5. Rețele neuronale convoluționale

5.1 Operația de convoluție

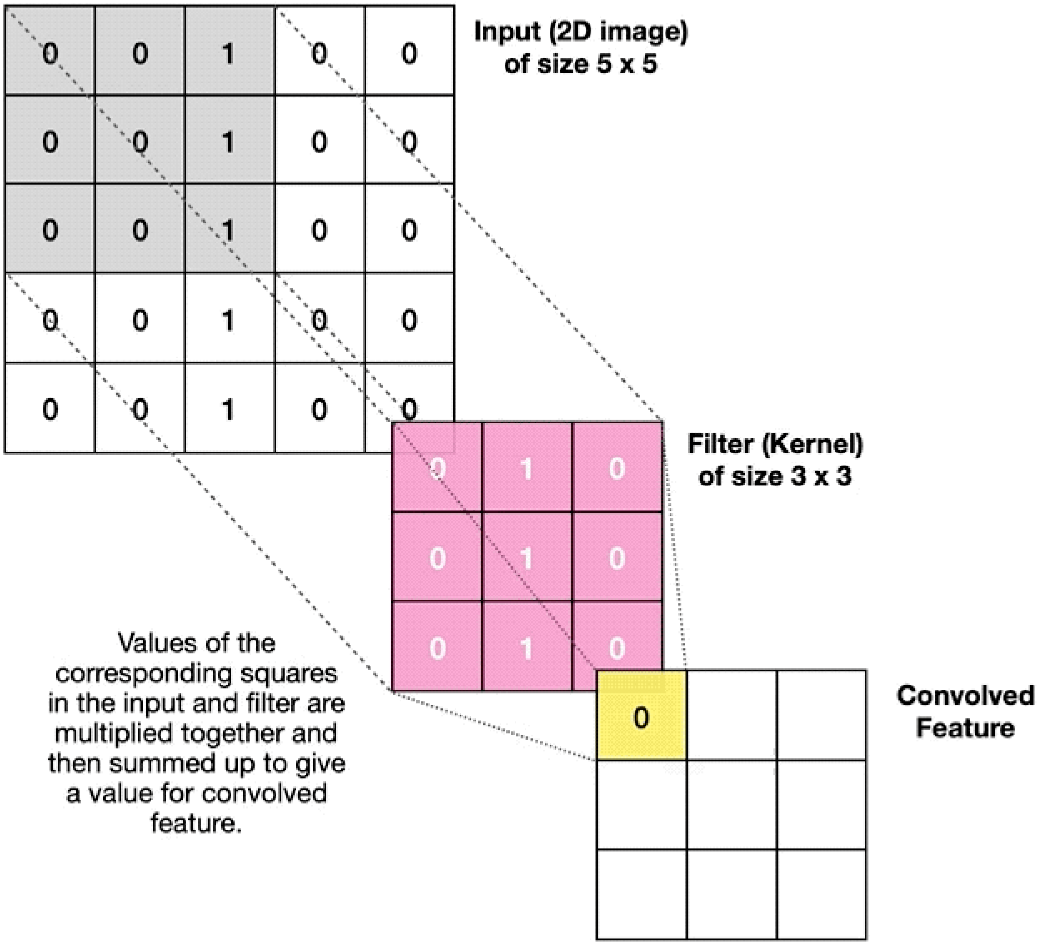


Fig. 5.1 Ilustrarea operației de convoluție (Dobilas, 2022)

