Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

Κωνσταντίνος Καραμανής

The University of Texas at Austin & Archimedes/Athena RC

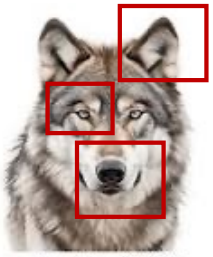
constantine@utexas.edu

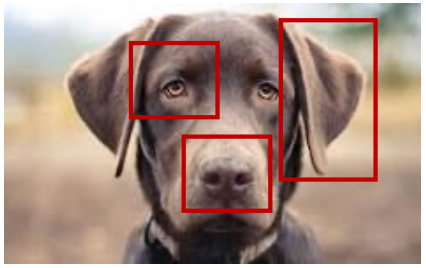
https://caramanis.github.io/

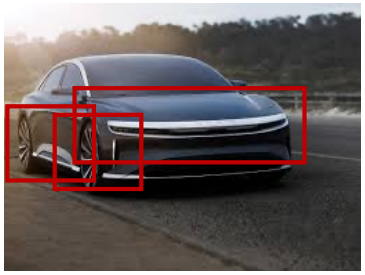
Νευρωνικά Δίκτυα

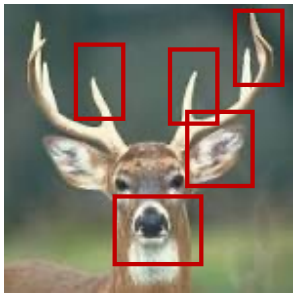
Μία απείρως πλούσια οικογένεια αλγορίθμων που εάν σχεδιαστούν σωστά, ταιριάζουν με πολλούς τομείς

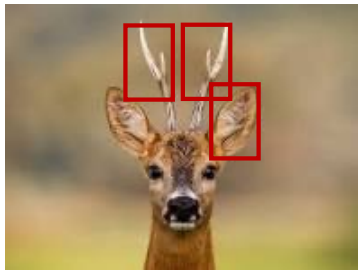
Πως διαφέρουν αυτές οι εικόνες;





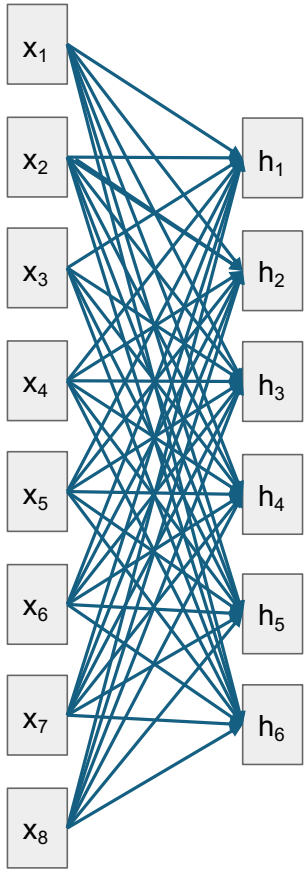






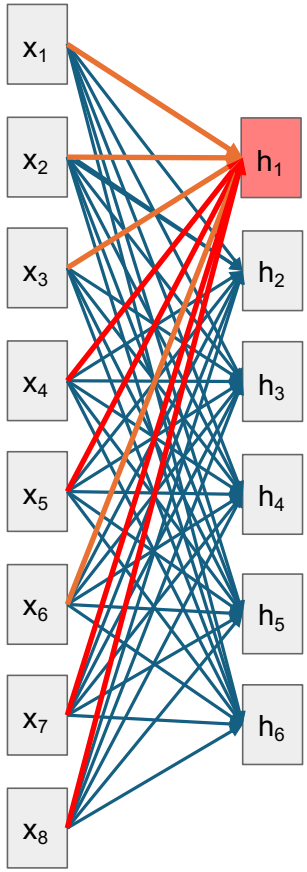
Πως διαφέρουν αυτές οι εικόνες;

Fully connected επίπεδο



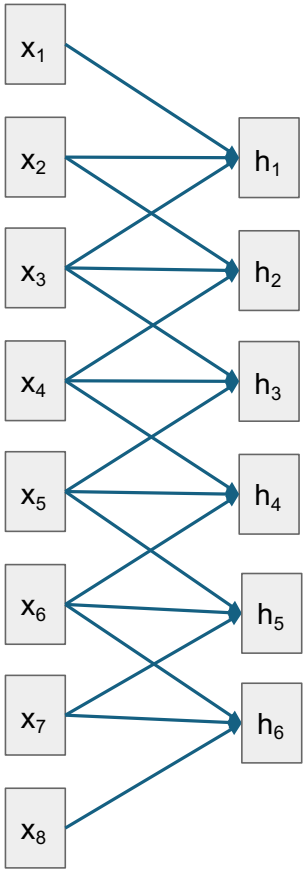
Κάθε νευρώνας «βλέπει» όλα τα δεδομένα – «όλη την εικόνα»

