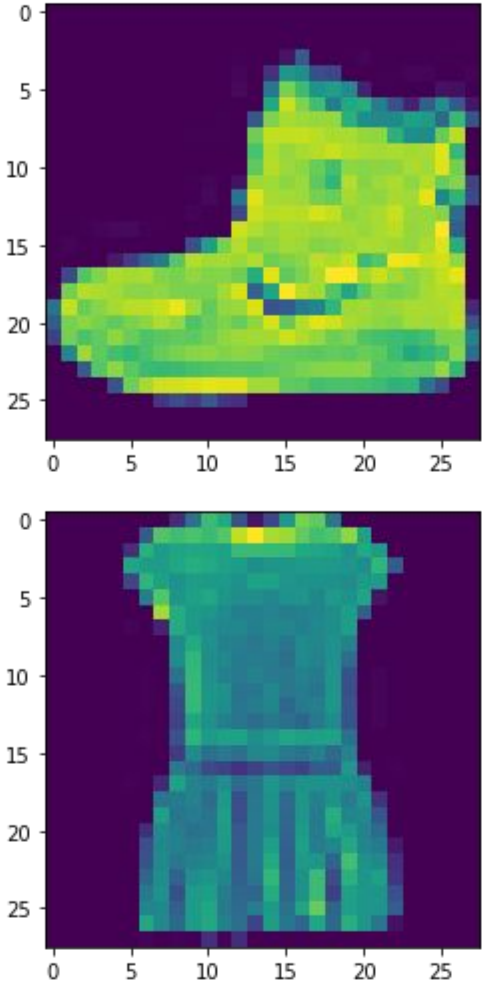
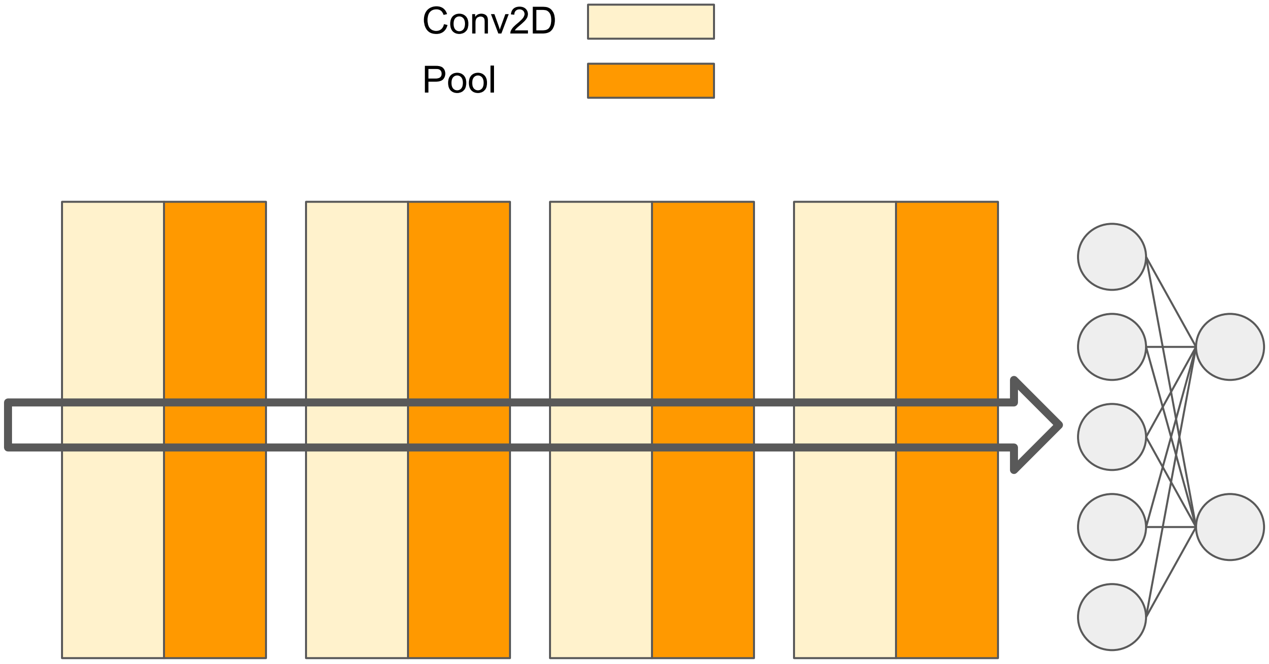
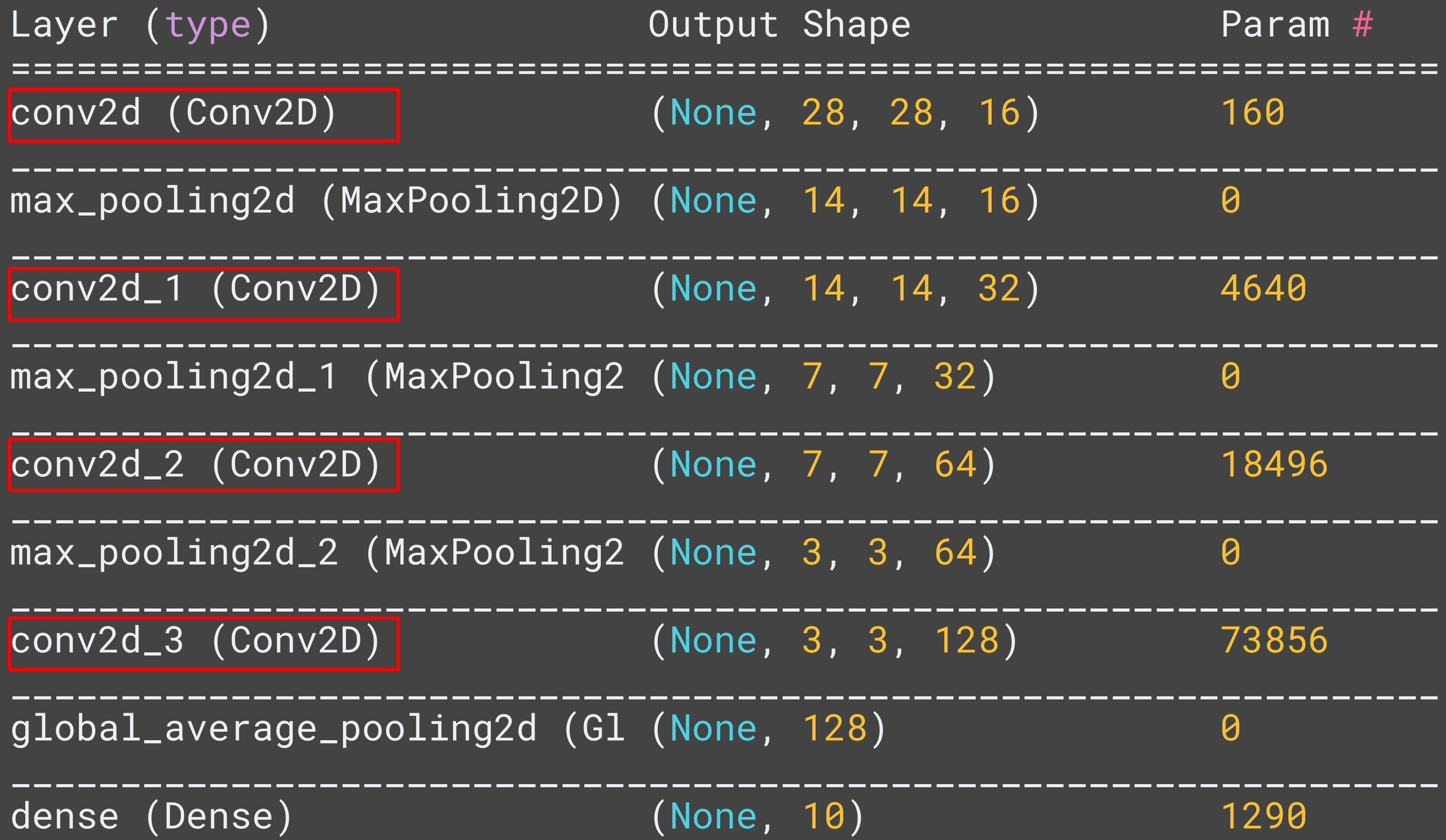
Copyright Notice

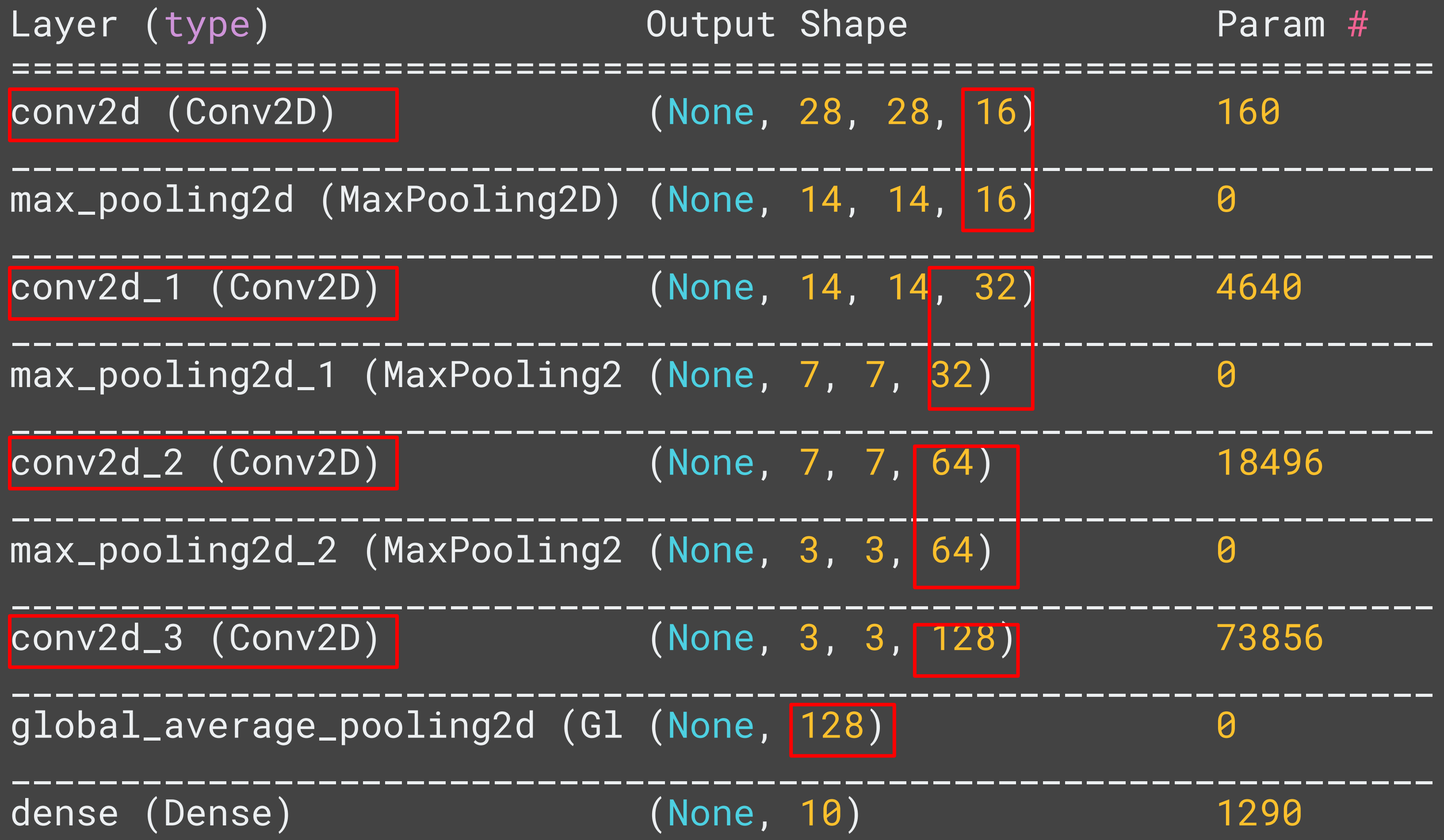

0
5
10
15
20
25
0
5
10
15
20
25


Conv2D
Pool

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160

-----------------------------------------------------------------
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0

-----------------------------------------------------------------
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640

-----------------------------------------------------------------
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0

-----------------------------------------------------------------
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496

-----------------------------------------------------------------
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0

-----------------------------------------------------------------
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856

-----------------------------------------------------------------
global_average_pooling2d (Gl (None, 128)               0

-----------------------------------------------------------------
dense (Dense)                (None, 10)                1290
```



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

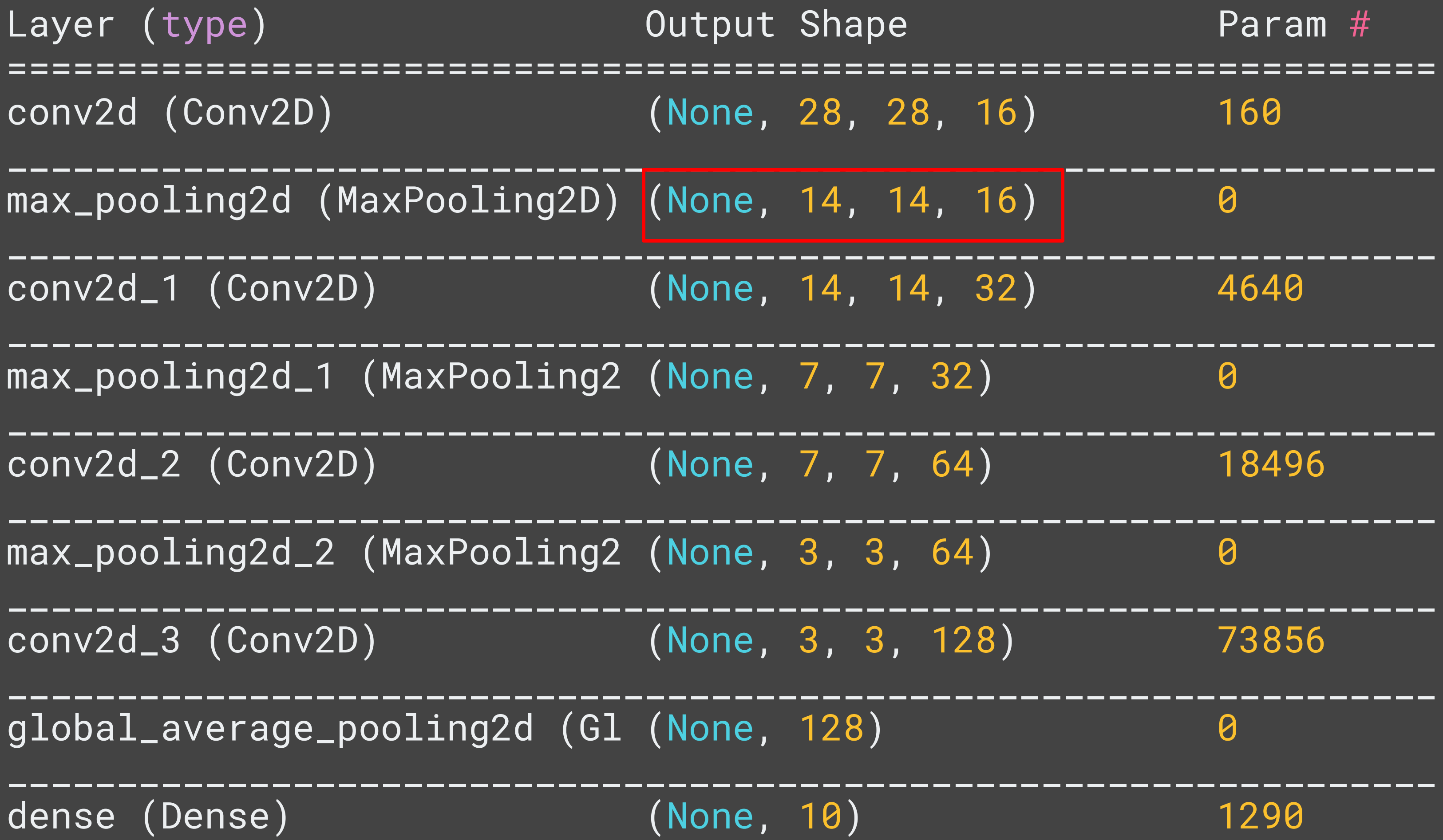