5.2 Operația de subeșantionare

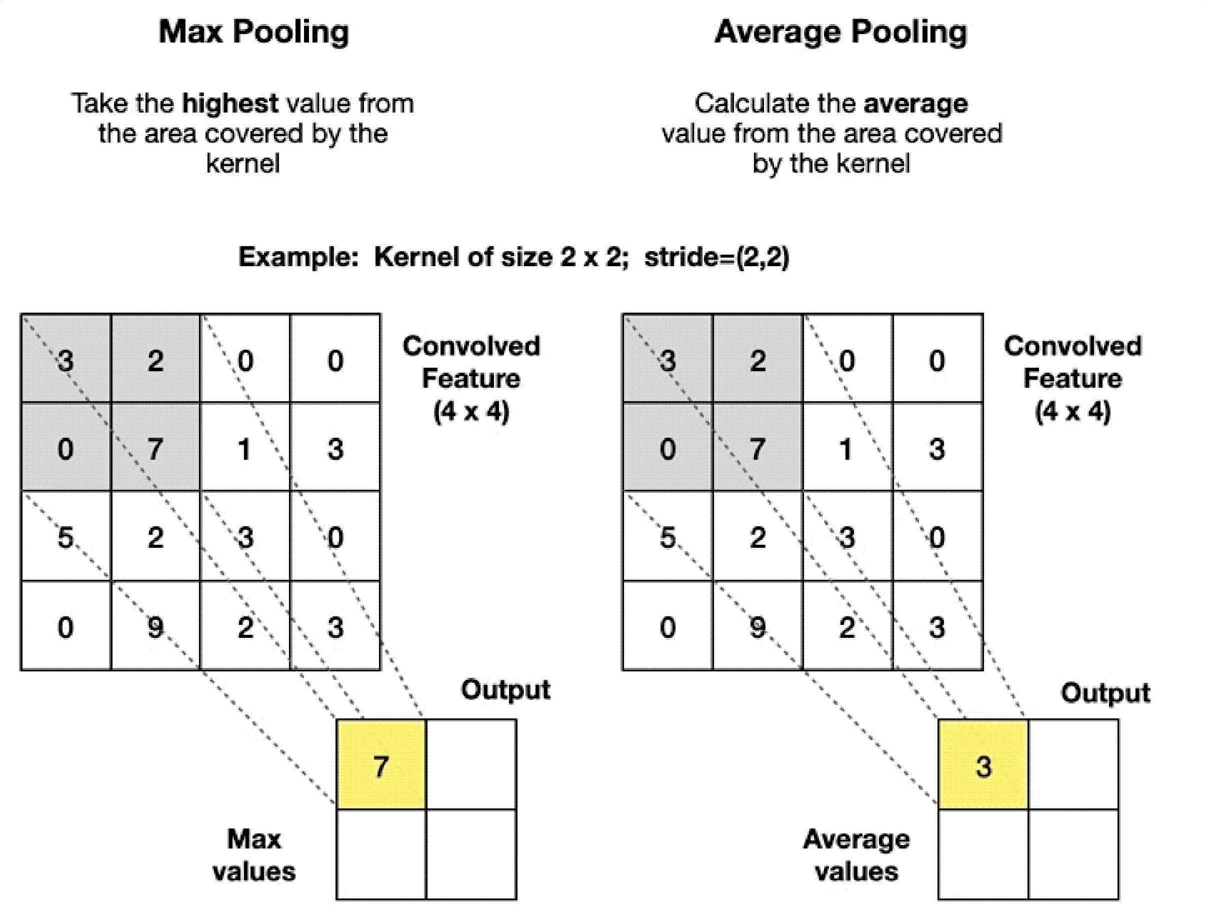


Fig. 5.2 Ilustrarea operației de subeșantionare (Dobilas, 2022)

5.3 Exemplificarea operației de convoluție

În continuare, vom exemplifica operația de convoluție folosind un strat convoluțional 2D.

input shape:

```
[samples, rows, columns, channels]
```

kernel shape:

```
[height, width, depth/input channels, output feature maps]
```

feature map shape:

```
[samples, rows, columns, filters]
```

```
import numpy as np
from keras import layers, models

input_image = np.zeros((8, 8), dtype='uint8')
input_image[:, 3:5] = 1
print(input_image)
```

```
[[0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]
 [0 0 0 1 1 0 0 0]]
```

```
input_image = input_image[np.newaxis, ..., np.newaxis]
print(input_image.shape)
```

```
(1, 8, 8, 1)
```

```
model = models.Sequential()
model.add(layers.Conv2D(filters=1, kernel_size=(3, 3),
                        input_shape=input_image.shape[1:]))

kernel = np.array([[0, 1, 0], [0, 1, 0], [0, 1, 0]])
print(kernel)
```

```
[[0 1 0]
 [0 1 0]
 [0 1 0]]
```

```
kernel = kernel[..., np.newaxis, np.newaxis]
print(kernel.shape)
```

```
(3, 3, 1, 1)
```

```
weights = [kernel, np.array([0])]

model.set_weights(weights)
print(model.get_weights())
```

```
[array([[[[0.]],

         [[1.]],

         [[0.]]],


        [[[0.]],

         [[1.]],

         [[0.]]],


        [[[0.]],

         [[1.]],

         [[0.]]]], dtype=float32), array([0.], dtype=float32)]
```

```
pred = model.predict(input_image)
print(pred.shape)
```

```
(1, 6, 6, 1)
```

```
for r in range(pred.shape[1]):
    print([pred[0, r, c, 0] for c in range(pred.shape[2])])
```

```
[[0. 0. 3. 3. 0. 0.]
 [0. 0. 3. 3. 0. 0.]
 [0. 0. 3. 3. 0. 0.]
 [0. 0. 3. 3. 0. 0.]
 [0. 0. 3. 3. 0. 0.]
 [0. 0. 3. 3. 0. 0.]]
```

