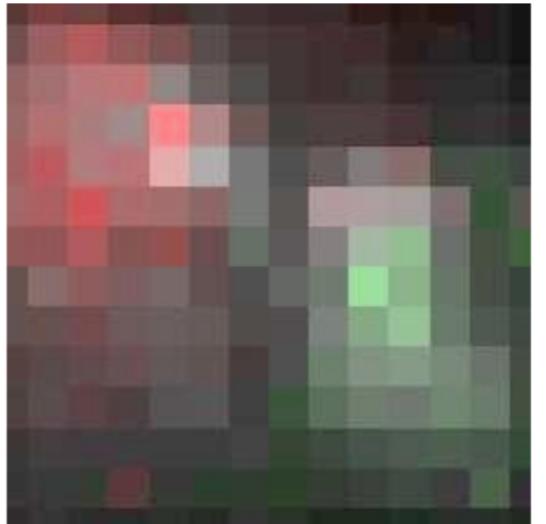
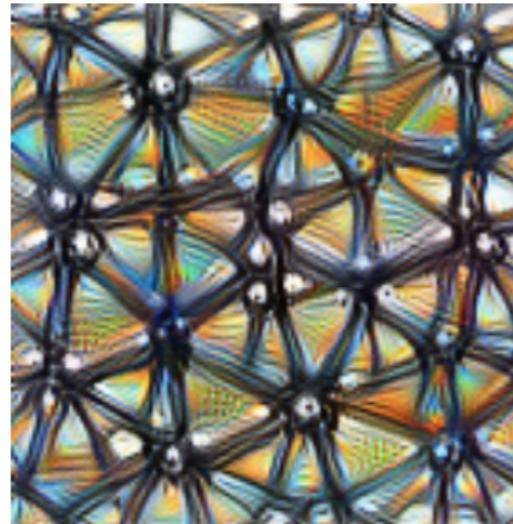
Senior Data Scientist, University of Washington

# What did we learn?

- Image classification
- Convolutions
- Reducing the number of parameters
  - Tweaking your convolutions
  - Adding pooling layers
- Improving your network
  - Regularization
- Understanding your network
  - Monitoring learning
  - Interpreting the parameters

# Model interpretation

<https://distill.pub/2017/feature-visualization/>



**Feature visualization** answers questions about what a network—or parts of a network—are looking for by generating examples.