```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

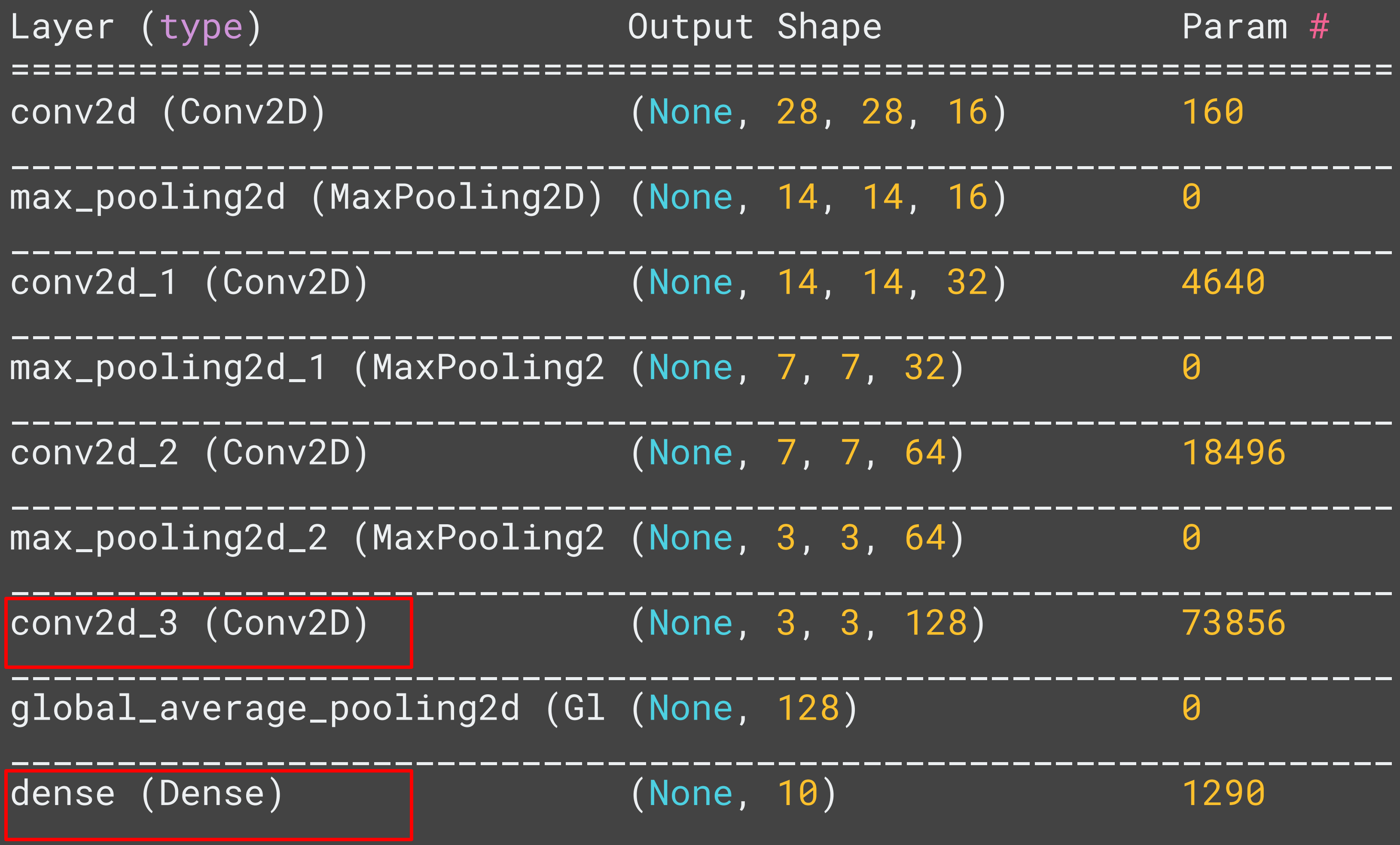


```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
-----------------------------------------------------------------
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
-----------------------------------------------------------------
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
-----------------------------------------------------------------
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
-----------------------------------------------------------------
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
-----------------------------------------------------------------
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
-----------------------------------------------------------------
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
-----------------------------------------------------------------
global_average_pooling2d (Gl (None, 128)               0
-----------------------------------------------------------------
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 10)                1290
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 128)         73856       -3
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0           -2
_________________________________________________________________
dense (Dense)                (None, 10)                1290
                                                                   -1
```

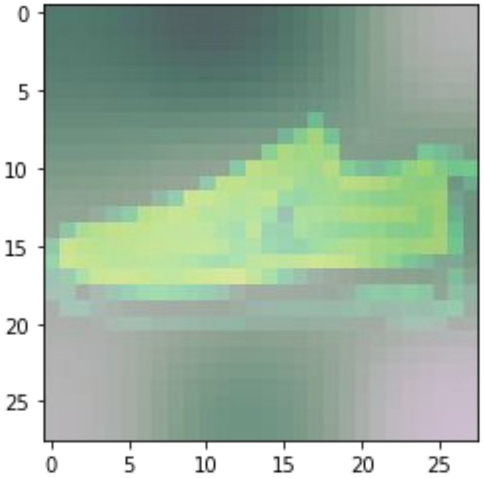
```
cam_model = Model(inputs=model.input,
                  outputs=(model.layers[-3].output,
                           model.layers[-1].output))
```

```
cam_model = Model(inputs=model.input,
                  outputs=(model.layers[-3].output,
                           model.layers[-1].output))
```

```
features,results = cam_model.predict(X_test)
```

```
def show_cam(image_index):
  features_for_img = features[image_index,:,:,:]
  prediction = np.argmax(results[image_index])
  class_activation_weights = gap_weights[:,prediction]
  class_activation_features =
        sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  cam_output  = np.dot(class_activation_features,class_activation_weights)
```

Class Activation Map



```
def show_cam(image_index):
  features_for_img = features[image_index,:,:,:]
  prediction = np.argmax(results[image_index])
  class_activation_weights = gap_weights[:,prediction]
  class_activation_features =
        sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  cam_output  = np.dot(class_activation_features,class_activation_weights)
```

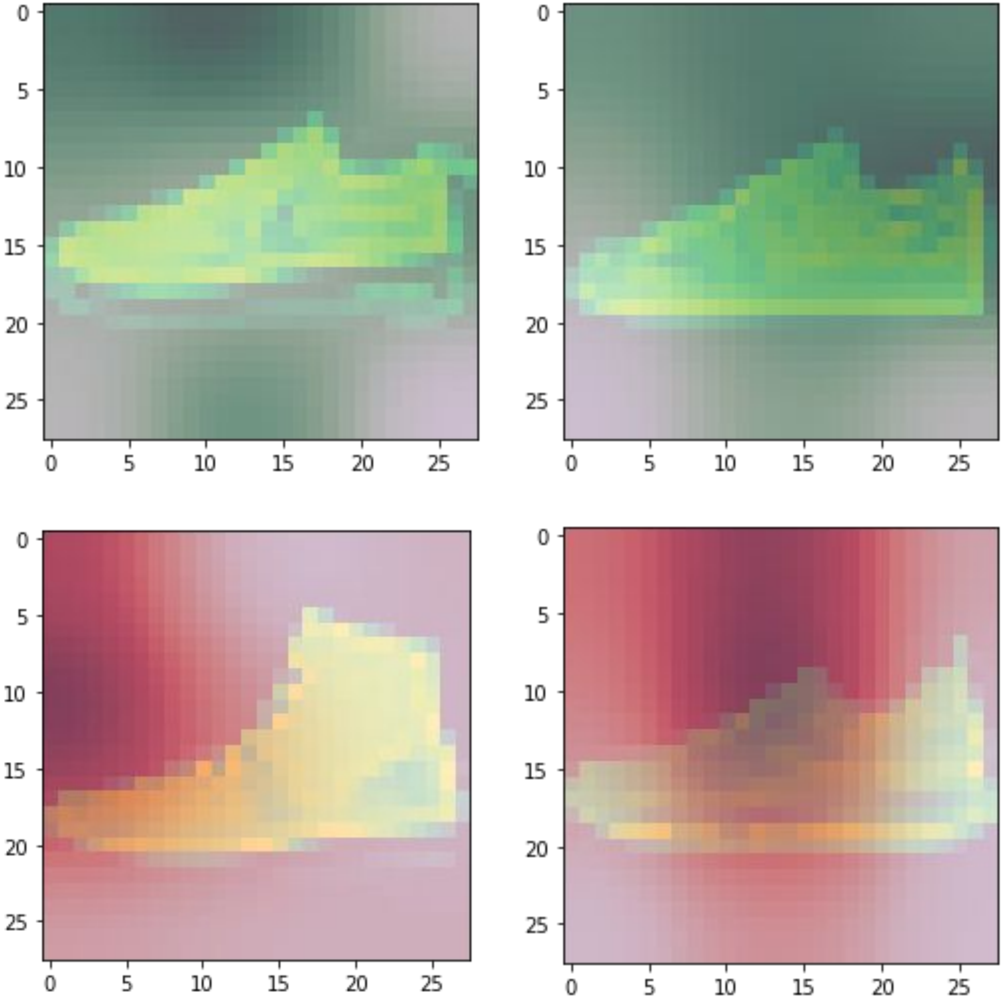
```
def show_cam(image_index):
  features_for_img = features[image_index,:,:,:]
  prediction = np.argmax(results[image_index])
  class_activation_weights = gap_weights[:,prediction]
  class_activation_features =
        sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  cam_output  = np.dot(class_activation_features,class_activation_weights)
```

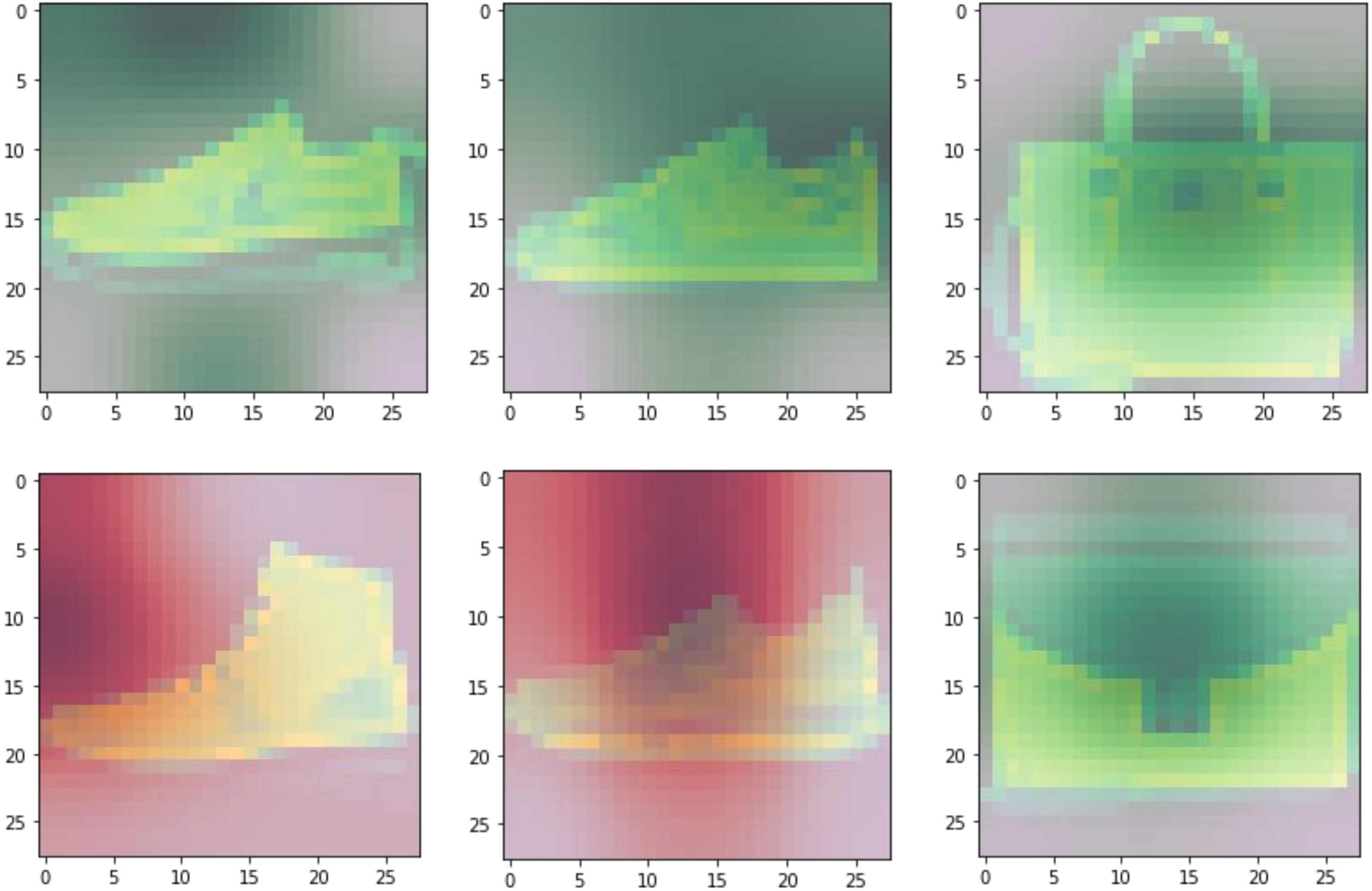
```python
def show_cam(image_index):
  features_for_img = features[image_index,:,:,:]
  prediction = np.argmax(results[image_index])
  class_activation_weights = gap_weights[:,prediction]
  class_activation_features =
        sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  cam_output  = np.dot(class_activation_features,class_activation_weights)
```