5.4 Vizualizarea filtrelor și a hărților de caracteristici

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

VGG-16 model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
```

```
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

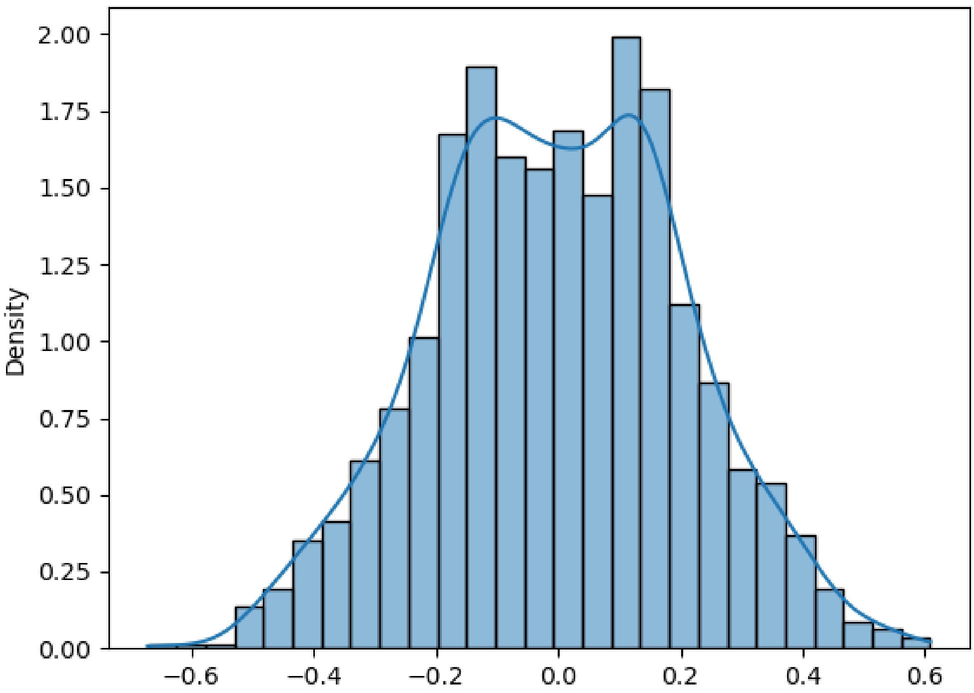
The layer indexes of the last convolutional layer in each block are [2, 5, 9, 13, 17].

Vizualizarea filtrelor

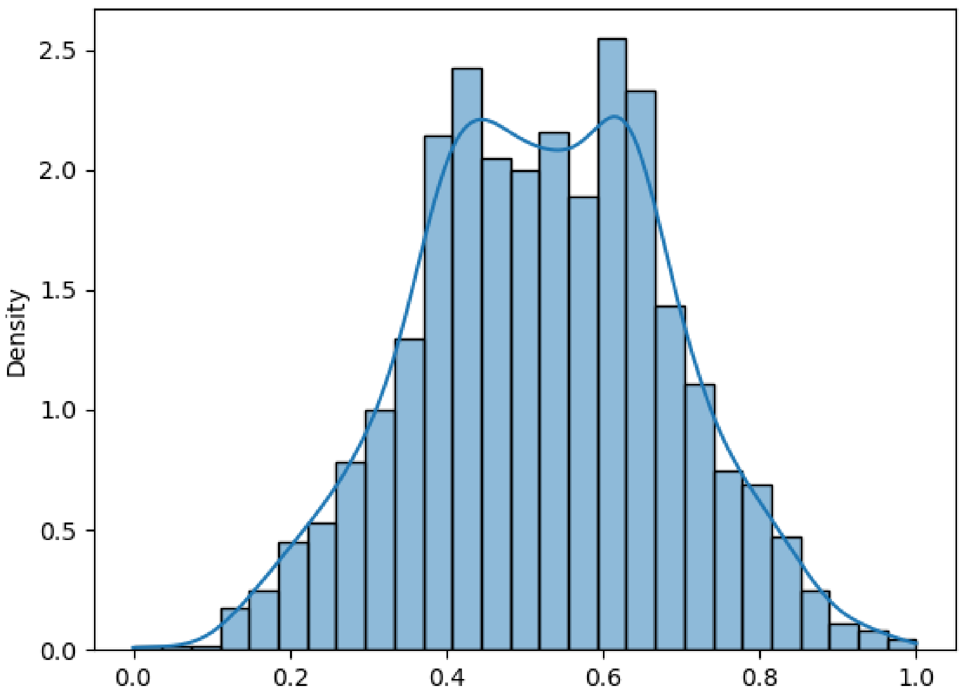
```python
from keras.applications.vgg16 import VGG16
import matplotlib.pyplot as plt
import seaborn as sns

model = VGG16(weights='imagenet', include_top=False)
model.summary()

for layer in model.layers:
    if 'conv' not in layer.name:
        continue
    filters, biases = layer.get_weights()
    print(layer.name, filters.shape)

filters, biases = model.layers[1].get_weights()
print(filters.shape)

sns.histplot(data=filters.flatten(), kde=True, stat='density')

f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)

sns.histplot(data=filters.flatten(), kde=True, stat='density')

n_filters = 6
fig, axs = plt.subplots(nrows=n_filters, ncols=3,
                        figsize=(6, 8))
for i in range(n_filters):
    f = filters[:, :, :, i]
    for j in range(3):
        axs[i, j].imshow(f[:, :, j], cmap='gray')
        axs[i, j].set_xticks([])
        axs[i, j].set_yticks([])
plt.show()
```

```
block1_conv1 (3, 3, 3, 64)
block1_conv2 (3, 3, 64, 64)
block2_conv1 (3, 3, 64, 128)
block2_conv2 (3, 3, 128, 128)
block3_conv1 (3, 3, 128, 256)
block3_conv2 (3, 3, 256, 256)
block3_conv3 (3, 3, 256, 256)
block4_conv1 (3, 3, 256, 512)
block4_conv2 (3, 3, 512, 512)
block4_conv3 (3, 3, 512, 512)
block5_conv1 (3, 3, 512, 512)
block5_conv2 (3, 3, 512, 512)
block5_conv3 (3, 3, 512, 512)
```