Fully connected επίπεδο



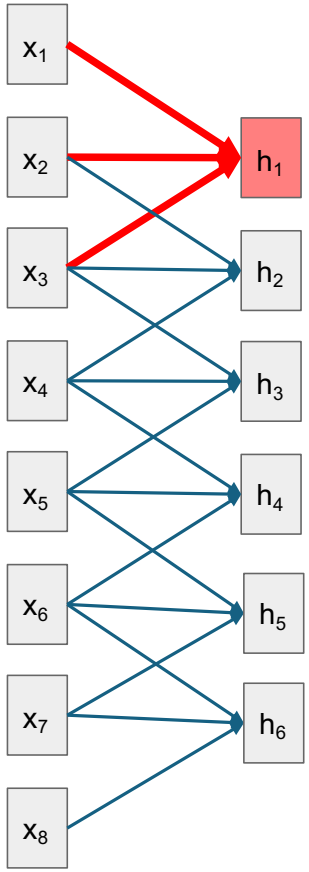
Κάθε νευρώνας «βλέπει» όλα τα δεδομένα – «όλη την εικόνα»

Convolutional επίπεδο



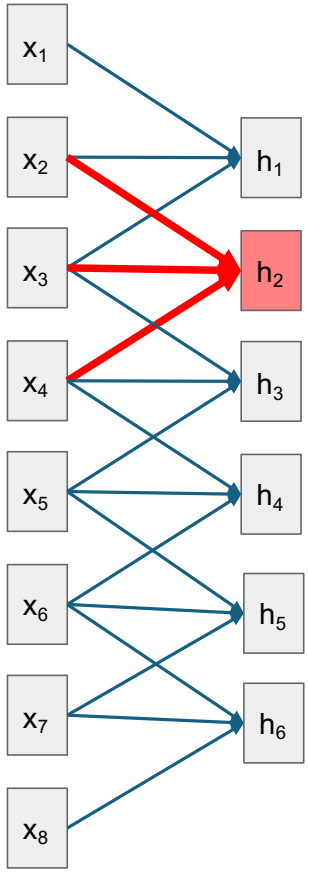
Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Convolutional επίπεδο



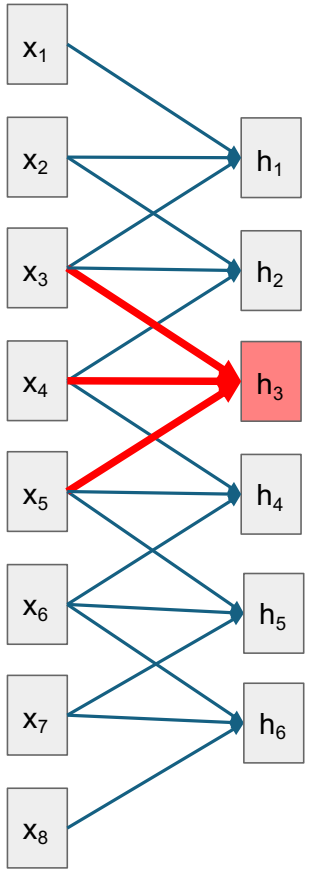
Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Convolutional επίπεδο



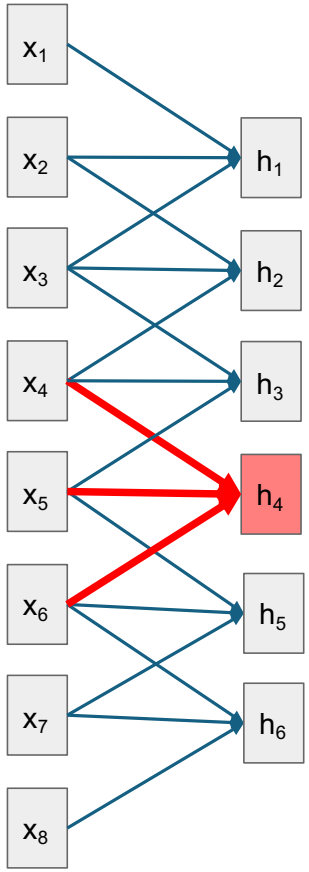
Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Convolutional επίπεδο



