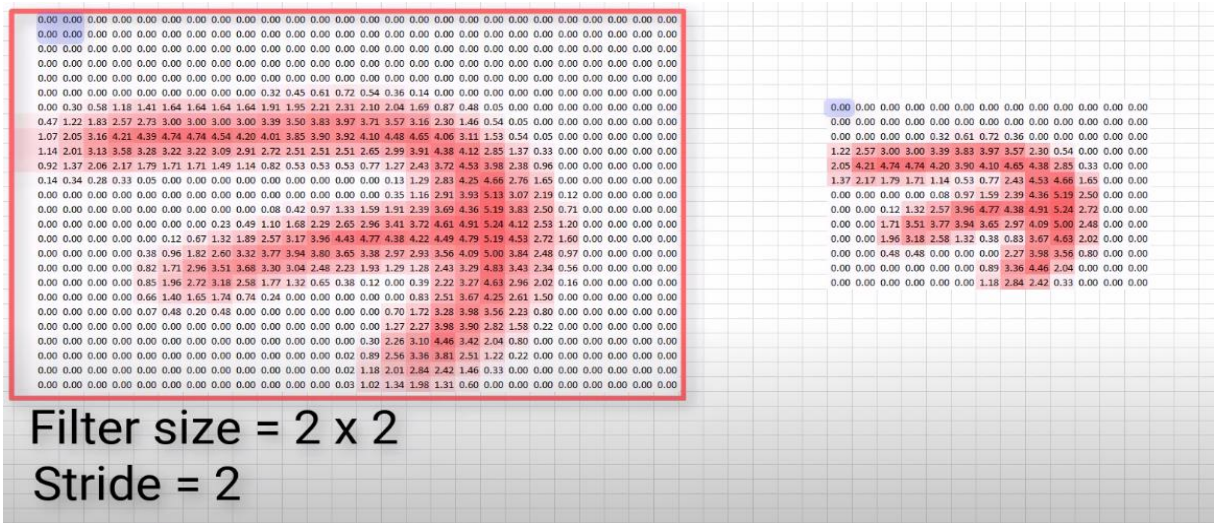
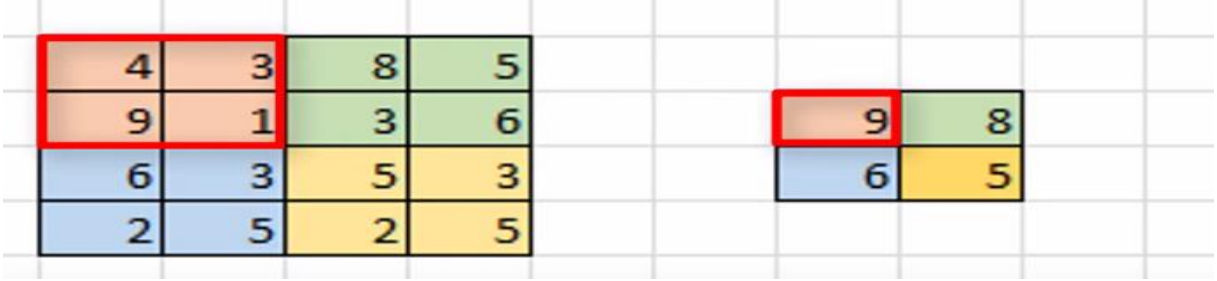
Elméleti Kiegészítések

Kicsit jobban utána szerettem volna járni, hogy pontosan mi történik az egyes folyamatok hátterében, szerettem volna jobban megérteni a kódot ezért kicsit beleástam magam az elméleti részekbe, hogy pontosan hogyan működik a konvolúciós neurális háló milyen paraméterei vannak és pontosan mit csinál a max-pooling. Ehhez nagy segítség volt a konzulens által ajánlott videósorozat ami az alábbi linken található:
https://www.youtube.com/watch?v=gZmobeGL0Yg&list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU&ab_channel=deeplizard

Max poolint tipikusan a konvolúciós neurális hálóknál alkalmazzuk. Max pooling segítségével csökkenteni tudjuk az input dimenzióját így hatékonyabbá téve a tanulási folyamatokat, kisebb méretű adaton kell dolgozni ez gyorsítja a tanulási folyamatokat és a max pooling segít, hogy ne forduljon elő az overfitting.



Itt láthatunk egy példát a működésére. Ahogy a képen is látszik, hogy a filter mérete 2x2-es ez azt jelenti, hogy ezzel a mérettel végigmegy az inputon és kiválasztja mindenhol a maximum értéket így pl 4 érték helyett 1-et kapunk, a legnagyobbat. Tehát ez a filter az input magasságát és szélességét is felére csökkenti, ha ezt a 2x2-es filtert kettővel toljuk arrébb, amit a Stride segítségével tudunk megadni.