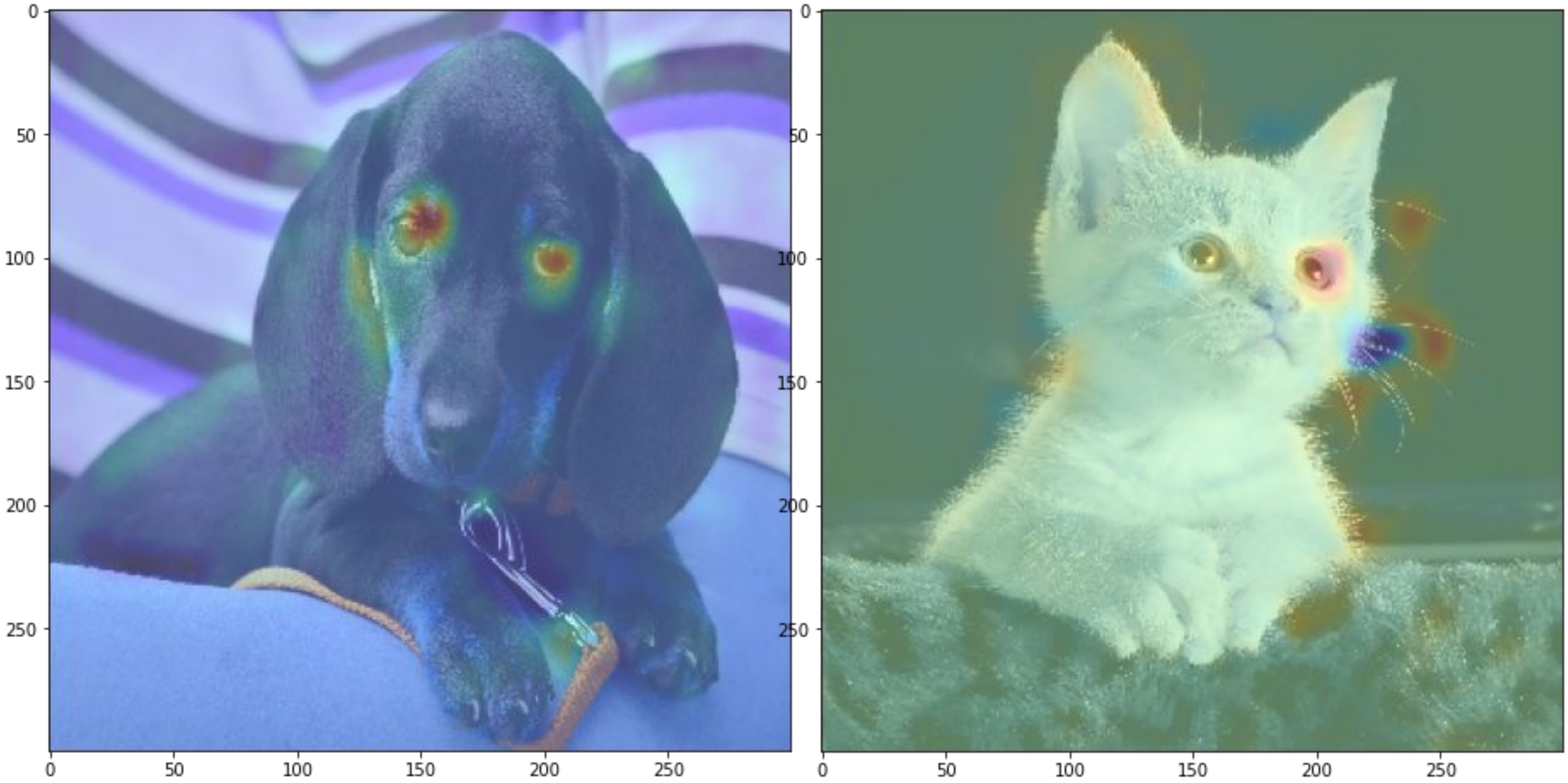
```
def show_cam(image_index):
  features_for_img = features[image_index,:,:,:]
  prediction = np.argmax(results[image_index])
  class_activation_weights = gap_weights[:,prediction]
  class_activation_features =
        sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  cam_output  = np.dot(class_activation_features,class_activation_weights)
```

```python
def show_cam(image_index):
  features_for_img = features[image_index,:,:,:]
  prediction = np.argmax(results[image_index])
  class_activation_weights = gap_weights[:,prediction]
  class_activation_features =
        sp.ndimage.zoom(features_for_img, (28/3, 28/3, 1), order=2)
  cam_output  = np.dot(class_activation_features,class_activation_weights)
```

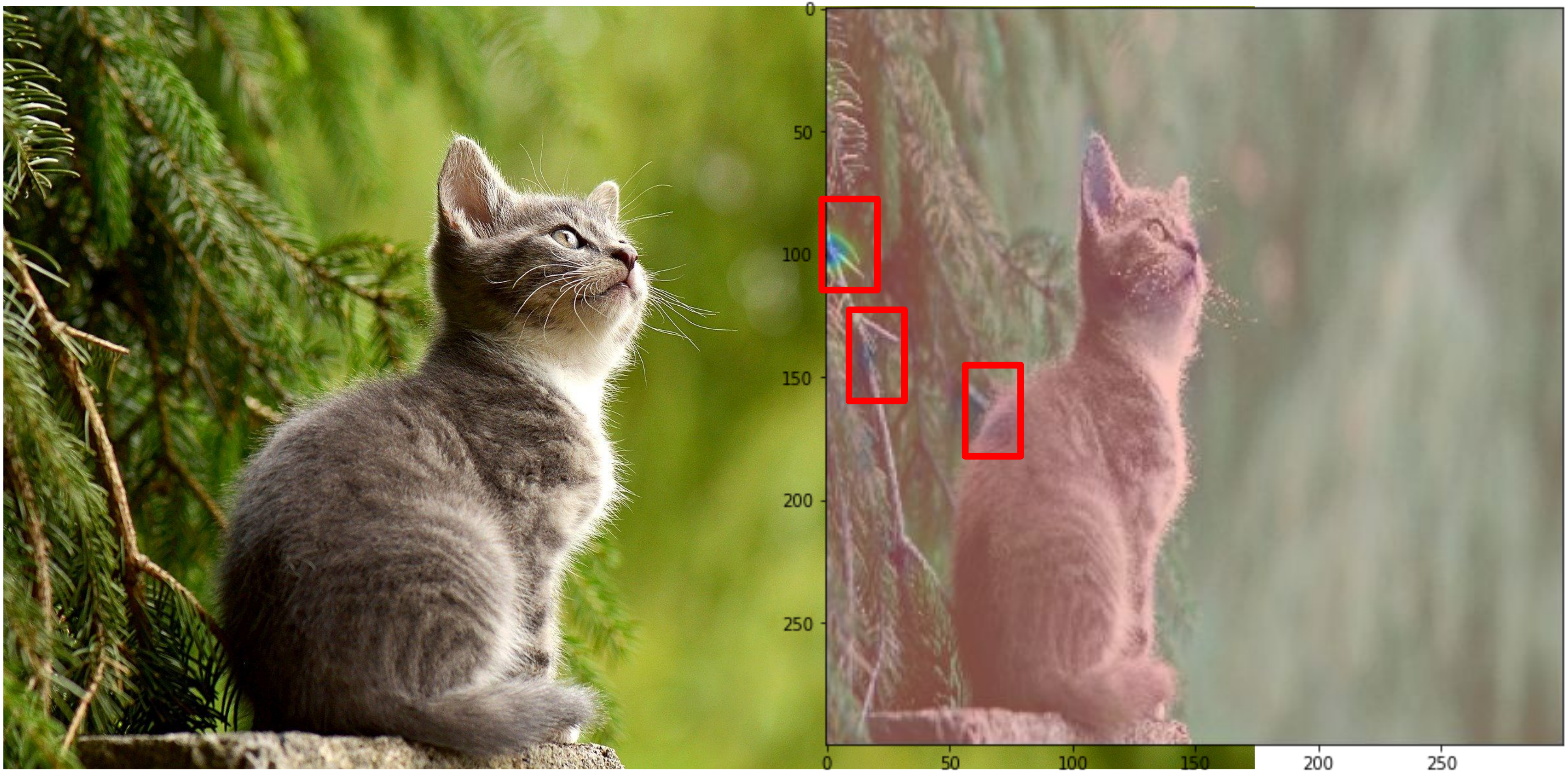
Class Activation Map




0
5
10
15
20
25


0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

```
class_index = 251   # Siberian Husky's class ID in ImageNet
num_classes = 1001
expected_output = tf.one_hot([class_index] * images.shape[0], num_classes)

with tf.GradientTape() as tape:
    inputs = tf.cast(images, tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)
```

```
class_index = 251   # Siberian Husky's class ID in ImageNet
num_classes = 1001
expected_output = tf.one_hot([class_index] * images.shape[0], num_classes)

with tf.GradientTape() as tape:
    inputs = tf.cast(images, tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)
```

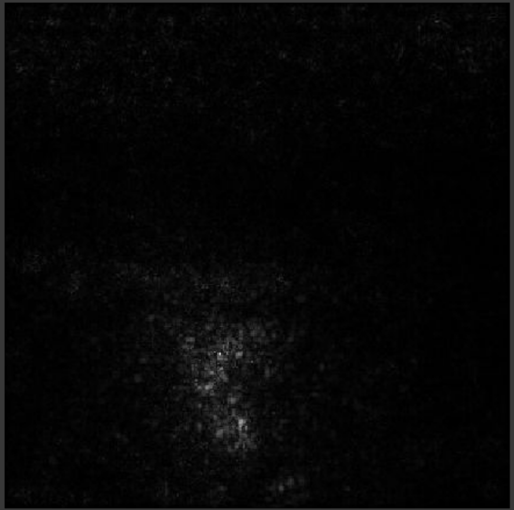
```
class_index = 251   # Siberian Husky's class ID in ImageNet
num_classes = 1001
expected_output = tf.one_hot([class_index] * images.shape[0], num_classes)

with tf.GradientTape() as tape:
    inputs = tf.cast(images, tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)
```

```
class_index = 251   # Siberian Husky's class ID in ImageNet
num_classes = 1001
expected_output = tf.one_hot([class_index] * images.shape[0], num_classes)

with tf.GradientTape() as tape:
    inputs = tf.cast(images, tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)
```

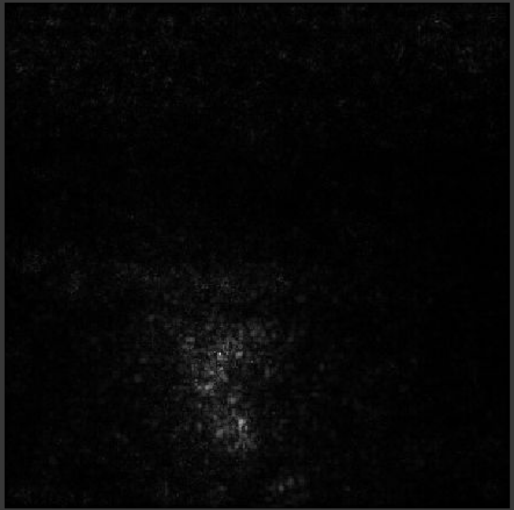
```
class_index = 251   # Siberian Husky's class ID in ImageNet
num_classes = 1001
expected_output = tf.one_hot([class_index] * images.shape[0], num_classes)

with tf.GradientTape() as tape:
    inputs = tf.cast(images, tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)
```

```
class_index = 251   # Siberian Husky's class ID in ImageNet
num_classes = 1001
expected_output = tf.one_hot([class_index] * images.shape[0], num_classes)

with tf.GradientTape() as tape:
    inputs = tf.cast(images, tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)
```

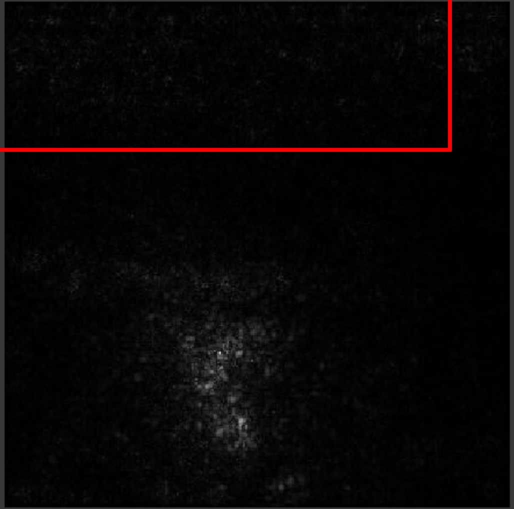
```
grayscale_tensor = tf.reduce_sum(tf.abs(gradients), axis=-1)
normalized_tensor = tf.cast(
    255 * (grayscale_tensor - tf.reduce_min(grayscale_tensor))
    / (tf.reduce_max(grayscale_tensor) - tf.reduce_min(grayscale_tensor)),
    tf.uint8,
)
normalized_tensor = tf.squeeze(normalized_tensor)

plt.figure(figsize=(8, 8))
plt.axis('off')
plt.imshow(normalized_tensor, cmap='gray')
plt.show()
```



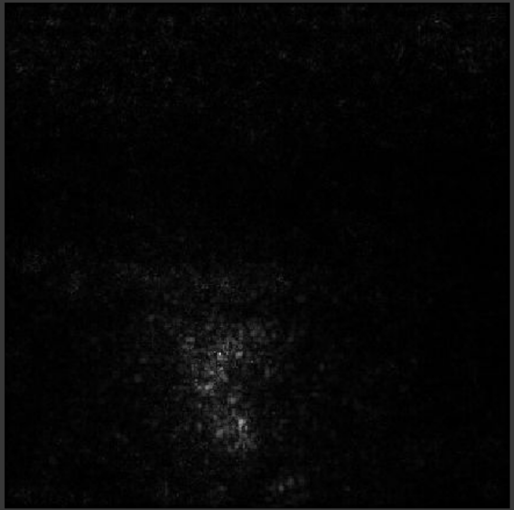
```
grayscale_tensor = tf.reduce_sum(tf.abs(gradients), axis=-1)
normalized_tensor = tf.cast(
    255 * (grayscale_tensor - tf.reduce_min(grayscale_tensor))
    / (tf.reduce_max(grayscale_tensor) - tf.reduce_min(grayscale_tensor)),
    tf.uint8,
)
normalized_tensor = tf.squeeze(normalized_tensor)

plt.figure(figsize=(8, 8))
plt.axis('off')
plt.imshow(normalized_tensor, cmap='gray')
plt.show()
```



```
grayscale_tensor = tf.reduce_sum(tf.abs(gradients), axis=-1)
normalized_tensor = tf.cast(
    255 * (grayscale_tensor - tf.reduce_min(grayscale_tensor))
    / (tf.reduce_max(grayscale_tensor) - tf.reduce_min(grayscale_tensor)),
    tf.uint8,
)
normalized_tensor = tf.squeeze(normalized_tensor)

plt.figure(figsize=(8, 8))
plt.axis('off')
plt.imshow(normalized_tensor, cmap='gray')
plt.show()
```

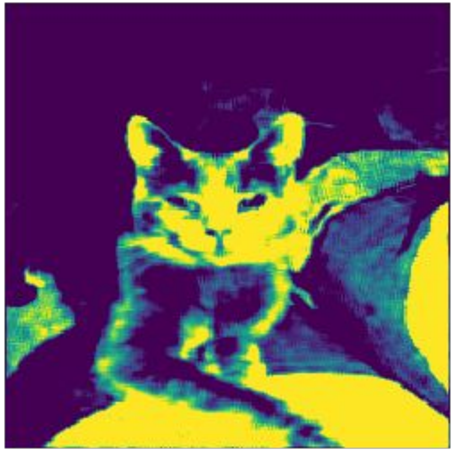


```
grayscale_tensor = tf.reduce_sum(tf.abs(gradients), axis=-1)
normalized_tensor = tf.cast(
    255 * (grayscale_tensor - tf.reduce_min(grayscale_tensor))
    / (tf.reduce_max(grayscale_tensor) - tf.reduce_min(grayscale_tensor)),
    tf.uint8,
)
normalized_tensor = tf.squeeze(normalized_tensor)

plt.figure(figsize=(8, 8))
plt.axis('off')
plt.imshow(normalized_tensor, cmap='gray')
plt.show()
```