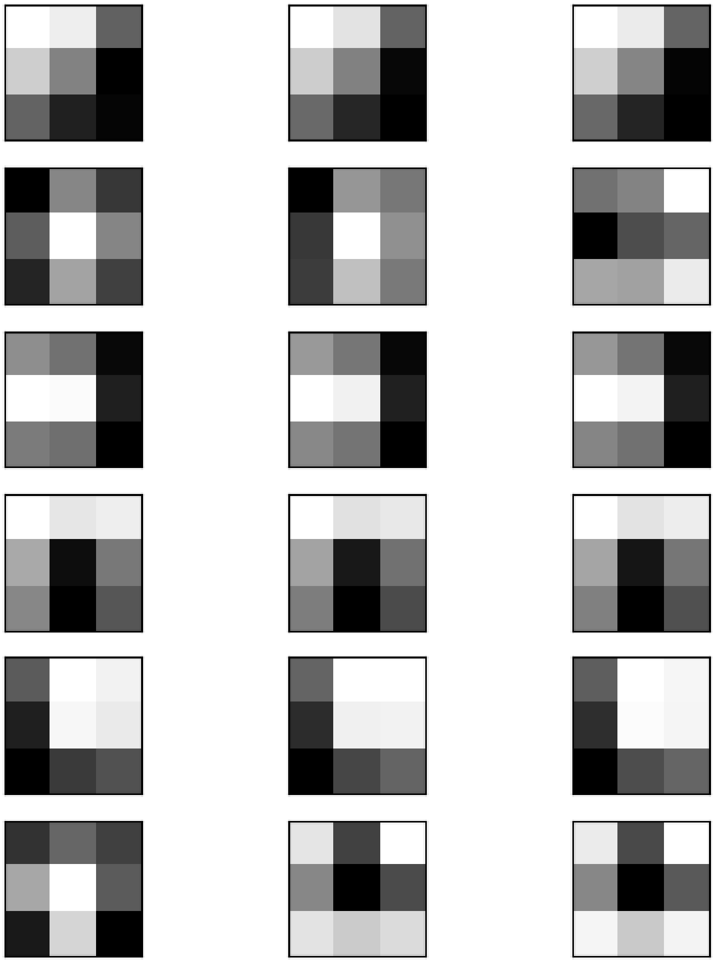
```
(3, 3, 3, 64)
```



Density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