Itt láthatunk rá egy példát:

```
model = Sequential([
        Dense(16, activation='relu', input_shape=(20,20,3)),
        Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same'),
        MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'),
        Conv2D(64, kernel_size=(5, 5), activation='relu', padding='same'),
        Flatten(),
        Dense(2, activation='softmax'),
    ])
```

pool_size paraméter megmondja mekkora legyen a filterünk a strides az eltolás mértéke a padding meg a kitöltést, aminél meg lehetne adni a „same” értéket is ilyenkor olyan állapotra hozza az inputot, hogy a különböző műveletek után az output mérete pont annyi legyen, mint az inputé ezért itt „valid” értéket használunk különben az egyész értelmét vesztené.

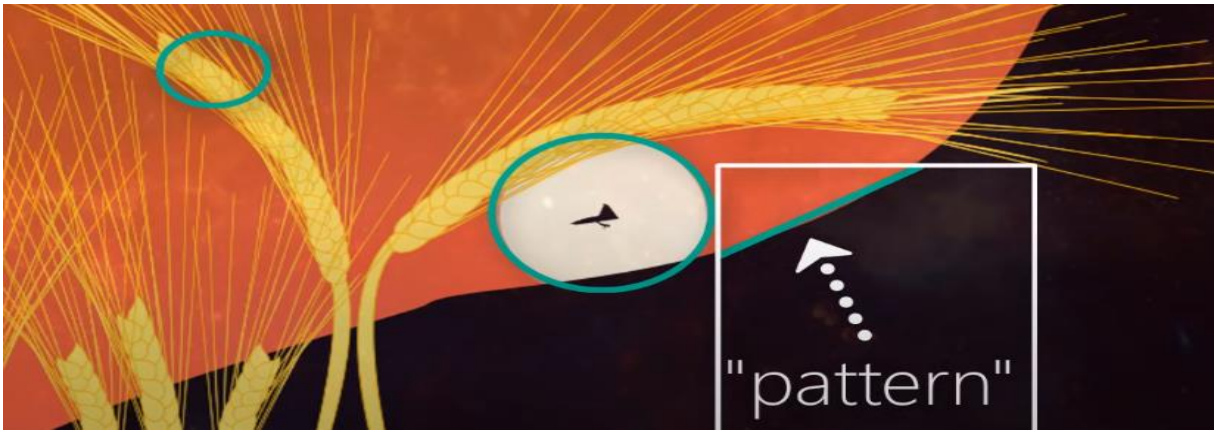
Összefoglalás:

```
model.summary()
```

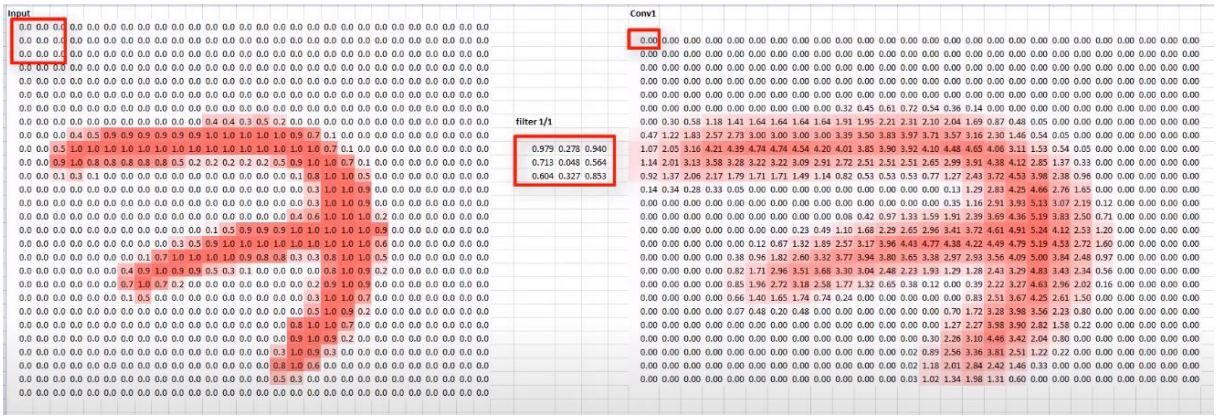
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 20, 20, 16)        64
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 20, 20, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 64)        51264
_________________________________________________________________
flatten_1 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 12802
=================================================================
Total params: 68,770
Trainable params: 68,770
```

Convolutional Neural Networks

A CNN az egyik legismertebb neurális háló a képfelismerés témakörében. A CNN egy olyan neurális háló, aminek az erősége az alakzatok megtalálása, tehát olyan speciális tulajdonságokkal rendelkezik, hogy hatékonyan ismeri fel a különböző alakzatokat filterek segítségével. A CNN tartalmaz