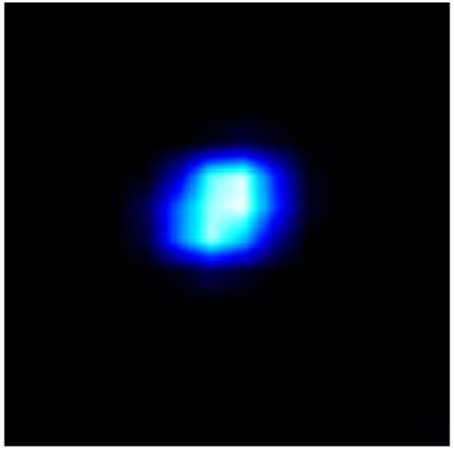


True label: 0
Predicted label: 0

Random feature map



Class Activation Map



Activation map superimposed

```
def get_CAM(processed_image, predicted_label, layer_name='block5_conv3'):
    model_grad = Model([model.inputs],
                       [model.get_layer(layer_name).output,
                        model.output])


    # Gradient tape
    with tf.GradientTape() as tape:
        conv_output_values, predictions = model_grad(processed_image)
        loss = predictions[:, predicted_label]
```

```
def get_CAM(processed_image, predicted_label, layer_name='block5_conv3'):
    model_grad = Model([model.inputs],
                       [model.get_layer(layer_name).output,
                       model.output])

    # Gradient tape
    with tf.GradientTape() as tape:
        conv_output_values, predictions = model_grad(processed_image)
        loss = predictions[:, predicted_label]
```

```python
def get_CAM(processed_image, predicted_label, layer_name='block5_conv3'):
    model_grad = Model([model.inputs],
                       [model.get_layer(layer_name).output,
                        model.output])


    # Gradient tape
    with tf.GradientTape() as tape:
        conv_output_values, predictions = model_grad(processed_image)
        loss = predictions[:, predicted_label]
```

```
def get_CAM(processed_image, predicted_label, layer_name='block5_conv3'):
    model_grad = Model([model.inputs],
                       [model.get_layer(layer_name).output,
                       model.output])

    # Gradient tape
    with tf.GradientTape() as tape:
        conv_output_values, predictions = model_grad(processed_image)
        loss = predictions[:, predicted_label]
```

```python
def get_CAM(processed_image, predicted_label, layer_name='block5_conv3'):
    model_grad = Model([model.inputs],
                       [model.get_layer(layer_name).output,
                       model.output])

    # Gradient tape
    with tf.GradientTape() as tape:
        conv_output_values, predictions = model_grad(processed_image)
        loss = predictions[:, predicted_label]
```

```
def get_CAM(processed_image, predicted_label, layer_name='block5_conv3'):
    model_grad = Model([model.inputs],
                       [model.get_layer(layer_name).output,
                        model.output])

    # Gradient tape
    with tf.GradientTape() as tape:
        conv_output_values, predictions = model_grad(processed_image)
        loss = predictions[:, predicted_label]
```

```
# Compute the gradients
grads_values = tape.gradient(loss, conv_output_values)

# Mean of gradients per feature map
grads_values = K.mean(grads_values, axis=(0,1,2))

conv_output_values = np.squeeze(conv_output_values.numpy())
grads_values = grads_values.numpy()
...
```

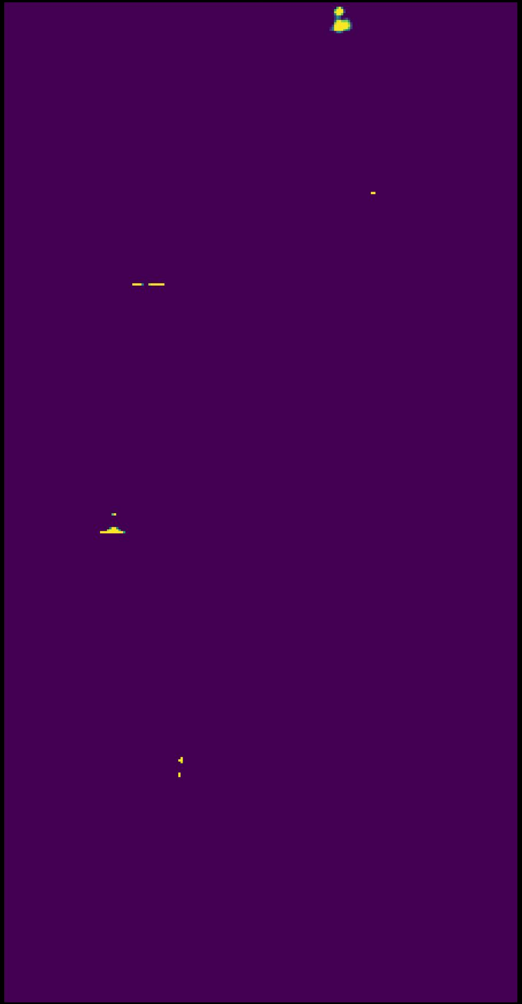
```
# Compute the gradients
grads_values = tape.gradient(loss, conv_output_values)

# Mean of gradients per feature map
grads_values = K.mean(grads_values, axis=(0,1,2))

conv_output_values = np.squeeze(conv_output_values.numpy())
grads_values = grads_values.numpy()
...
```

```
# Compute the gradients
grads_values = tape.gradient(loss, conv_output_values)

# Mean of gradients per feature map
grads_values = K.mean(grads_values, axis=(0,1,2))

conv_output_values = np.squeeze(conv_output_values.numpy())
grads_values = grads_values.numpy()
...
```

```
# Compute the gradients
grads_values = tape.gradient(loss, conv_output_values)

# Mean of gradients per feature map
grads_values = K.mean(grads_values, axis=(0,1,2))

conv_output_values = np.squeeze(conv_output_values.numpy())
grads_values = grads_values.numpy()
...
```

```python
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]


# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss


return heatmap
```

```python
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]

# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss

return heatmap
```



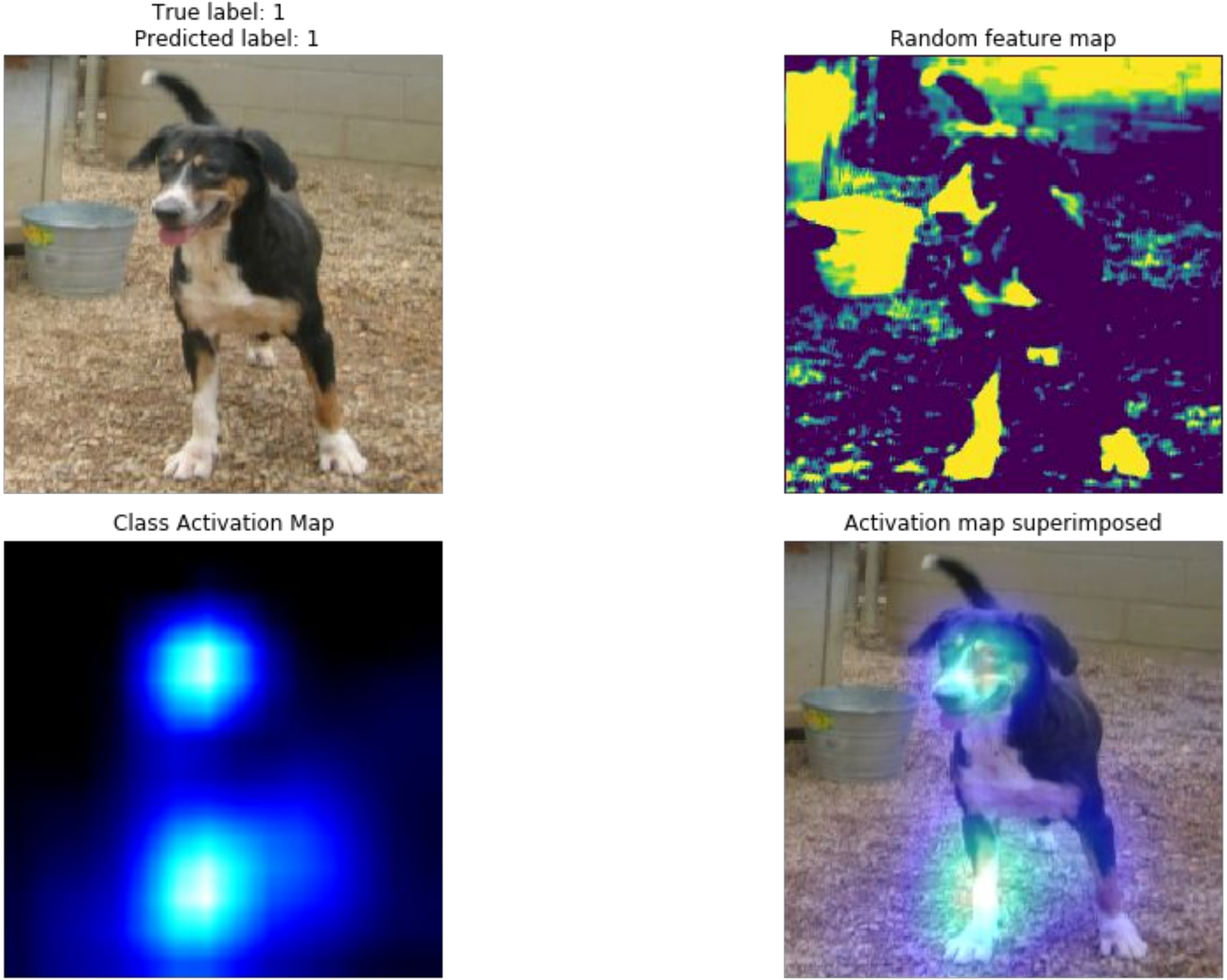
```
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]

# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss

return heatmap
```

```
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]

# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss

return heatmap
```

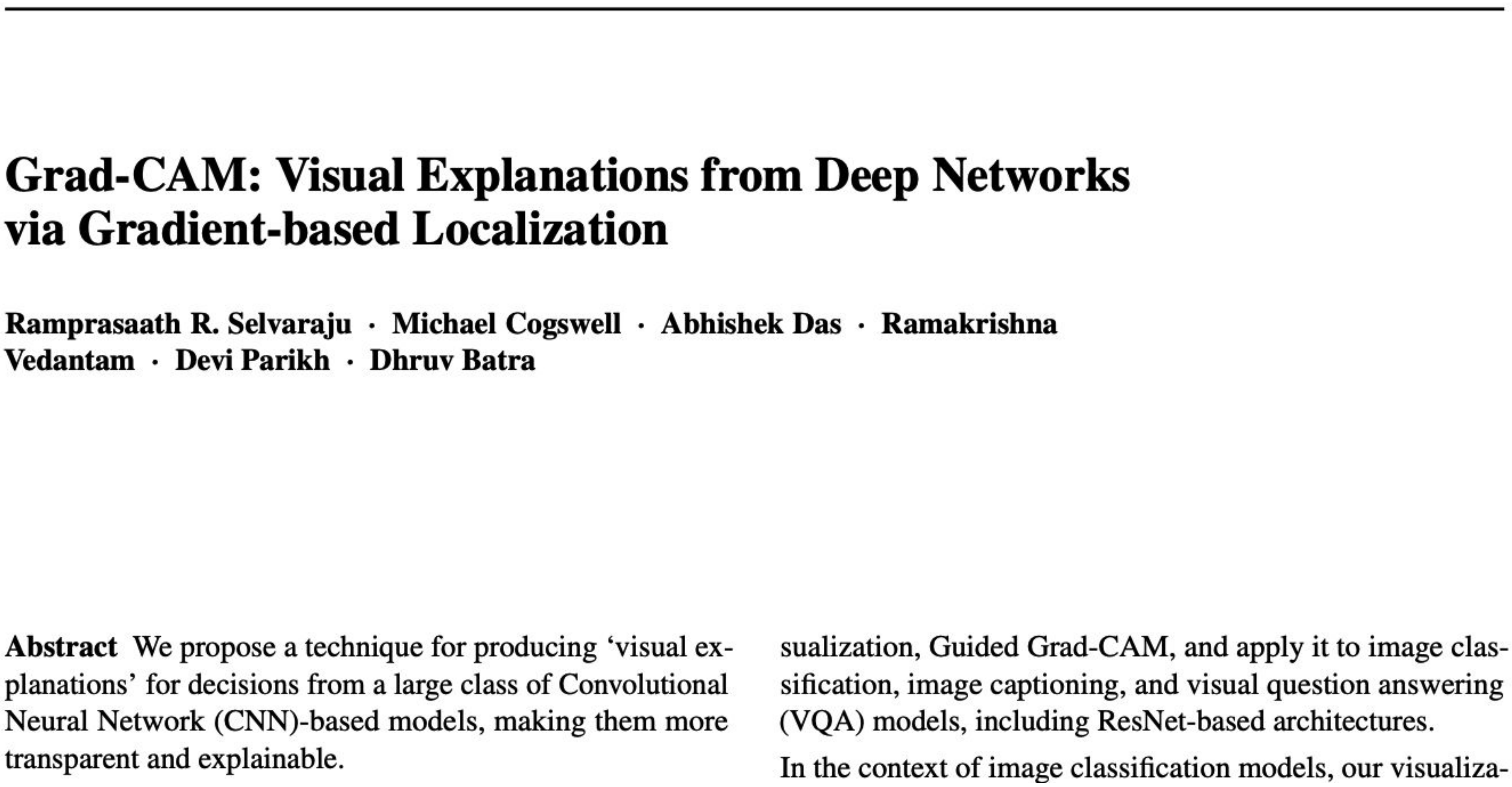
```python
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]


# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss


return heatmap
```

```
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]


# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss


return heatmap
```

```python
for i in range(512): # we have 512 features in our last conv layer
    conv_output_values[:,:,i] *= grads_values[i]

# Heatmap
heatmap = np.mean(conv_output_values, axis=-1)
# Remove negative values
heatmap = np.maximum(heatmap, 0)
# Normalize
heatmap /= heatmap.max()
del model_grad, conv_output_values, grads_values, loss

return heatmap
```


True label: 1
Predicted label: 1
Random feature map
Class Activation Map
Activation map superimposed

https://arxiv.org/pdf/1610.02391.pdf



Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Ramprasaath R. Selvaraju · Michael Cogswell · Abhishek Das · Ramakrishna Vedantam · Devi Parikh · Dhruv Batra

Abstract We propose a technique for producing 'visual explanations' for decisions from a large class of Convolutional Neural Network (CNN)-based models, making them more transparent and explainable.