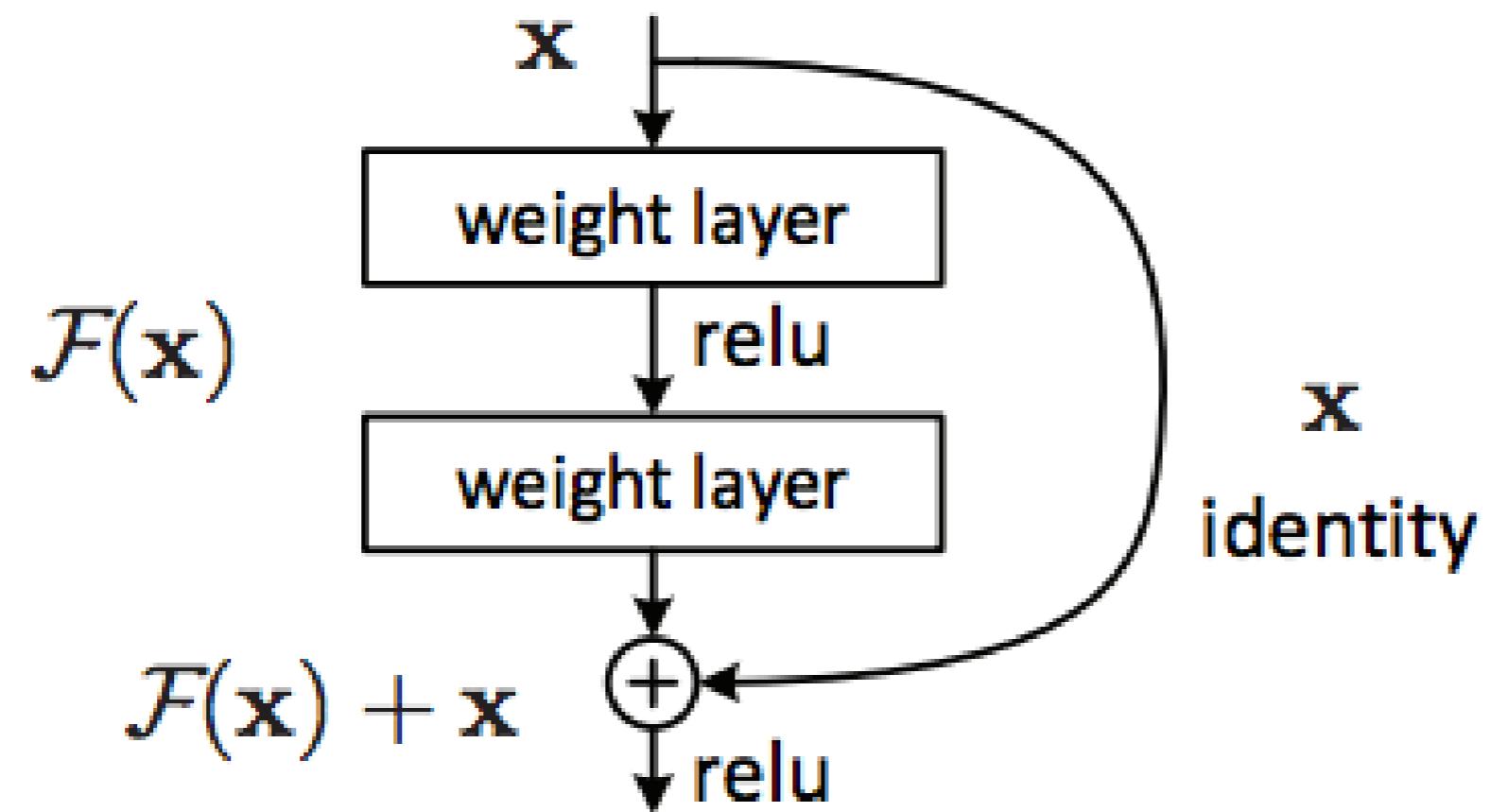
**Attribution** <sup>1</sup> studies what part of an example is responsible for the network activating a particular way.