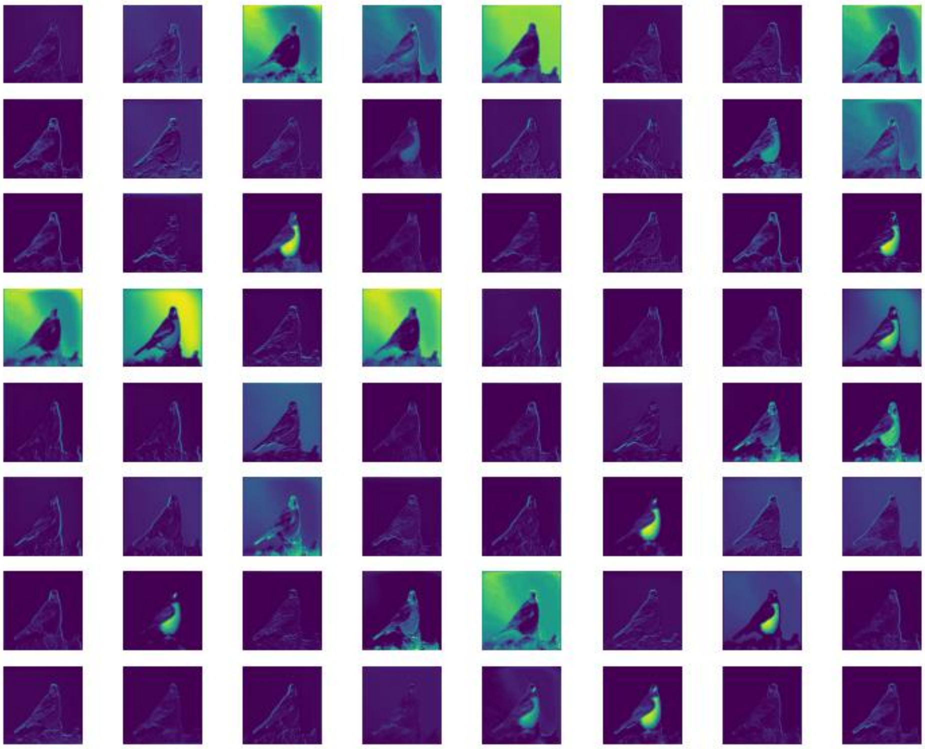


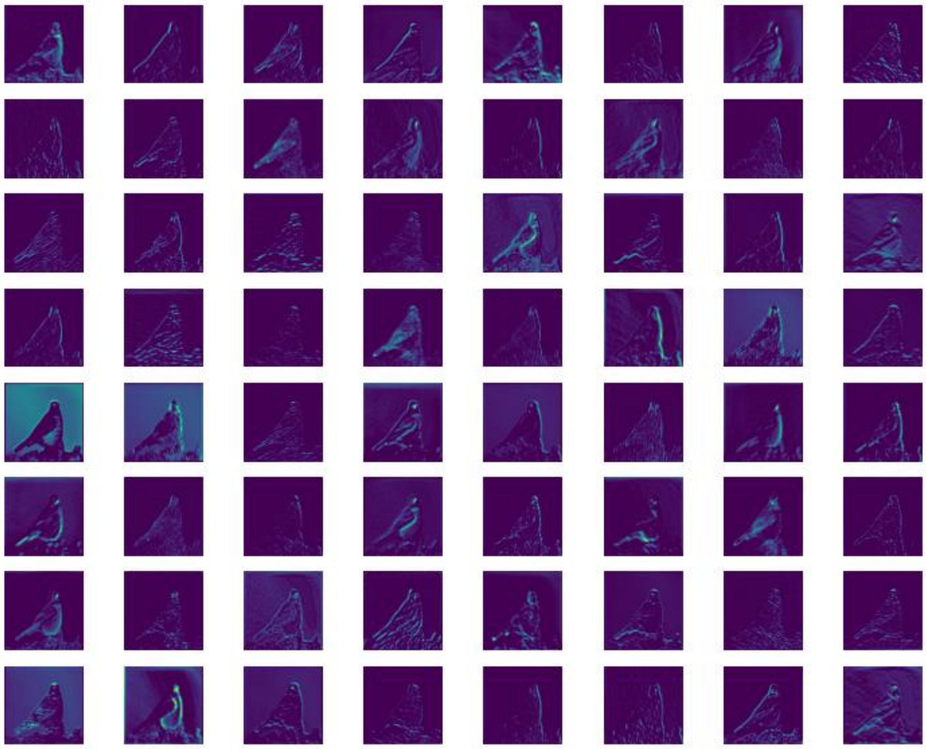
Density
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