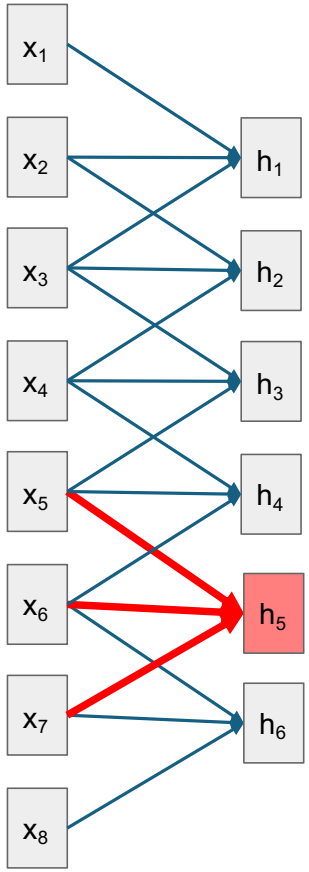
Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Convolutional επίπεδο



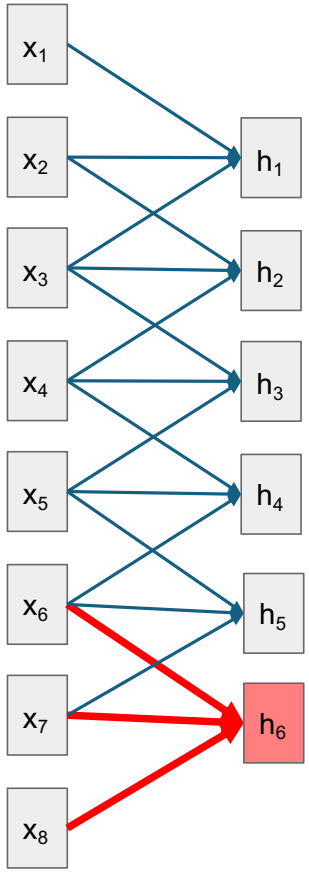
Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Convolutional επίπεδο



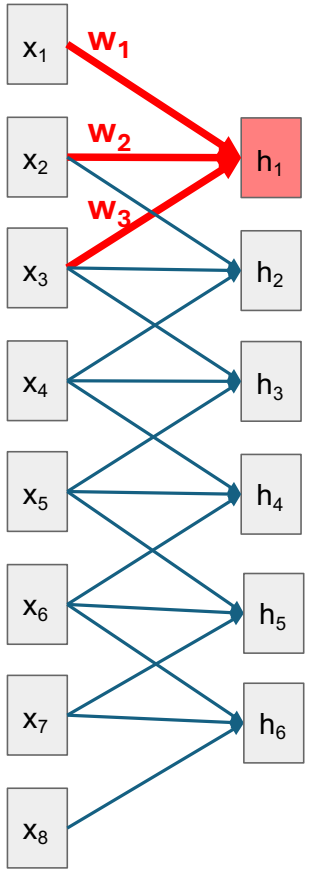
Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

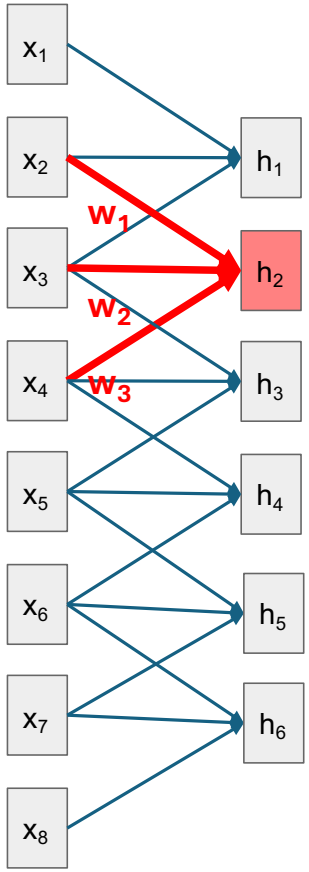
Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Κάθε νευρώνας κάνει τον ίδιο υπολογισμό!