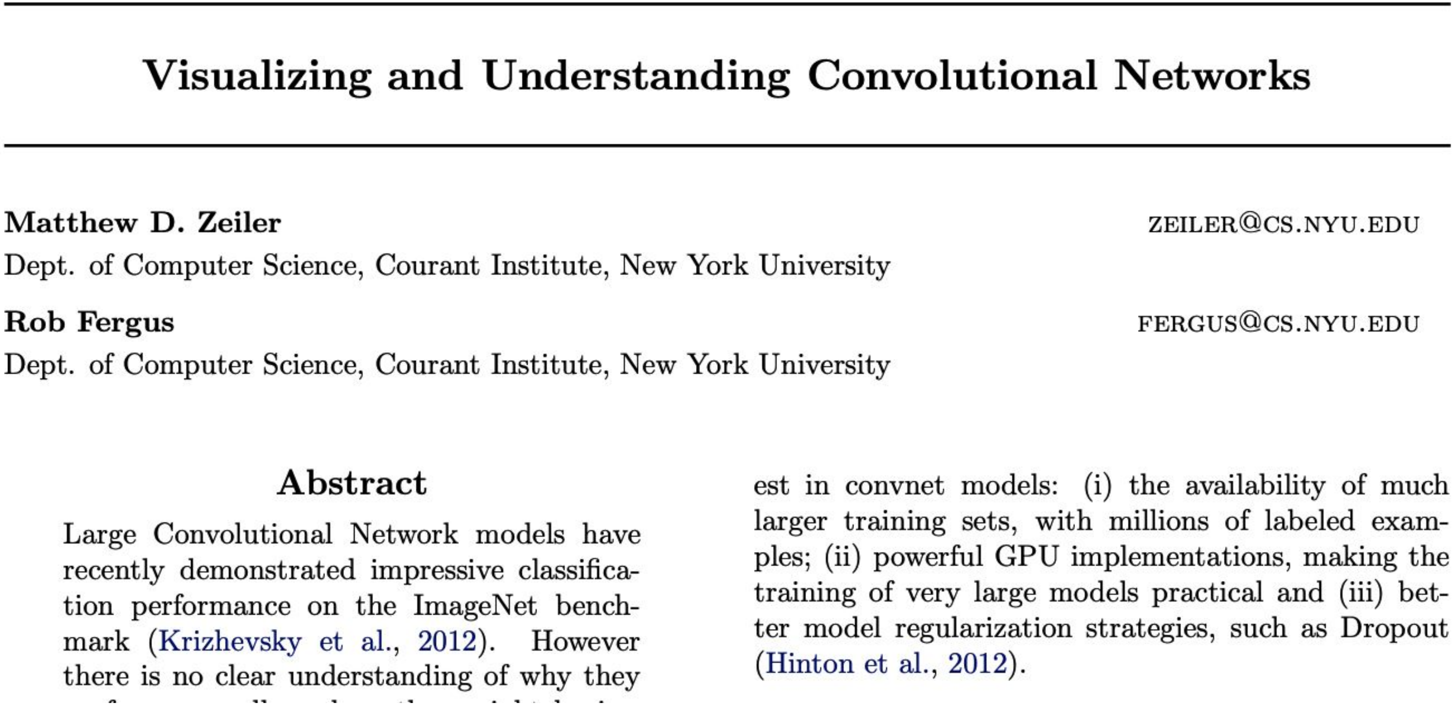
sualization, Guided Grad-CAM, and apply it to image classification, image captioning, and visual question answering (VQA) models, including ResNet-based architectures.

In the context of image classification models, our visualiza-

ZFNet

- Winner of ILSVLC 2013
- Fine tuned the AlexNet architecture
- Employed a Deconvolutional technique
- Provides visualization of activations for each layer

https://arxiv.org/pdf/1311.2901v3.pdf



Visualizing and Understanding Convolutional Networks

Matthew D. Zeiler ZEILER@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

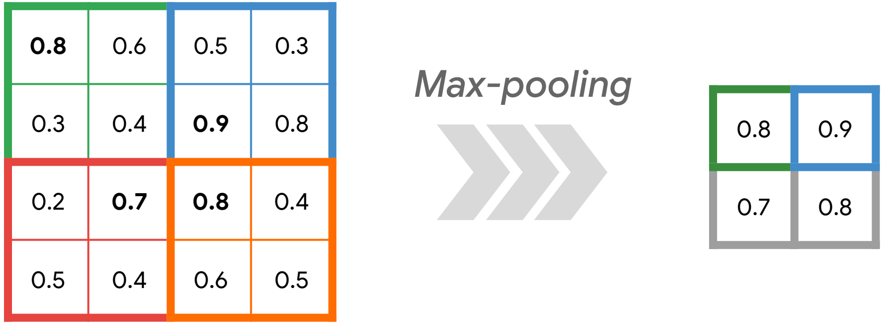
Rob Fergus FERGUS@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

Abstract

Large Convolutional Network models have recently demonstrated impressive classification performance on the ImageNet benchmark (Krizhevsky et al., 2012). However there is no clear understanding of why they

est in convnet models: (i) the availability of much larger training sets, with millions of labeled examples; (ii) powerful GPU implementations, making the training of very large models practical and (iii) better model regularization strategies, such as Dropout (Hinton et al., 2012).

Unpooling

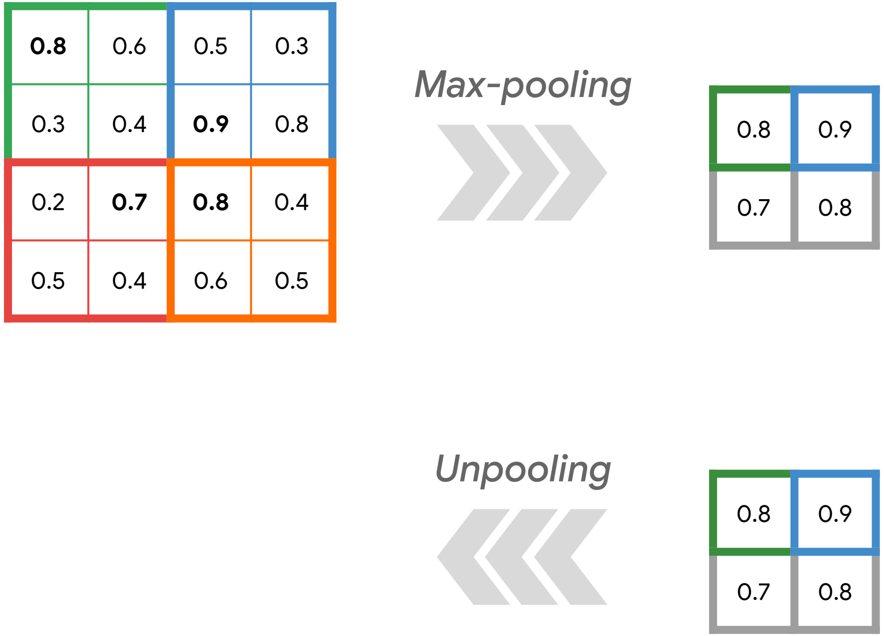


0.8	0.6	0.5	0.3
0.3	0.4	**0.9**	0.8
0.2	**0.7**	**0.8**	0.4
0.5	0.4	0.6	0.5

Max-pooling

0.8	0.9
0.7	0.8

Unpooling



0.8	0.6	0.5	0.3
0.3	0.4	**0.9**	0.8
0.2	**0.7**	**0.8**	0.4
0.5	0.4	0.6	0.5

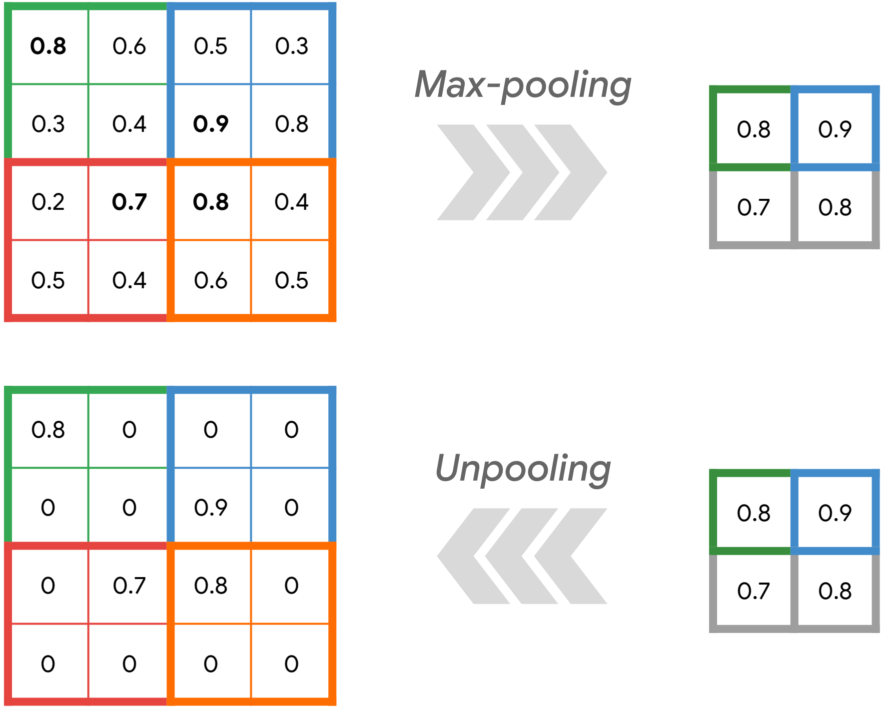
Max-pooling

0.8	0.9
0.7	0.8

Unpooling

0.8	0.9
0.7	0.8

Unpooling



0.8	0.6	0.5	0.3
0.3	0.4	**0.9**	0.8
0.2	**0.7**	**0.8**	0.4
0.5	0.4	0.6	0.5

Max-pooling

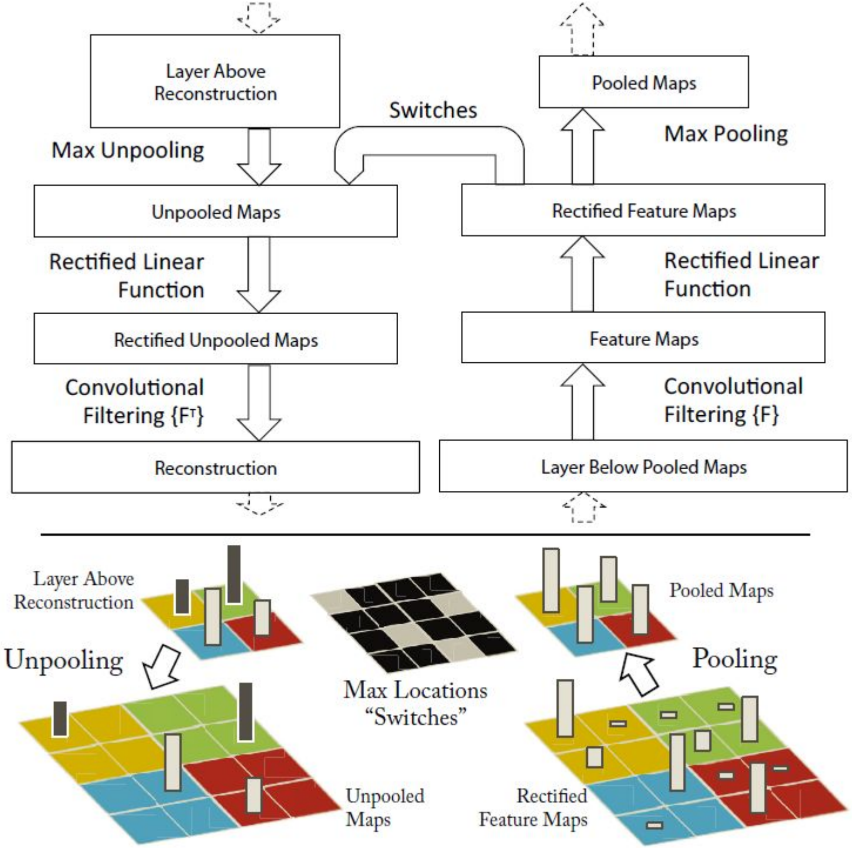
0.8	0.9
0.7	0.8

0.8	0	0	0
0	0	0.9	0
0	0.7	0.8	0
0	0	0	0

Unpooling

0.8	0.9
0.7	0.8

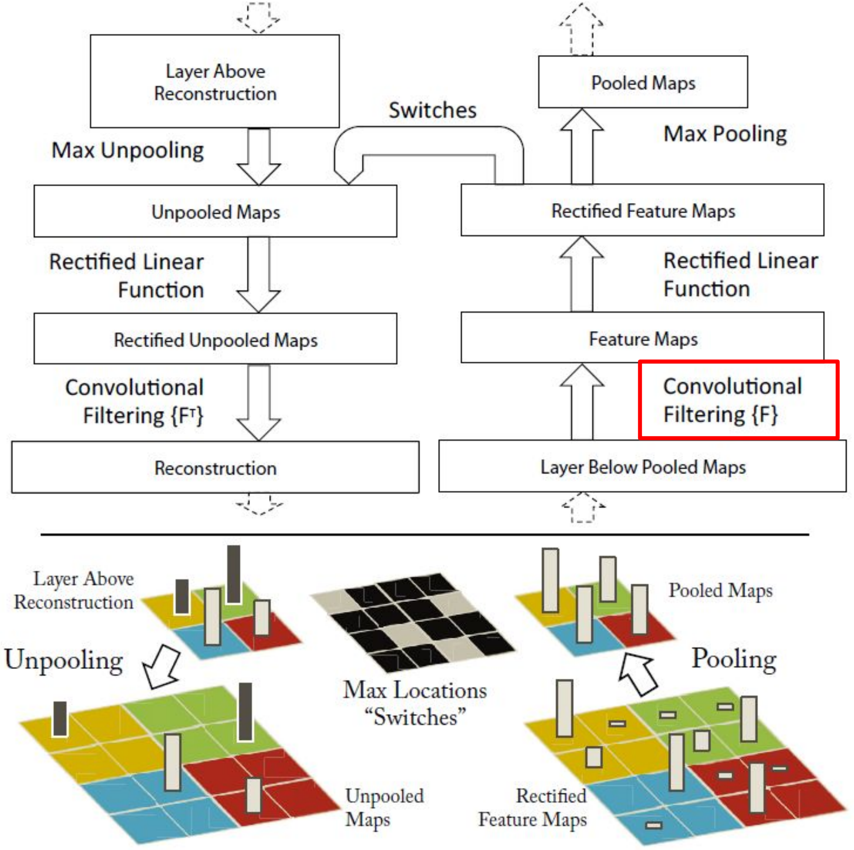
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

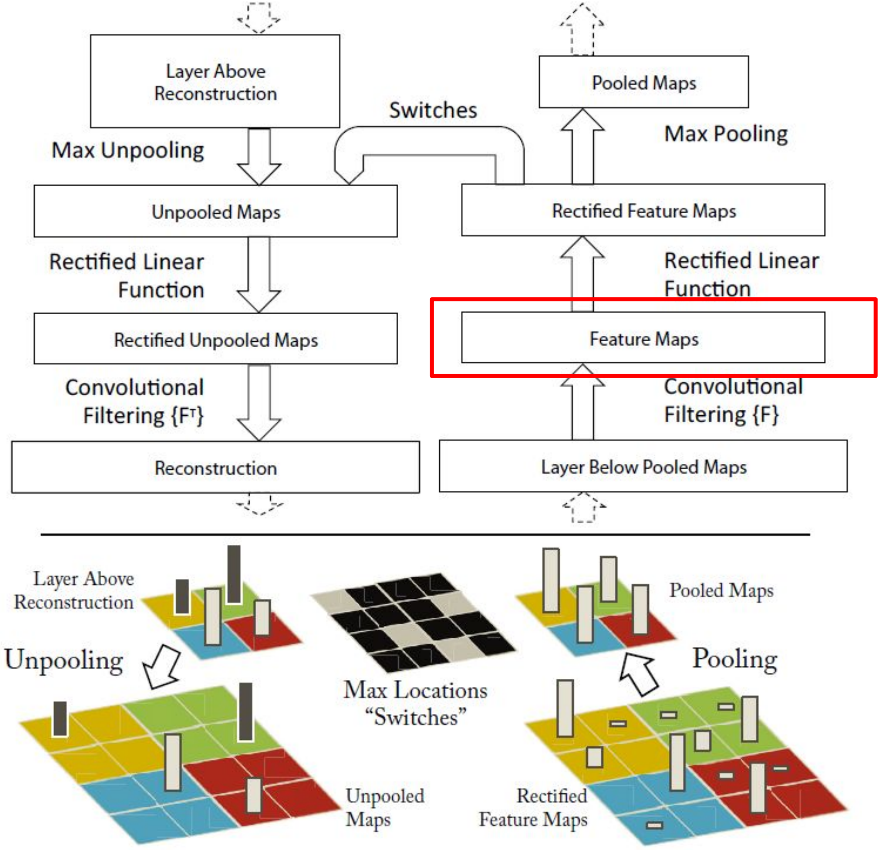
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