# What next?

- Even deeper networks
- Residual networks

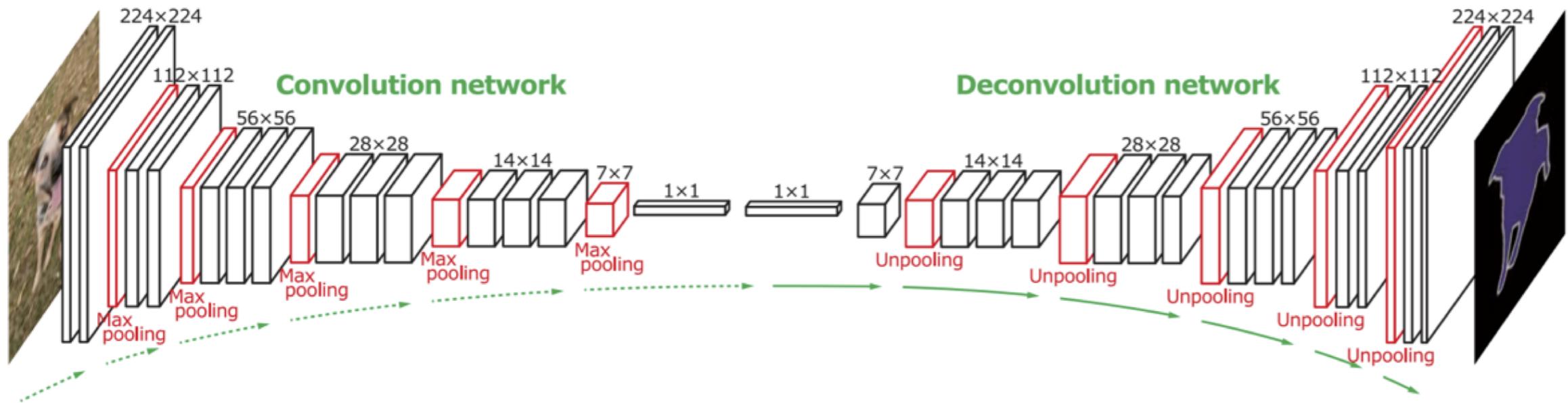
# Residual networks



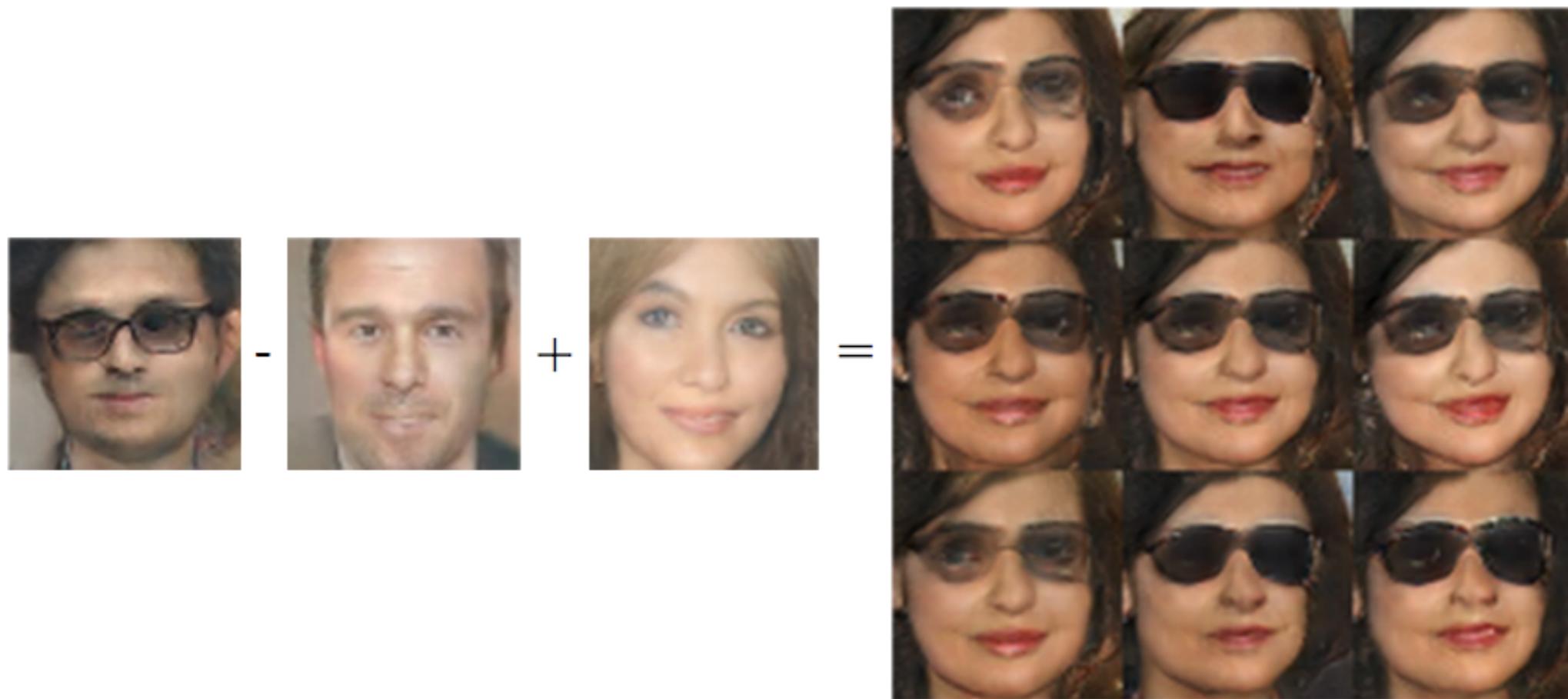
# What next?

- Even deeper networks
- Residual networks
- Transfer learning
- Fully convolutional networks

# Fully convolutional networks



# Generative adversarial networks



# What next?

- Even deeper networks
- Residual networks
- Transfer learning
- Fully convolutional networks
- Generative adversarial networks
- ...



## CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE PROCESSING

**Good luck!**