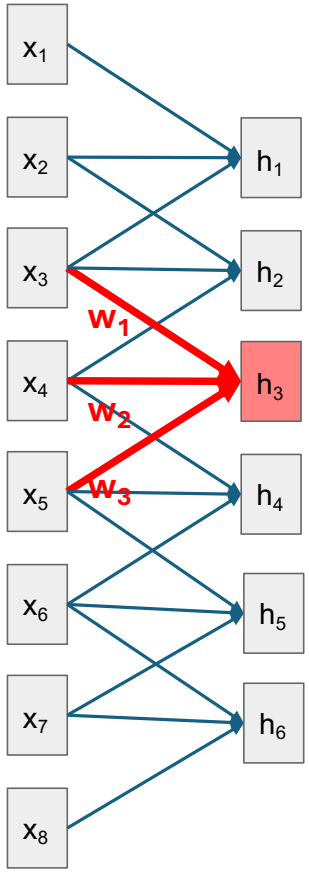
Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Κάθε νευρώνας κάνει τον ίδιο υπολογισμό!

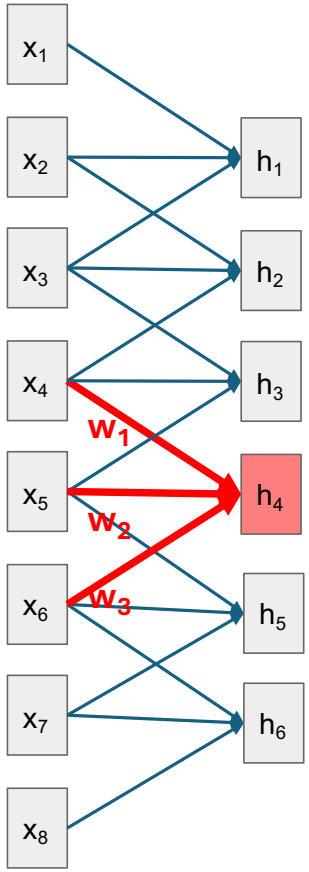
Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Κάθε νευρώνας κάνει τον ίδιο υπολογισμό!

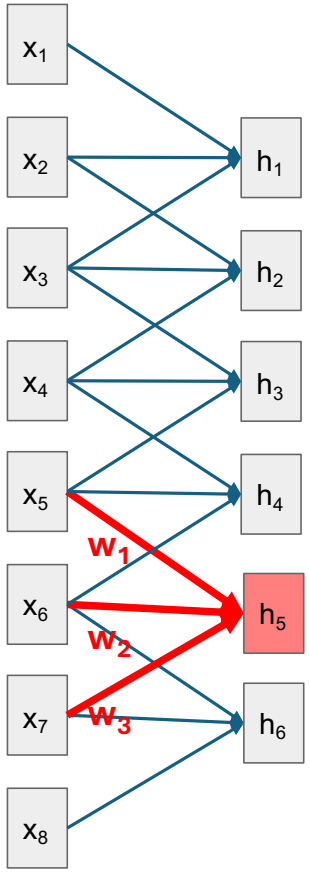
Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Κάθε νευρώνας κάνει τον ίδιο υπολογισμό!

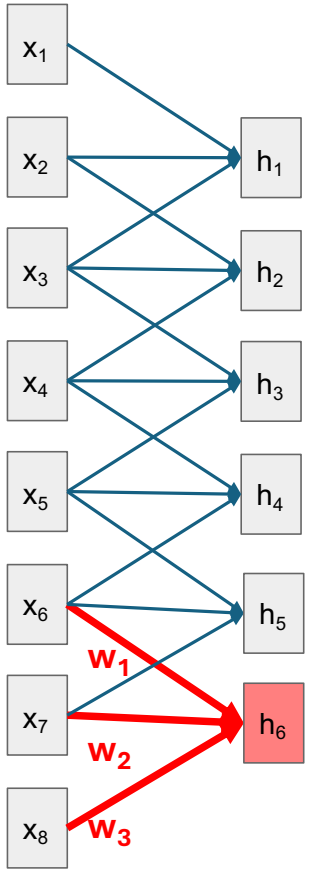
Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Κάθε νευρώνας κάνει τον ίδιο υπολογισμό!