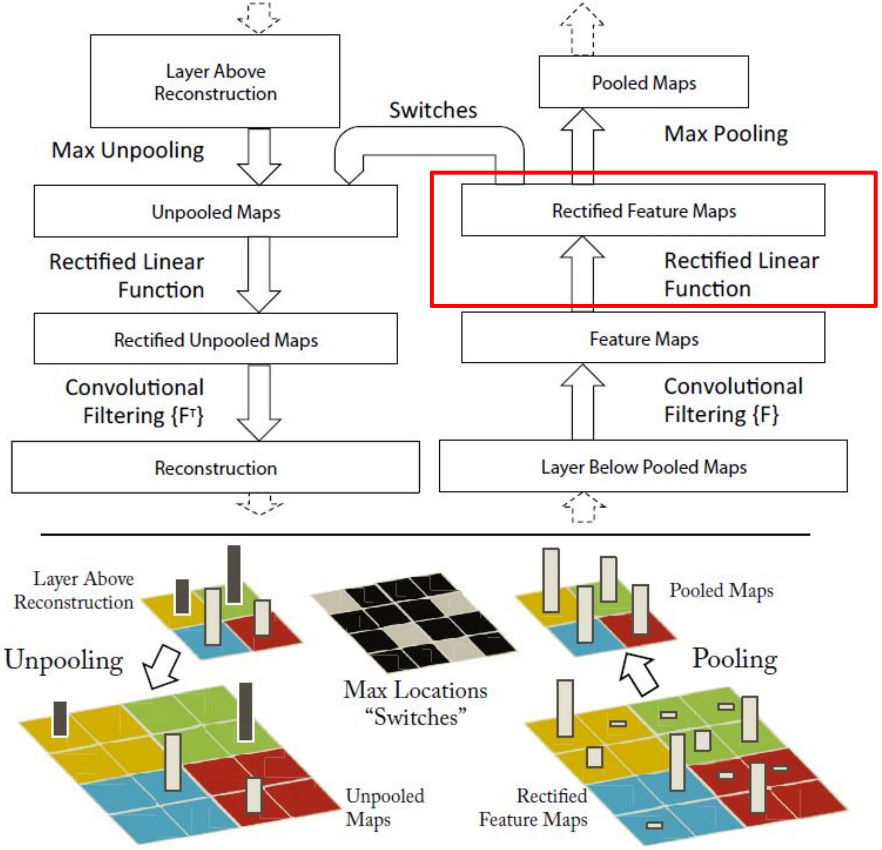
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

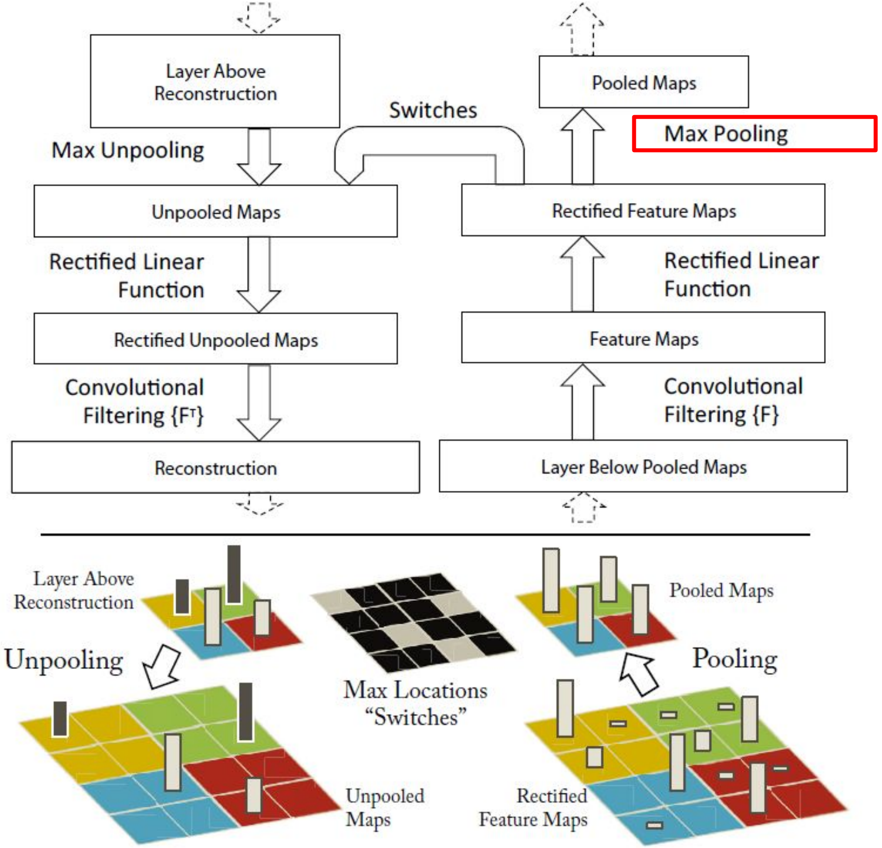
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

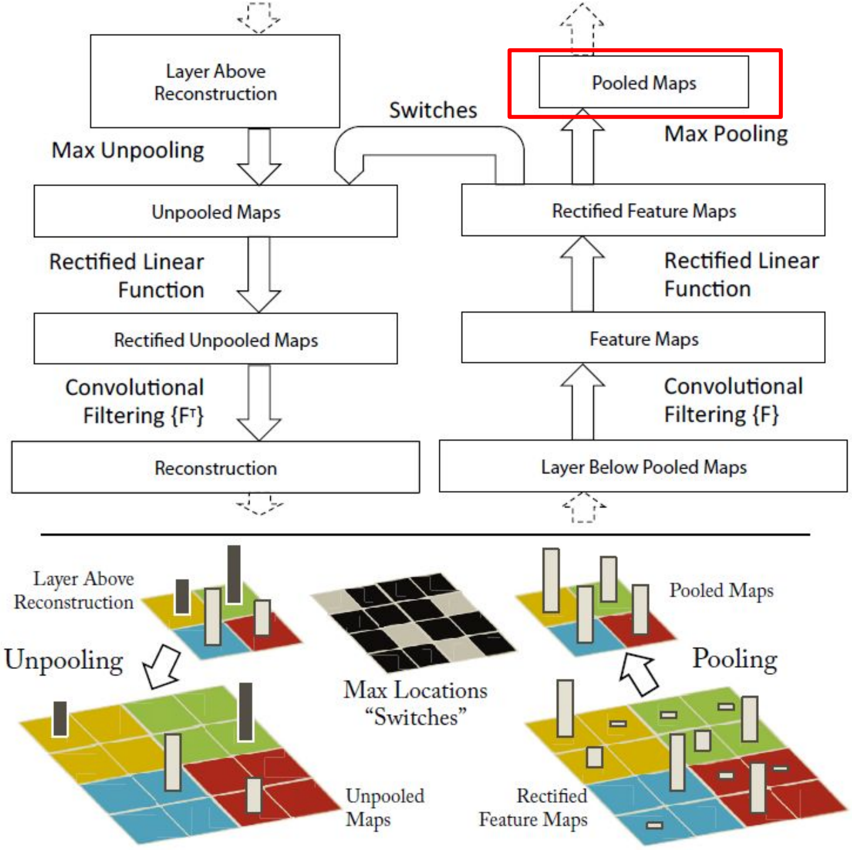
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

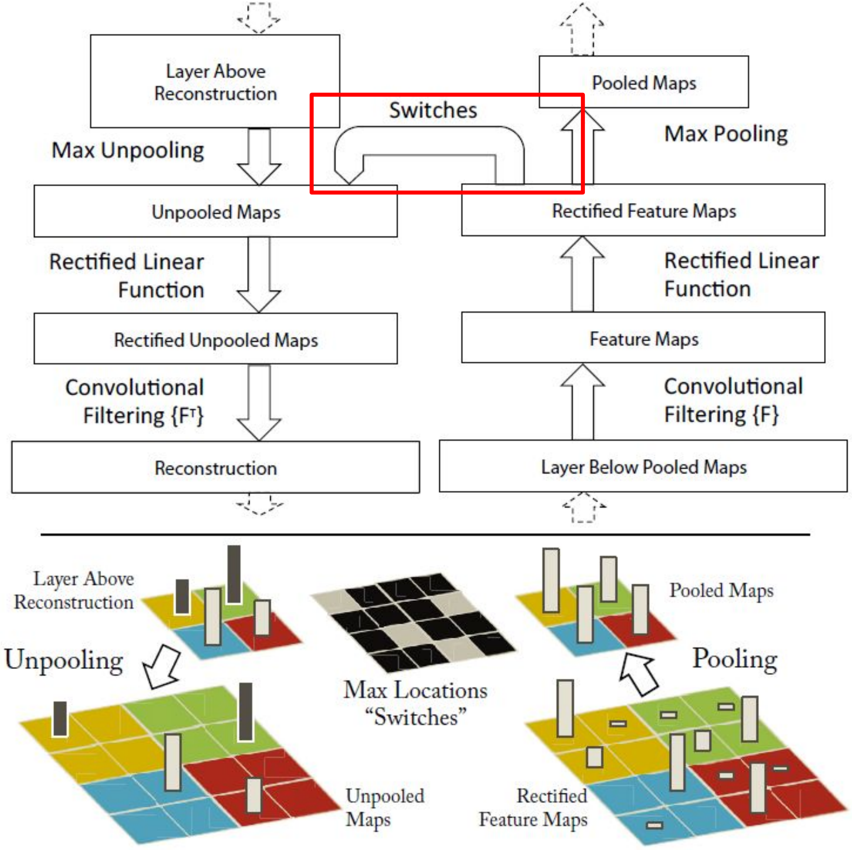
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

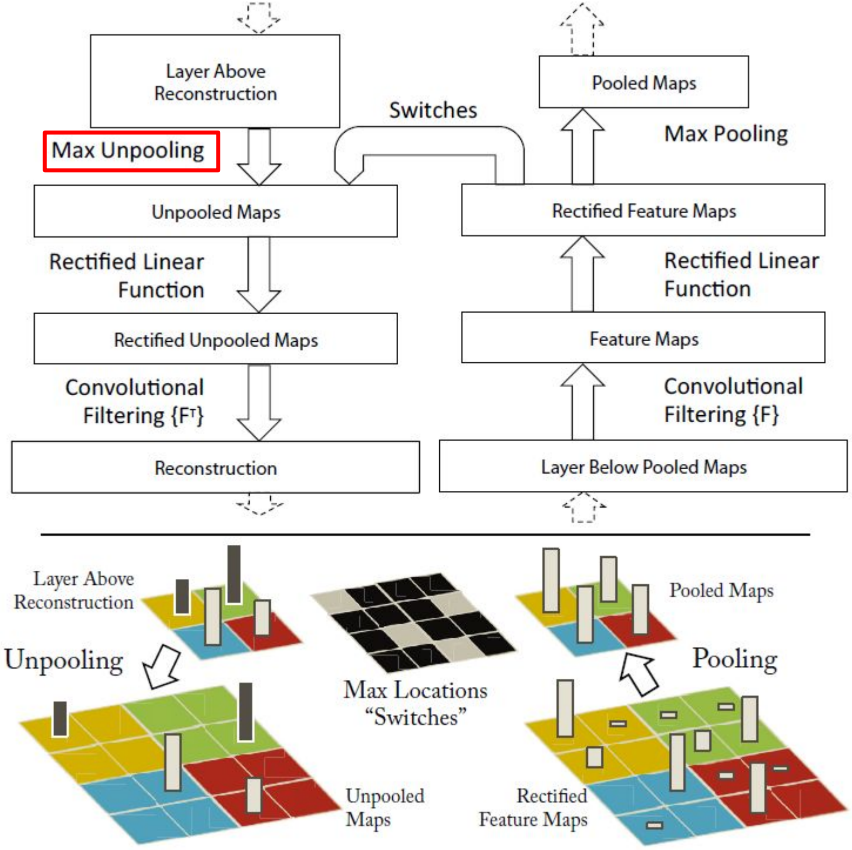
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

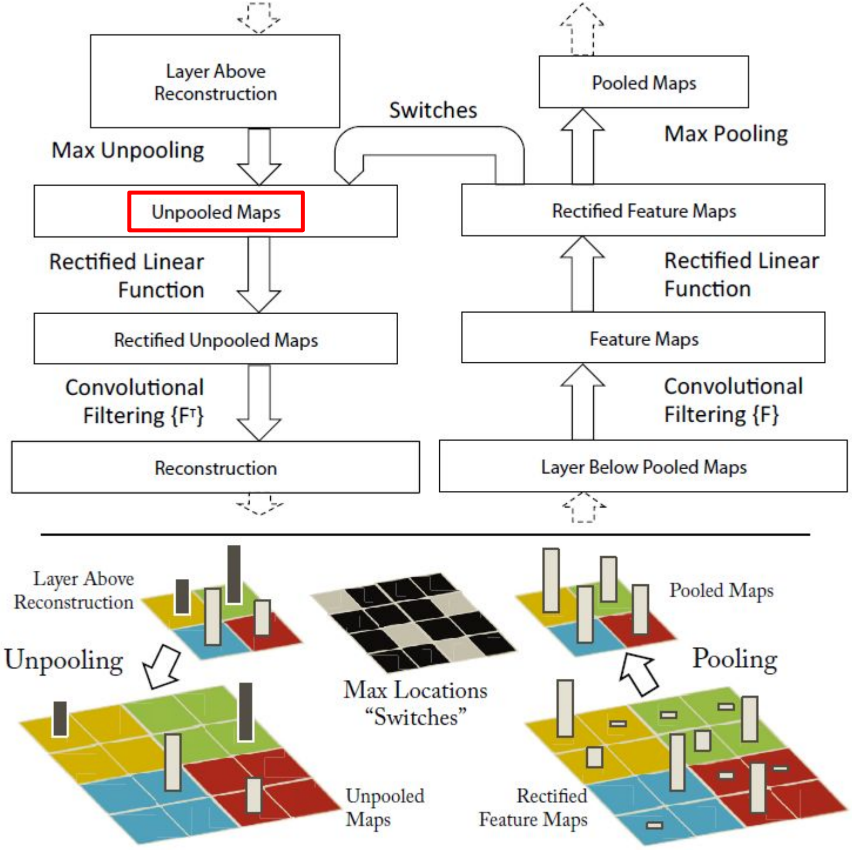
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

