**Car/Truck Classification with CNN and Maximum Pooling**

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from matplotlib import gridspec

import learntools.computer\_vision.visiontools as visiontools

plt.rc('figure', autolayout=True)

plt.rc('axes', labelweight='bold', labelsize='large',

titleweight='bold', titlesize=18, titlepad=10)

plt.rc('image', cmap='magma')

*# Read image*

image\_path = '../input/computer-vision-resources/car\_illus.jpg'

image = tf.io.read\_file(image\_path)

image = tf.io.decode\_jpeg(image, channels=1)

image = tf.image.resize(image, size=[400, 400])

*# Embossing kernel*

kernel = tf.constant([

[-2, -1, 0],

[-1, 1, 1],

[0, 1, 2],

])

*# Reformat for batch compatibility.*

image = tf.image.convert\_image\_dtype(image, dtype=tf.float32)

image = tf.expand\_dims(image, axis=0)

kernel = tf.reshape(kernel, [\*kernel.shape, 1, 1])

kernel = tf.cast(kernel, dtype=tf.float32)

image\_filter = tf.nn.conv2d(

input=image,

filters=kernel,

strides=1,

padding='VALID',

)

image\_detect = tf.nn.relu(image\_filter)

*# Show what we have so far*

plt.figure(figsize=(12, 6))

plt.subplot(131)

plt.imshow(tf.squeeze(image), cmap='gray')

plt.axis('off')

plt.title('Input')

plt.subplot(132)

plt.imshow(tf.squeeze(image\_filter))

plt.axis('off')

plt.title('Filter')

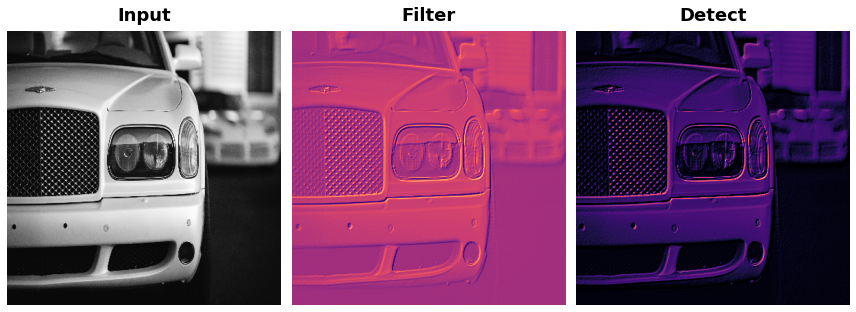
plt.subplot(133)

plt.imshow(tf.squeeze(image\_detect))

plt.axis('off')

plt.title('Detect')

plt.show();



1) Apply Pooling to Condense

For for the last step in the sequence, apply maximum pooling using a 2×22×2 pooling window. You can copy this code to get started:

*# YOUR CODE HERE*

image\_condense = tf.nn.pool(

input=image\_detect,

window\_shape=(2,2),

pooling\_type='MAX',

strides=(2, 2),

padding='SAME',

)

Run the next cell to see what maximum pooling did to the feature!



We learned about how MaxPool2D layers give a convolutional network the property of **translation invariance** over small distances. In this exercise, you'll have a chance to observe this in action.

This next code cell will randomly apply a small shift to a circle and then condense the image several times with maximum pooling. Run the cell once and make note of the image that results at the end.

REPEATS = 4

SIZE = [64, 64]

*# Create a randomly shifted circle*

image = visiontools.circle(SIZE, r\_shrink=4, val=1)

image = tf.expand\_dims(image, axis=-1)

image = visiontools.random\_transform(image, jitter=3, fill\_method='replicate')

image = tf.squeeze(image)

plt.figure(figsize=(16, 4))

plt.subplot(1, REPEATS+1, 1)

plt.imshow(image, vmin=0, vmax=1)

plt.title("Original**\n**Shape: **{}**x**{}**".format(image.shape[0], image.shape[1]))

plt.axis('off')

*# Now condense with maximum pooling several times*

for i **in** range(REPEATS):

ax = plt.subplot(1, REPEATS+1, i+2)

image = tf.reshape(image, [1, \*image.shape, 1])