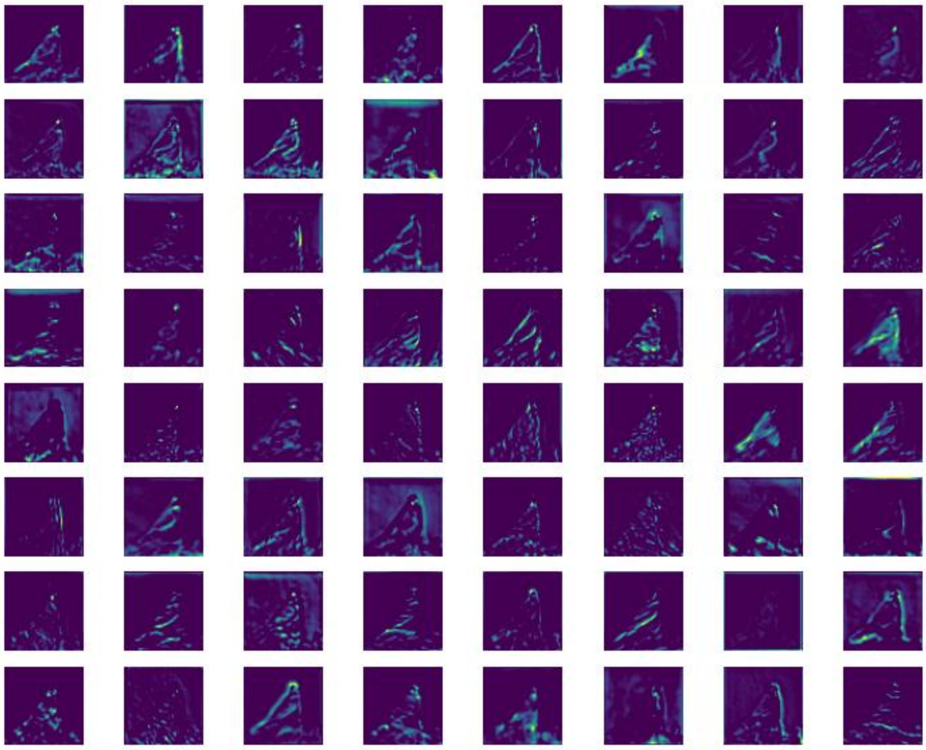


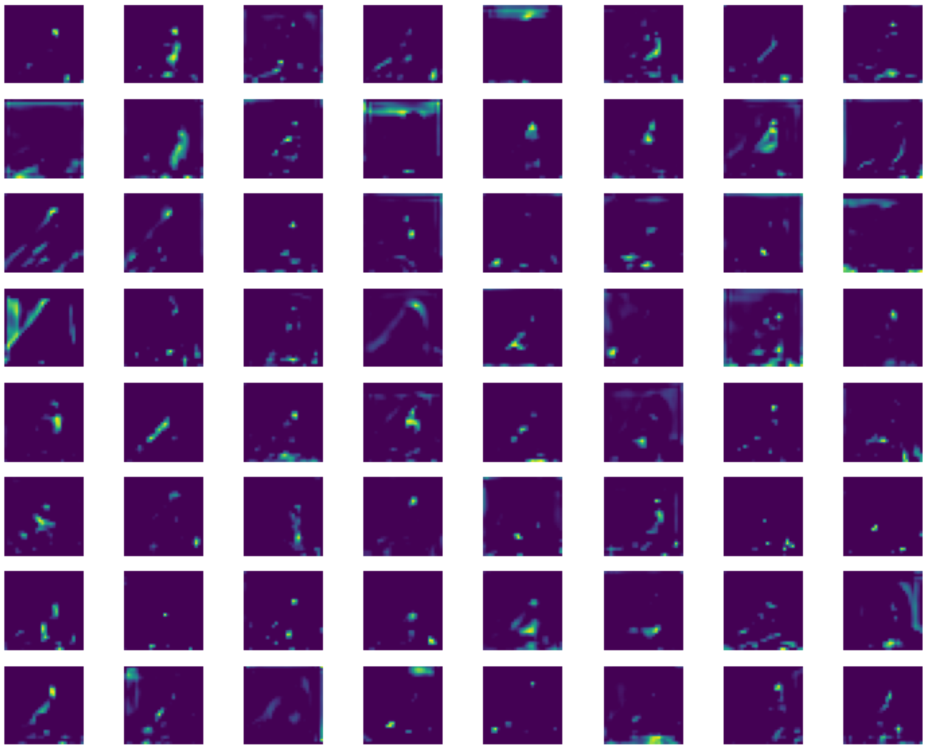
Vizualizarea hărților de caracteristici

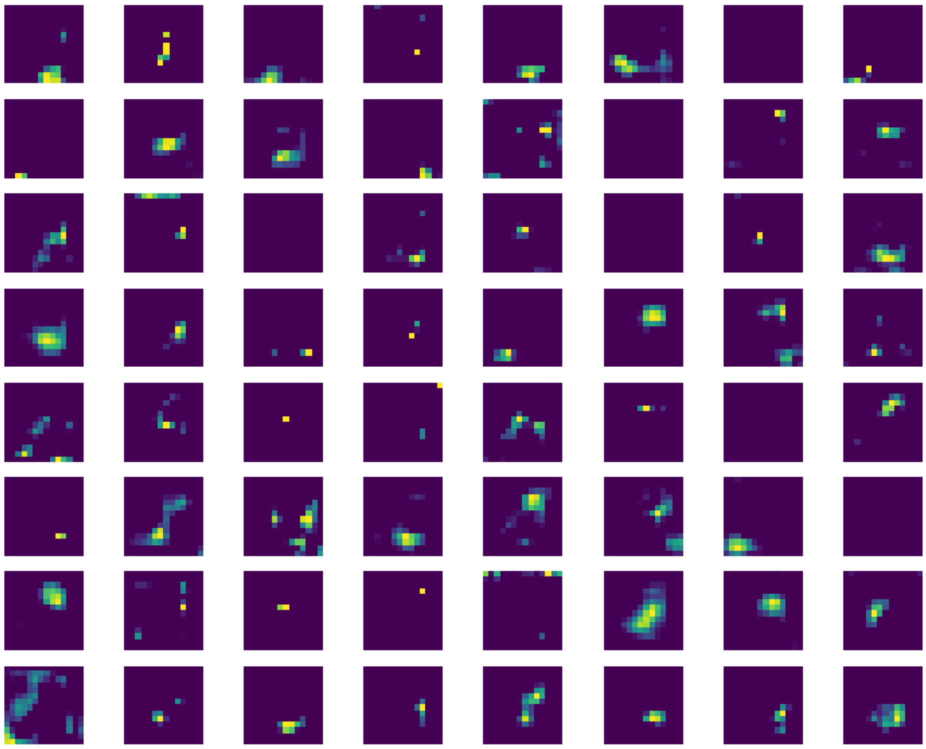
```
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
import matplotlib.pyplot as plt
import numpy as np

model = VGG16(weights='imagenet', include_top=False)

ixs = [2, 5, 9, 13, 17]
outputs = [model.layers[i].output for i in ixs]
model = Model(inputs=model.inputs, outputs=outputs)

img = load_img('bird.jpg', target_size=(224, 224))
img = img_to_array(img)

img = np.expand_dims(img, axis=0)
img = preprocess_input(img)

feature_maps = model.predict(img)

for fmap in feature_maps:
    n_maps = 64
    square = 8
    fig, axs = plt.subplots(nrows=square, ncols=square,