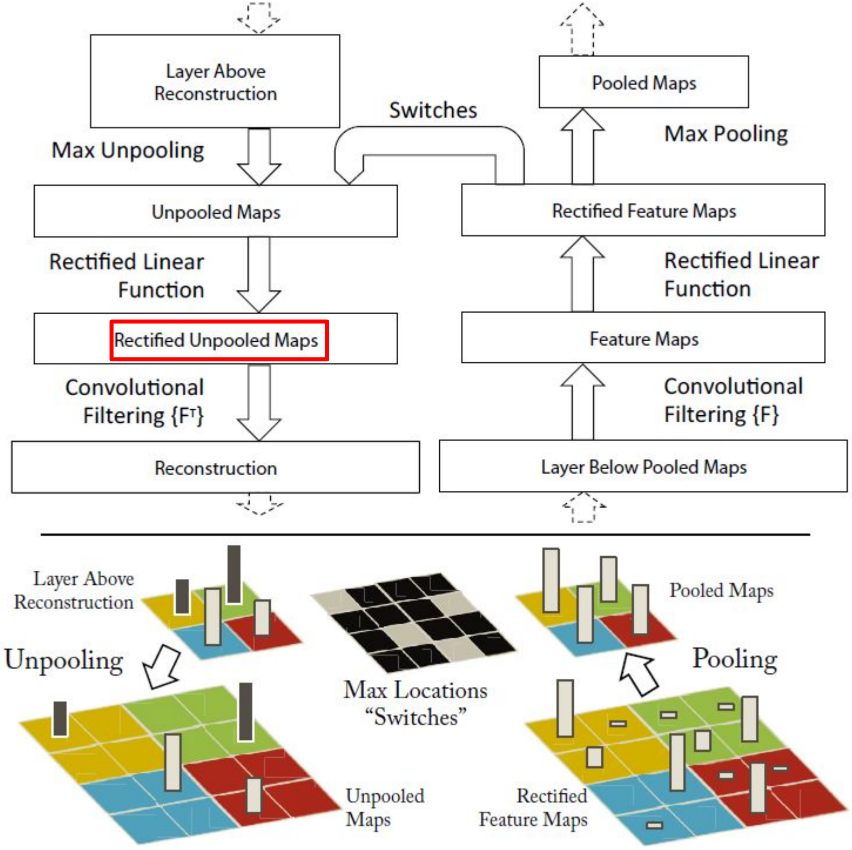
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

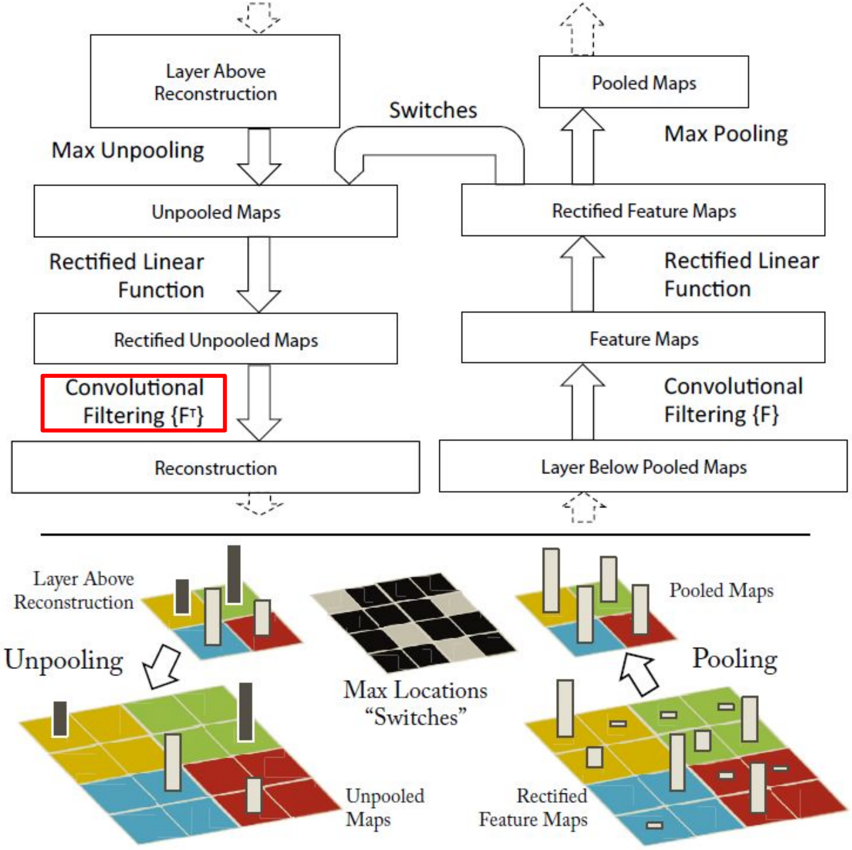
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

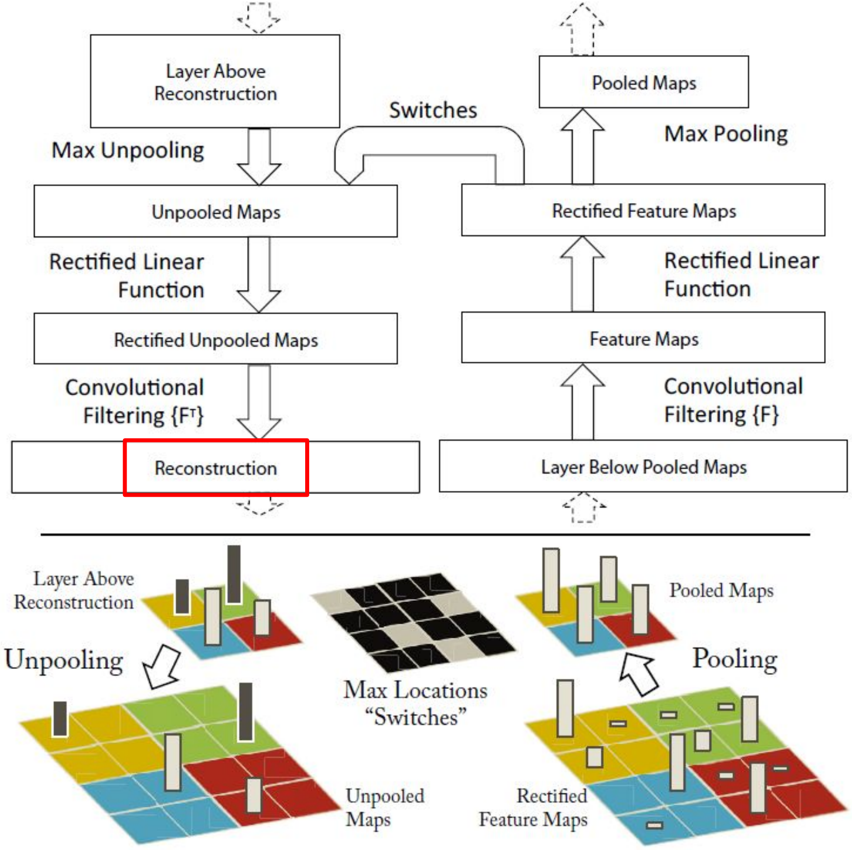
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

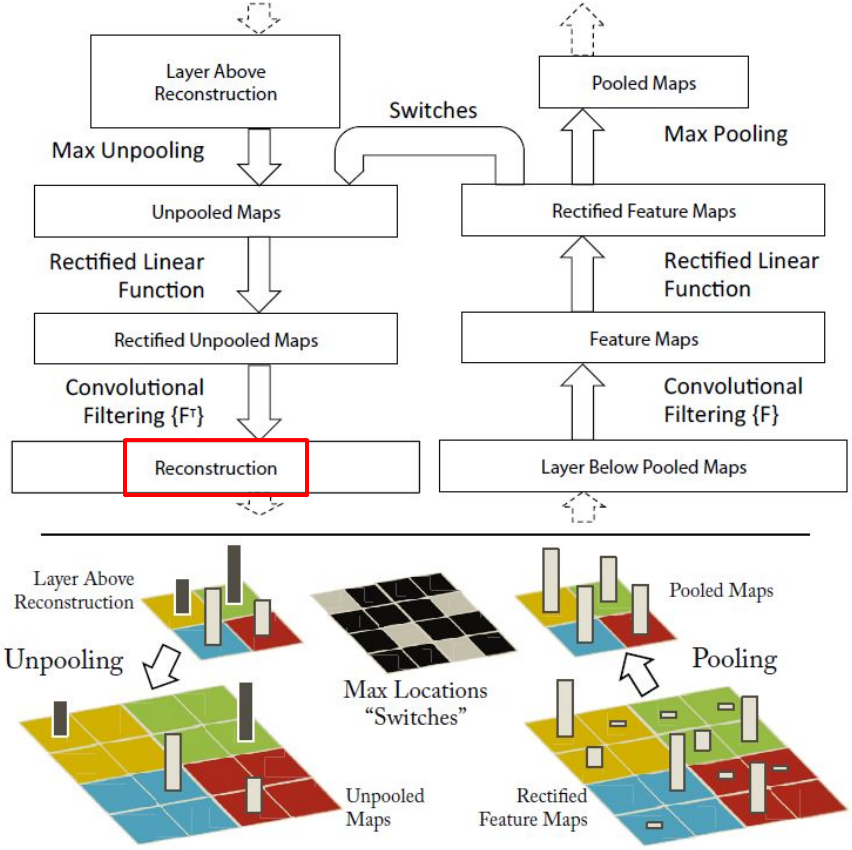
Deconv for visualization



https://arxiv.org/abs/1311.2901

ZFNet by Zeiler and Fergus (2013)

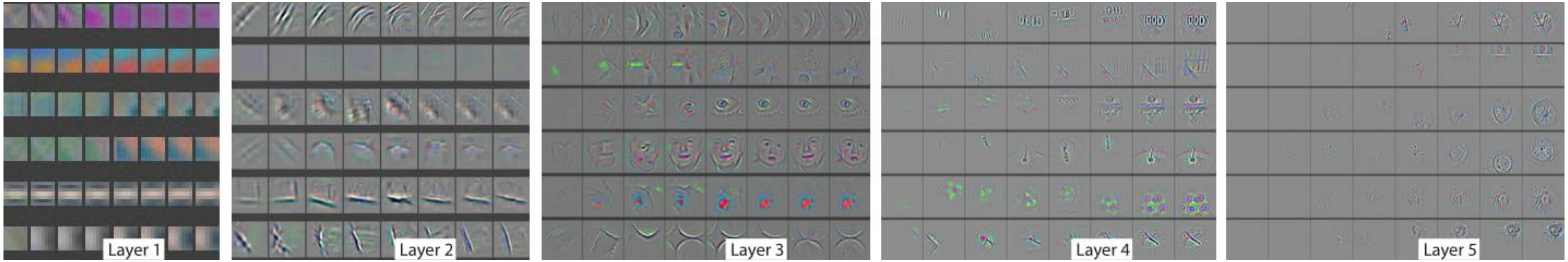
Deconv for visualization



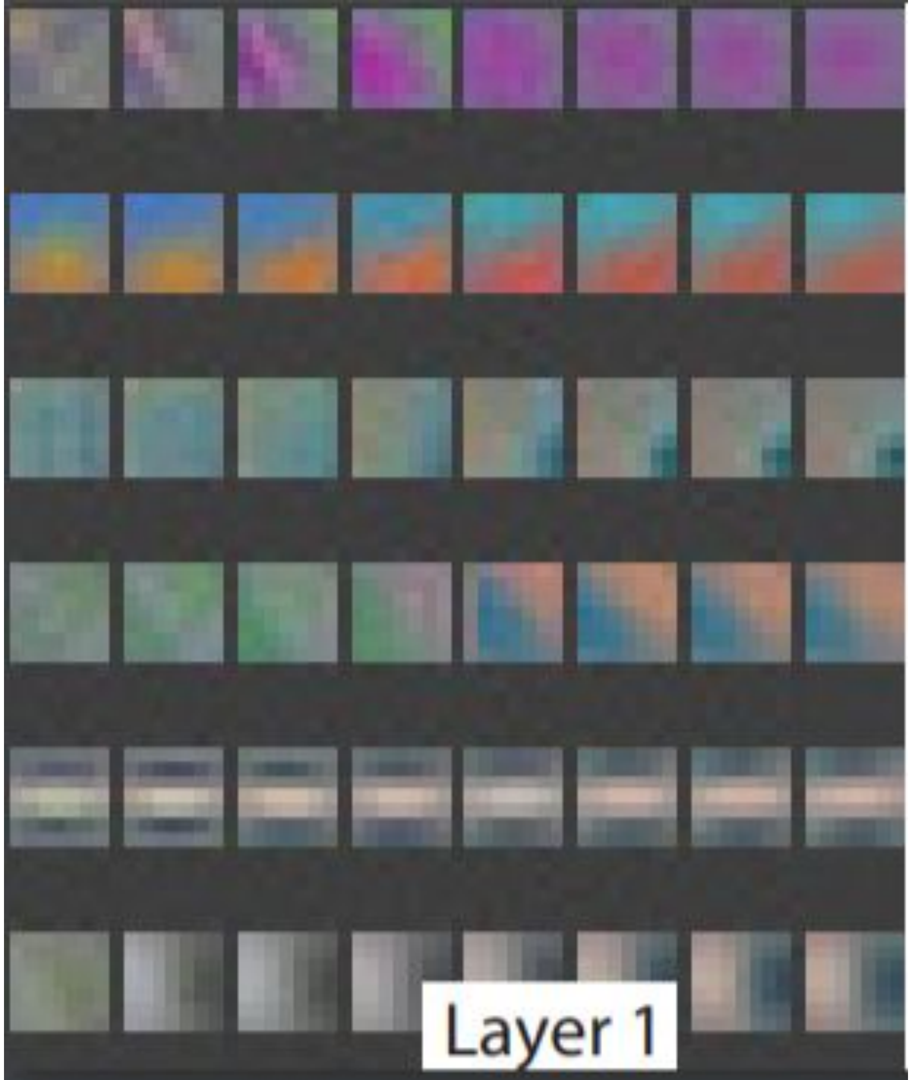
https://arxiv.org/abs/1311.2901

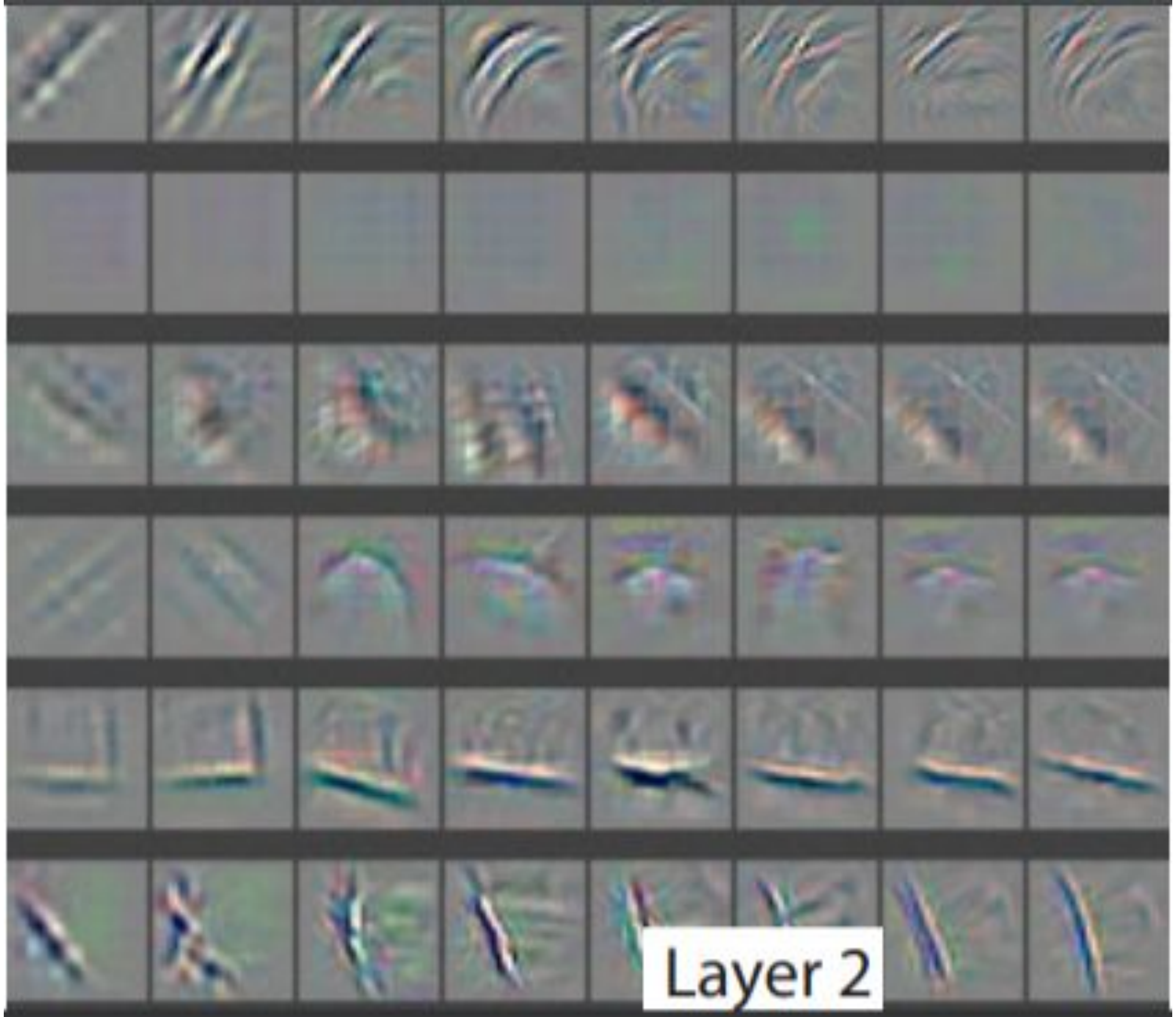
ZFNet by Zeiler and Fergus (2013)