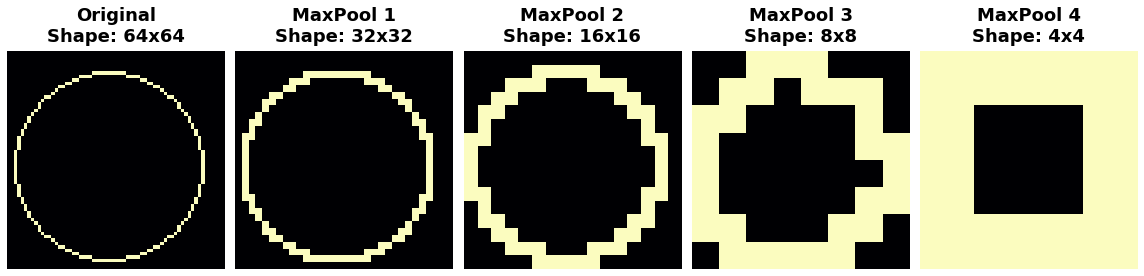
image = tf.nn.pool(image, window\_shape=(2,2), strides=(2, 2), padding='SAME', pooling\_type='MAX')

image = tf.squeeze(image)

plt.imshow(image, vmin=0, vmax=1)

plt.title("MaxPool **{}\n**Shape: **{}**x**{}**".format(i+1, image.shape[0], image.shape[1]))

plt.axis('off')



# 2) Explore Invariance

Suppose you had made a small shift in a different direction -- what effect would you expect that have on the resulting image? Try running the cell a few more times, if you like, to get a new random shift.

Solution: In the tutorial, we talked about how maximum pooling creates **translation invariance** over small distances. This means that we would expect small shifts to disappear after repeated maximum pooling. If you run the cell multiple times, you can see the resulting image is always the same; the pooling operation destroys those small translations.

# Global Average Pooling

We mentioned in the previous exercise that average pooling has largely been superceeded by maximum pooling within the convolutional base. There is, however, a kind of average pooling that is still widely used in the head of a convnet. This is **global average pooling**. A GlobalAvgPool2D layer is often used as an alternative to some or all of the hidden Dense layers in the head of the network, like so:

model = keras.Sequential([

pretrained\_base,

layers.GlobalAvgPool2D(),

layers.Dense(1, activation='sigmoid'),

])

What is this layer doing? Notice that we no longer have the Flatten layer that usually comes after the base to transform the 2D feature data to 1D data needed by the classifier. Now the GlobalAvgPool2D layer is serving this function. But, instead of "unstacking" the feature (like Flatten), it simply replaces the entire feature map with its average value. Though very destructive, it often works quite well and has the advantage of reducing the number of parameters in the model.

Let's look at what GlobalAvgPool2D does on some randomly generated feature maps. This will help us to understand how it can "flatten" the stack of feature maps produced by the base.

Run this next cell a few times until you get a feel for how this new layer works.

feature\_maps = [visiontools.random\_map([5, 5], scale=0.1, decay\_power=4) for \_ **in** range(8)]

gs = gridspec.GridSpec(1, 8, wspace=0.01, hspace=0.01)

plt.figure(figsize=(18, 2))

for i, feature\_map **in** enumerate(feature\_maps):

plt.subplot(gs[i])

plt.imshow(feature\_map, vmin=0, vmax=1)

plt.axis('off')

plt.suptitle('Feature Maps', size=18, weight='bold', y=1.1)

plt.show()

*# reformat for TensorFlow*

feature\_maps\_tf = [tf.reshape(feature\_map, [1, \*feature\_map.shape, 1])

for feature\_map **in** feature\_maps]

global\_avg\_pool = tf.keras.layers.GlobalAvgPool2D()

pooled\_maps = [global\_avg\_pool(feature\_map) for feature\_map **in** feature\_maps\_tf]

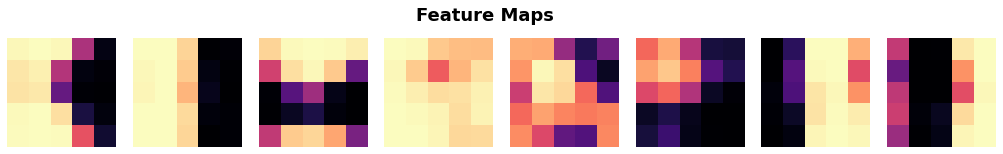
img = np.array(pooled\_maps)[:,:,0].T

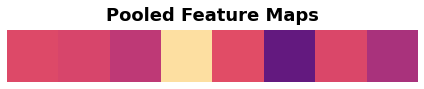
plt.imshow(img, vmin=0, vmax=1)

plt.axis('off')

plt.title('Pooled Feature Maps')

plt.show();





Let's pass some images from our Car or Truck dataset through VGG16 and examine the features that result after pooling. First run this cell to define the model and load the dataset.