                            figsize=(9, 9))
    axs = axs.flatten()
    for i in range(n_maps):
        axs[i].imshow(fmap[0, :, :, i])
        axs[i].set_axis_off()
plt.show()
```











5.5 Convoluții separabile în adâncime

Convoluțiile separabile în adâncime au fost introduse de Sifre (2014) și au fost adoptate de arhitecturi de modele populare, cum ar fi MobileNet (Howard et al., 2017) și Xception (Chollet, 2017).

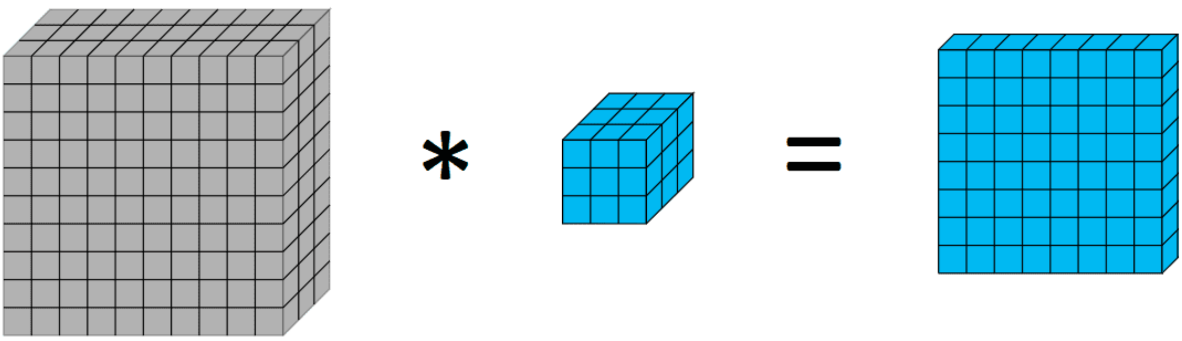


Fig. 5.3 O convoluție obișnuită (Chng, 2022)

Convoluția separabilă în adâncime este o convoluție la nivel de canal, urmată de o convoluție punctuală.

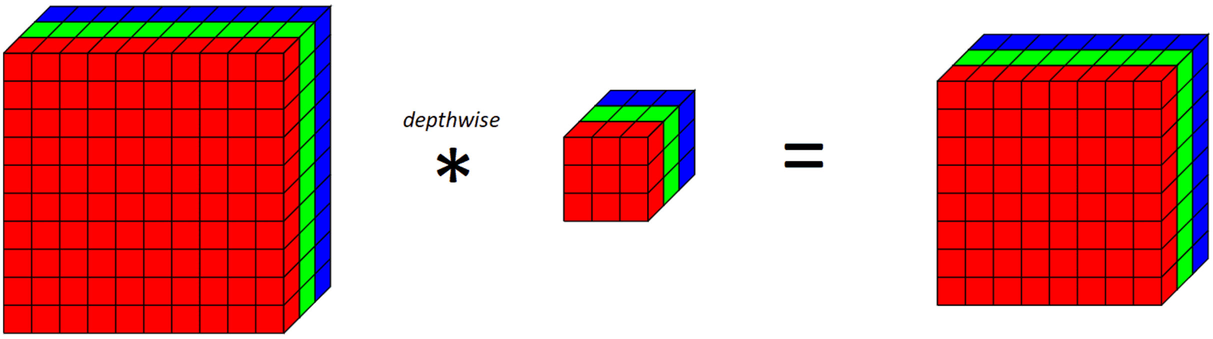


Fig. 5.4 Convoluția la nivel de canal (Chng, 2022)