úgynevezett convolutional layer-eket innen jön az elnevezése. Ezek a rétegek ugyanúgy, mint a többi az inputot transzformációk segítségével átalakítja és továbbítja a következő rétegnek csak itt ezek a transzformációs lépéseket konvolúciós műveletek. Különböző filterek segítségével képes felismerni a képen lévő alakzatokat, mint például ma kör vagy valaminek a sarka vagy az alakzatok szélei.



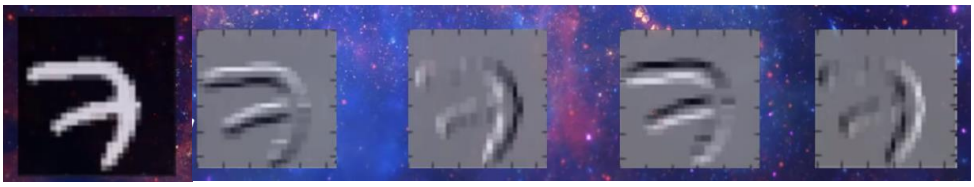
Filterek:



Mint ahogy a max poolingnál is láttuk itt is filterek vannak és hasonlóan működnek annyi különbséggel hogy itt nem a maximumot választjuk hanem minden értéket megszorzunk a megfelelő filter értékkel és összegezzük így születik meg egy pixel értéke Tehát itt nem az input nagyságának csökkentése a cél de sajnos ez is bekövetkezik mivel a széleket nem tudja számolni a filterrel ezért az első és utolsó sort és oszlopot törli de erre nyújt majd megoldást a padding amiről már volt szó a max poolingnál.

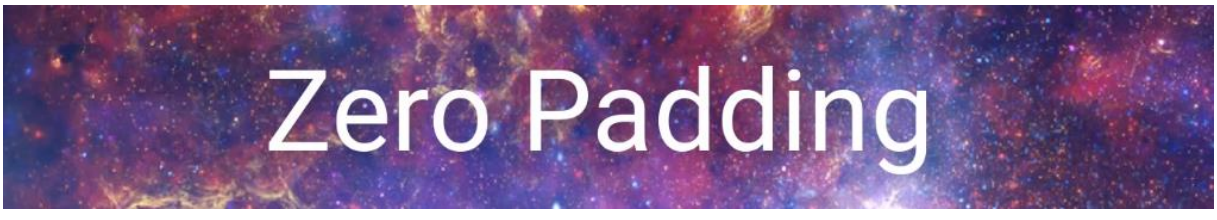
Különleges filterek:

Alkalmazásuk:

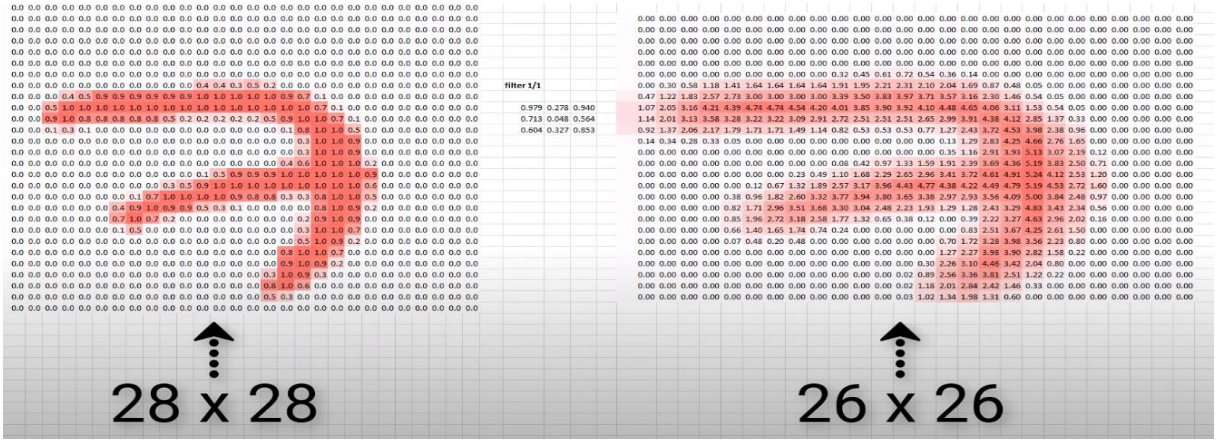


Látszik hogy alakzatunk élei különböző oldalról látszanak így konnyen felismerhetővé válik.

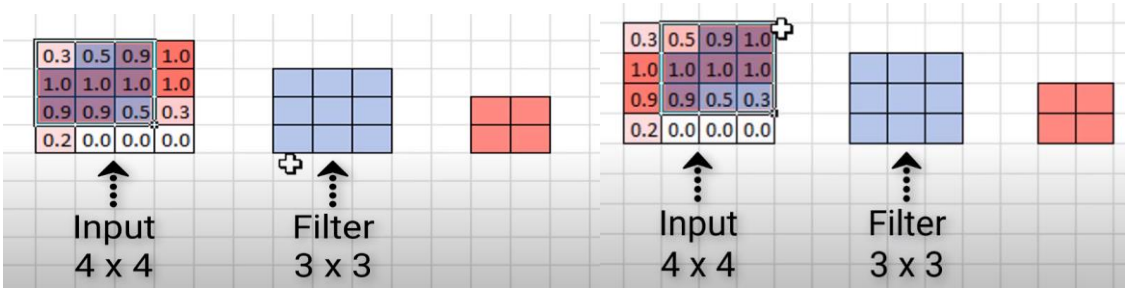
Látszik hogy ezeknél sokkal bonyolúltabb filterek is léteznek erre láthatunk példát a képen.



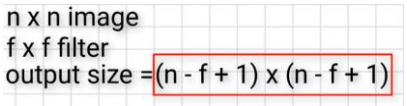
Mint ahogy már emlitettem a CNN transzformációk során csökkenti az input méretét és erre ad nekünk megoldást a padding. Több fajtája is létezik mint például a „same” ami azt eredményezi hogy az input és az output ugyanolyan méretű ming a „valid”-nál nem használunk paddinget igy az output mérete eltérhet az inputétól.