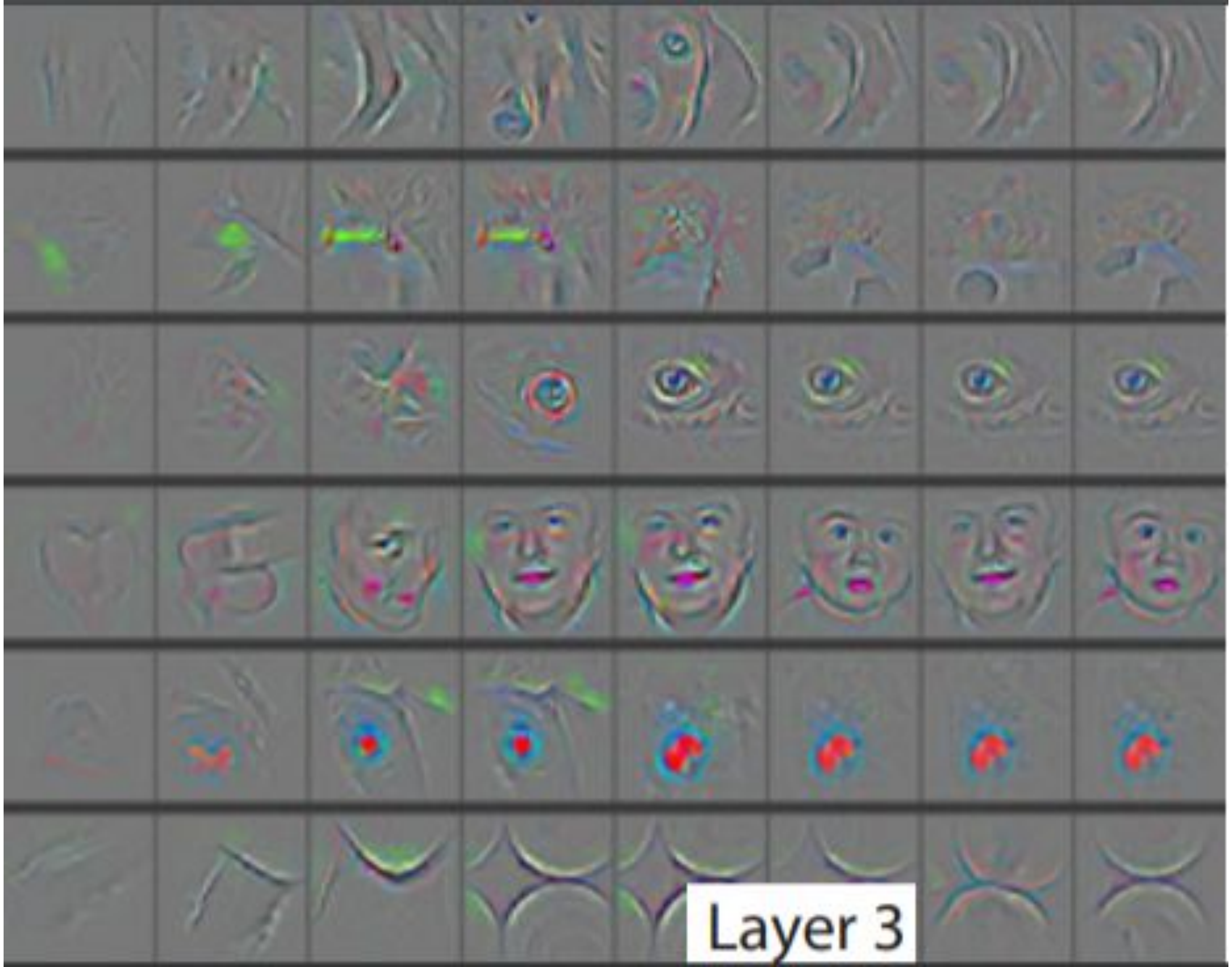
Visualizing AlexNet with Deconv

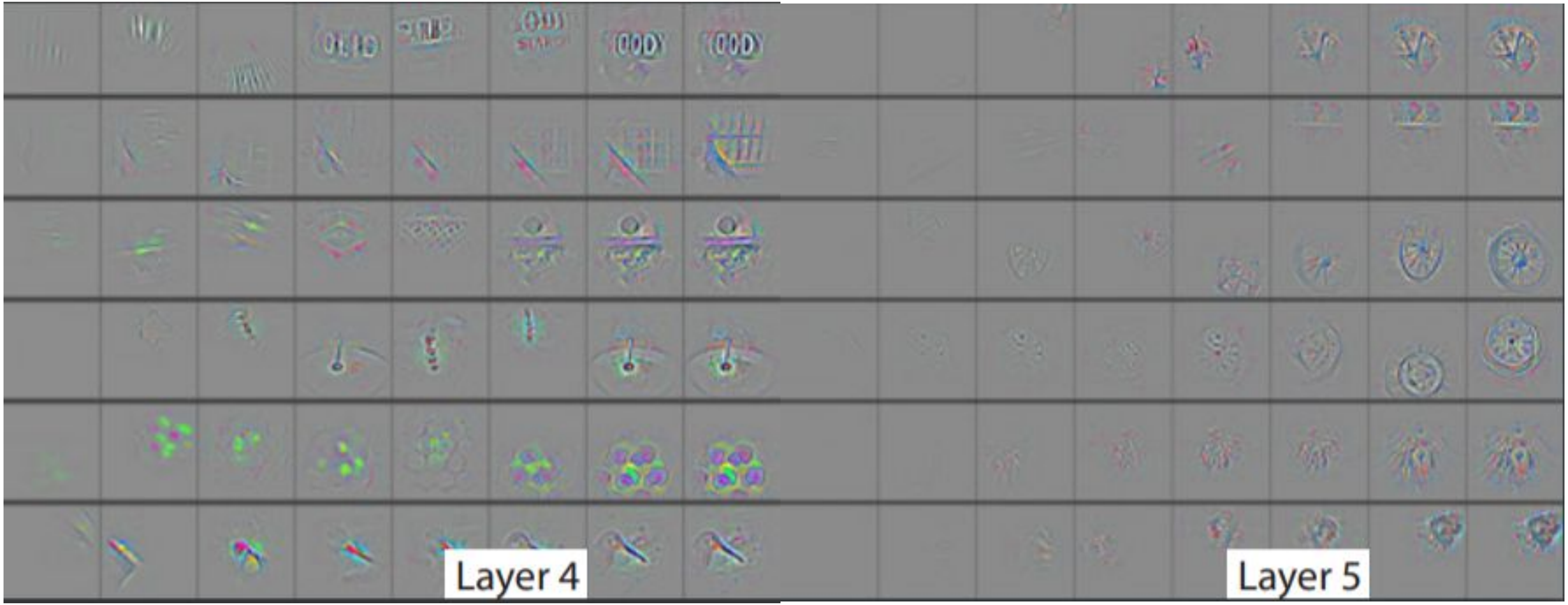


https://arxiv.org/abs/1311.2901

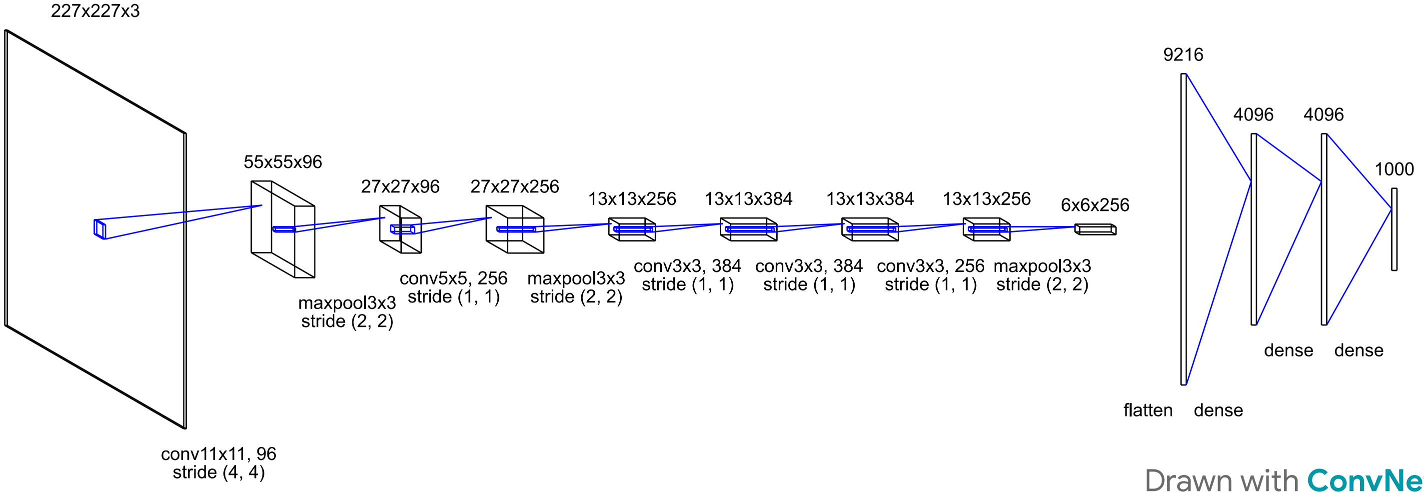

Layer 1


Layer 2

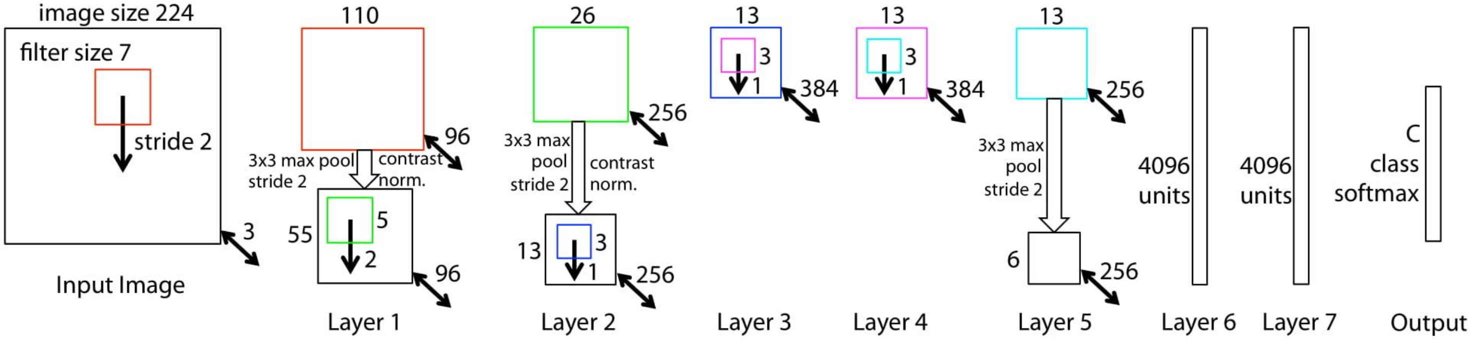

Layer 3



https://arxiv.org/abs/1311.2901

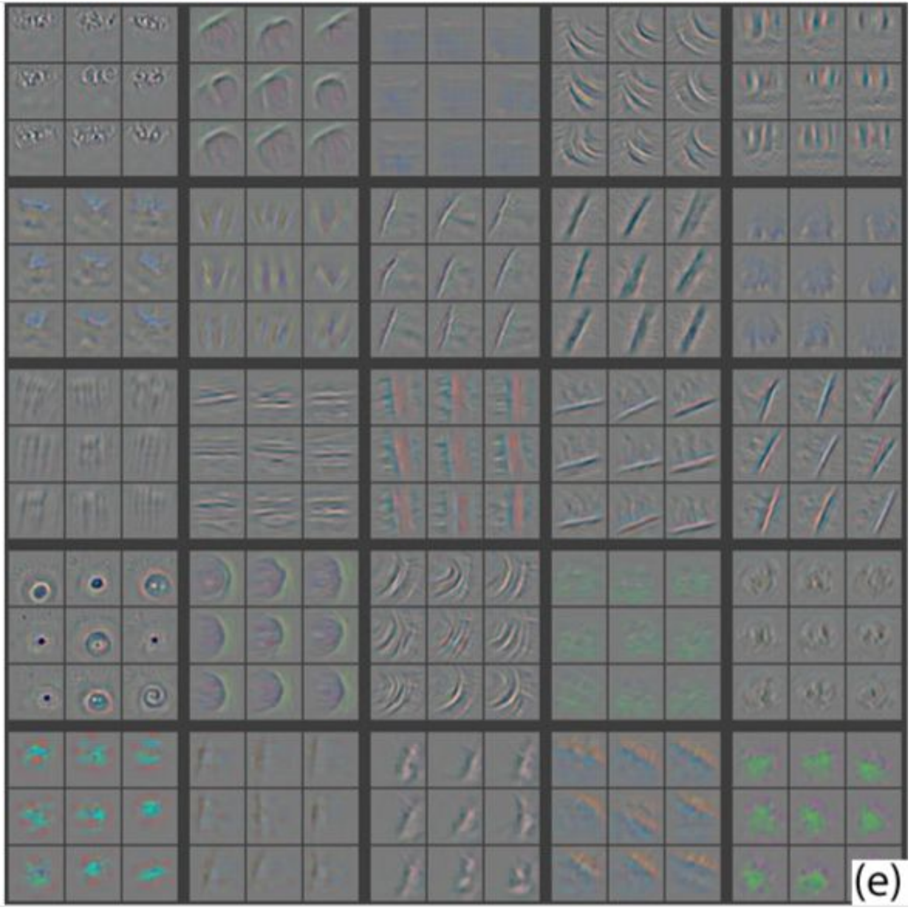


Drawn with **ConvNet Drawer**



ZFNet by Zeiler and Fergus (2013)

Layer 2 (AlexNet)



Layer 2 (ZFNet)