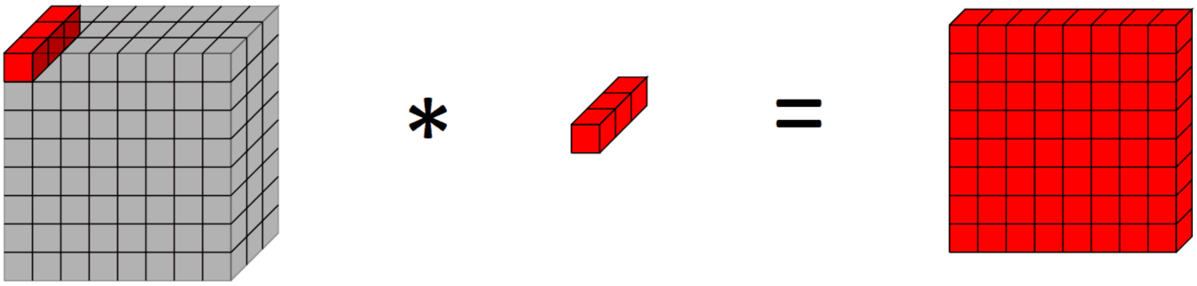


Fig. 5.5 Convoluția punctuală (Chng, 2022)

Să presupunem că dorim să aplicăm 64 de filtre convoluționale unei imagini RGB pentru a obține 64 de canale la ieșire.

Numărul de parametri în stratul convoluțional normal este $3 \times 3 \times 3 \times 64 + 64 = 1792$. Pe de altă parte, un strat convoluțional separabil în adâncime ar avea doar $(3 \times 3 \times 1 \times 3 + 3) + (1 \times 1 \times 3 \times 64 + 64) = 30 + 256 = 286$ de parametri, ceea ce reprezintă o reducere semnificativă.

Exemplu:

```
def vgg_block(layer_in, n_filters, n_conv):
    for _ in range(n_conv):
        layer_in = Conv2D(filters=n_filters, kernel_size=(3, 3),
                          padding='same', activation='relu')(layer_in)
    layer_in = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(layer_in)
    return layer_in
```

```python
visible = Input(shape=(32, 32, 3))
layer = vgg_block(visible, 64, 2)
layer = vgg_block(layer, 128, 2)
layer = vgg_block(layer, 256, 2)
layer = Flatten()(layer)
layer = Dense(units=10, activation='softmax')(layer)

model = Model(inputs=visible, outputs=layer)

model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        1792      
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        36928     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       73856     
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 128)       147584    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 128)         0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 256)         295168    
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         590080    
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 256)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 10)                40970     
=================================================================
Total params: 1,186,378
Trainable params: 1,186,378
Non-trainable params: 0
_________________________________________________________________
```

```python
def vgg_depthwise_block(layer_in, n_filters, n_conv):
   for _ in range(n_conv):
      layer_in = SeparableConv2D(filters=n_filters, kernel_size=(3, 3),
                                 padding='same', activation='relu')(layer_in)
   layer_in = MaxPooling2D(pool_size=(2,2), strides=(2,2))(layer_in)
   return layer_in
```

```
visible = Input(shape=(32, 32, 3))
layer = vgg_depthwise_block(visible, 64, 2)
layer = vgg_depthwise_block(layer, 128, 2)
layer = vgg_depthwise_block(layer, 256, 2)
layer = Flatten()(layer)
layer = Dense(units=10, activation='softmax')(layer)

model = Model(inputs=visible, outputs=layer)

model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         (None, 32, 32, 3)         0         
_________________________________________________________________
separable_conv2d_1 (Separabl (None, 32, 32, 64)        283       
_________________________________________________________________
separable_conv2d_2 (Separabl (None, 32, 32, 64)        4736      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 64)        0         
_________________________________________________________________
separable_conv2d_3 (Separabl (None, 16, 16, 128)       8896      
_________________________________________________________________
separable_conv2d_4 (Separabl (None, 16, 16, 128)       17664     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 128)         0         
_________________________________________________________________
separable_conv2d_5 (Separabl (None, 8, 8, 256)         34176     
_________________________________________________________________
separable_conv2d_6 (Separabl (None, 8, 8, 256)         68096     
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 256)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                40970     
=================================================================
Total params: 174,821
Trainable params: 174,821
Non-trainable params: 0
_________________________________________________________________
```