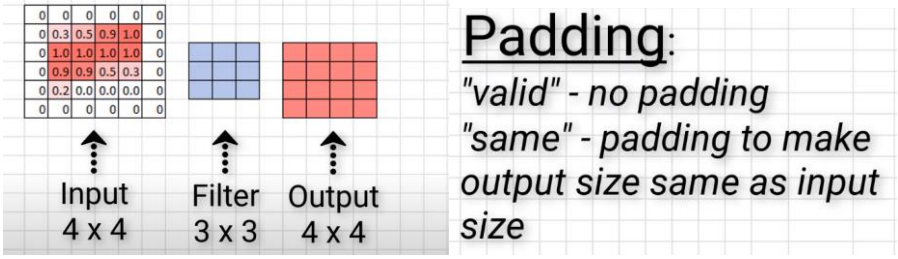
CNN a konvolúciós műveletek hatására átalakítja az output méretét okai:



Látszik hogy a filter soronként és oszloponként is csak kétszer fér el az inputban ezért a kimenet 2x2-es lesz.



Ezzel a képlettel tudjuk leírni a csökkenés méretét.



Ezen a képen jól látszik hogy működik a zero padding. Az inputot kiegészítjuk csupa 0 értékkel felveszünk hozza egy új sort és oszlopot ennek hatására az output ugyanakkora méretűvé válik mint az input volt mivel a filter így csak azt az oszlopot és sort vágja le amit hozzá vettünk.

Példa:

```
model_valid = Sequential([
        Dense(16, activation='relu', input_shape=(20,20,3)),
        Conv2D(32, kernel_size=(3, 3), activation='relu', padding='valid'),
        Conv2D(64, kernel_size=(5, 5), activation='relu', padding='valid'),
        Conv2D(128, kernel_size=(7, 7), activation='relu', padding='valid'),
        Flatten(),
        Dense(2, activation='softmax'),
    ])
```

```
Output Shape
=========================
(None, 20, 20, 16)
(None, 18, 18, 32)
(None, 14, 14, 64)
(None, 8, 8, 128)
(None, 8192)
(None, 2)
```

```
model_same = Sequential([
        Dense(16, activation='relu', input_shape=(20,20,3)),
        Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same'),
        Conv2D(64, kernel_size=(5, 5), activation='relu', padding='same'),
        Conv2D(128, kernel_size=(7, 7), activation='relu', padding='same'),
        Flatten(),
        Dense(2, activation='softmax'),
    ])
```

```
Output Shape
=========================
(None, 20, 20, 16)
(None, 20, 20, 32)
(None, 20, 20, 64)
(None, 20, 20, 128)
(None, 51200)
(None, 2)
```