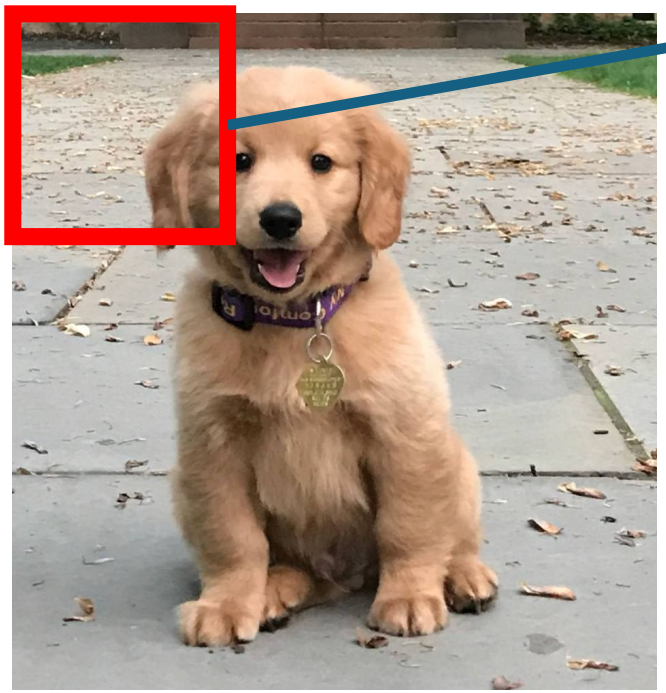
Convolutional επίπεδο



Ο νευρώνας h_1 βλέπει το (x_1,x_2,x_3), ο h_2 το (x_2,x_3,x_4), ο h_3 το (x_3,x_4,x_5), ο h_4 το (x_4,x_5,x_6), ο h_5 το (x_5,x_6,x_7), ο h_6 το (x_6,x_7,x_8).

Κάθε νευρώνας κάνει τον ίδιο υπολογισμό!