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

*# Load VGG16*

pretrained\_base = tf.keras.models.load\_model(

'../input/cv-course-models/cv-course-models/vgg16-pretrained-base',

)

model = keras.Sequential([

pretrained\_base,

*# Attach a global average pooling layer after the base*

layers.GlobalAvgPool2D(),

])

*# Load dataset*

ds = image\_dataset\_from\_directory(

'../input/car-or-truck/train',

labels='inferred',

label\_mode='binary',

image\_size=[128, 128],

interpolation='nearest',

batch\_size=1,

shuffle=True,

)

ds\_iter = iter(ds)

Notice how we've attached a GlobalAvgPool2D layer after the pretrained VGG16 base. Ordinarily, VGG16 will produce 512 feature maps for each image. The GlobalAvgPool2D layer reduces each of these to a single value, an "average pixel", if you like.

This next cell will run an image from the Car or Truck dataset through VGG16 and show you the 512 average pixels created by GlobalAvgPool2D. Run the cell a few times and observe the pixels produced by cars versus the pixels produced by trucks.

car = next(ds\_iter)

car\_tf = tf.image.resize(car[0], size=[128, 128])

car\_features = model(car\_tf)

car\_features = tf.reshape(car\_features, shape=(16, 32))

label = int(tf.squeeze(car[1]).numpy())

plt.figure(figsize=(8, 4))

plt.subplot(121)

plt.imshow(tf.squeeze(car[0]))

plt.axis('off')

plt.title(["Car", "Truck"][label])

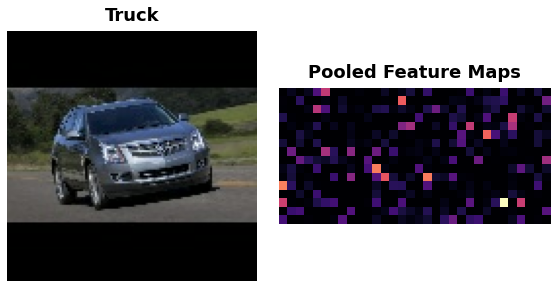
plt.subplot(122)

plt.imshow(car\_features)

plt.title('Pooled Feature Maps')

plt.axis('off')

plt.show();



# 3) Understand the Pooled Features

What do you see? Are the pooled features for cars and trucks different enough to tell them apart? How would you interpret these pooled values? How could this help the classification?

The VGG16 base produces 512 feature maps. We can think of each feature map as representing some high-level visual feature in the original image -- maybe a wheel or window. Pooling a map gives us a single number, which we could think of as a score for that feature: large if the feature is present, small if it is absent. Cars tend to score high with one set of features, and Trucks score high with another. Now, instead of trying to map raw features to classes, the head only has to work with these scores that GlobalAvgPool2D produced, a much easier problem for it to solve.

Global average pooling is often used in modern convnets. One big advantage is that it greatly reduces the number of parameters in a model, while still telling you if some feature was present in an image or not -- which for classification is usually all that matters. If you're creating a convolutional classifier it's worth trying out!

# Conclusion

In this lesson we explored the final operation in the feature extraction process: **condensing** with **maximum pooling**. Pooling is one of the essential features of convolutional networks and helps provide them with some of their characteristic advantages: efficiency with visual data, reduced parameter size compared to dense networks, translation invariance. We've seen that it's used not only in the base during feature extraction, but also can be used in the head during classification. Understanding it is essential to a full understanding of convnets.

# Keep Going

In the next lesson, we'll conclude our discussion of the feature extraction operations with **sliding windows**, the typical way of describing how the convolution and pooling operations scan over an image. We'll describe here the final two parameters in the Conv2D and MaxPool2D layers: strides and padding.