Convolutional επίπεδο για εικόνες



h_1

h_2

h_3

h_4

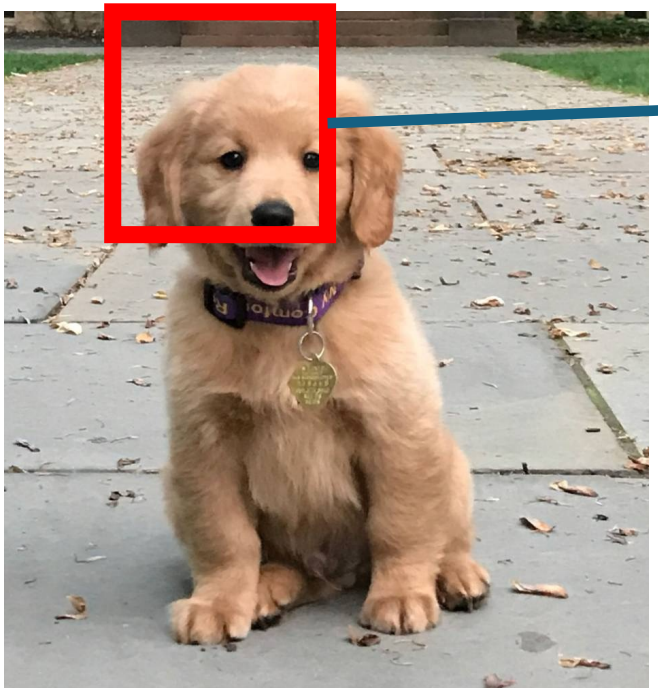
h_5

h_6

h_7

h_8

Convolutional επίπεδο για εικόνες



h_1

h_2

h_3

h_4

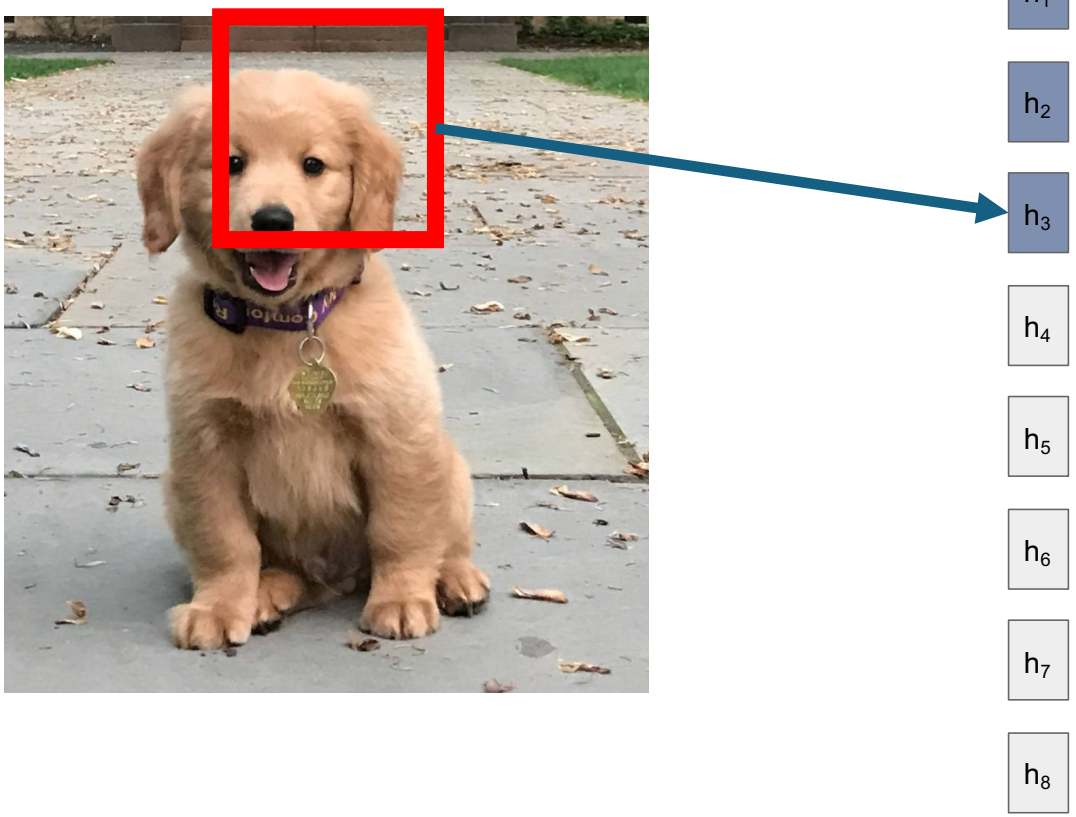
h_5

h_6

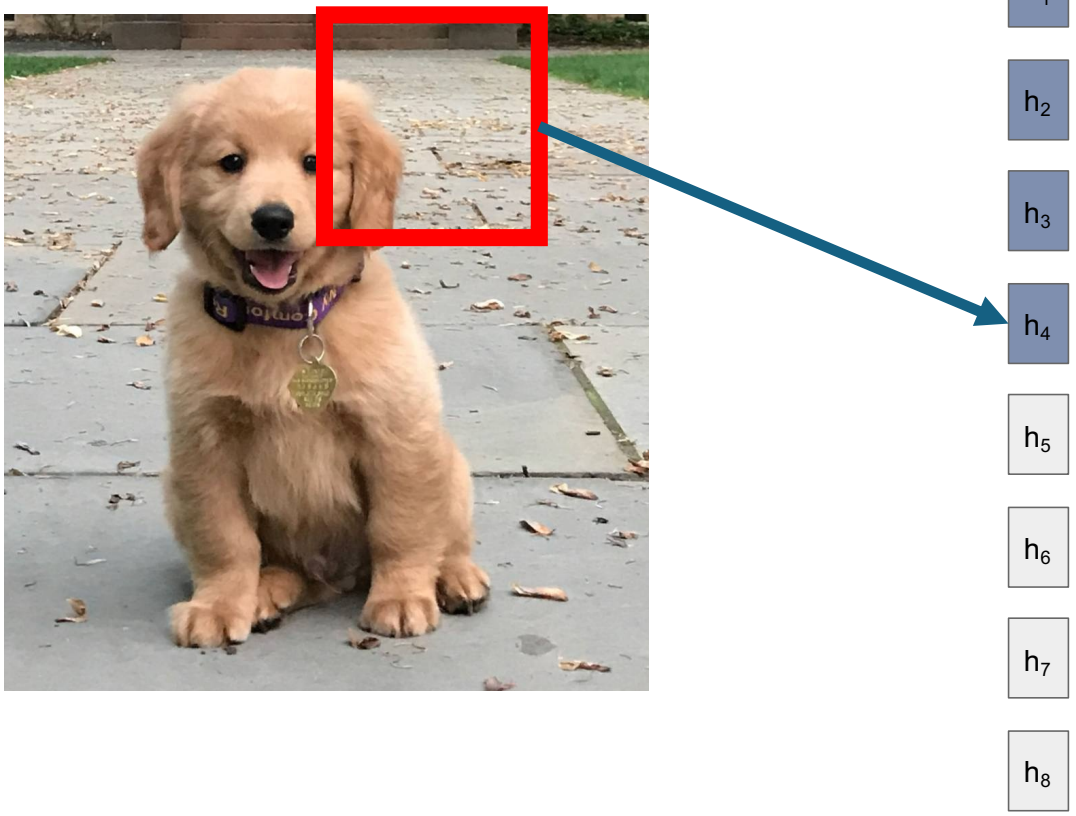
h_7

h_8

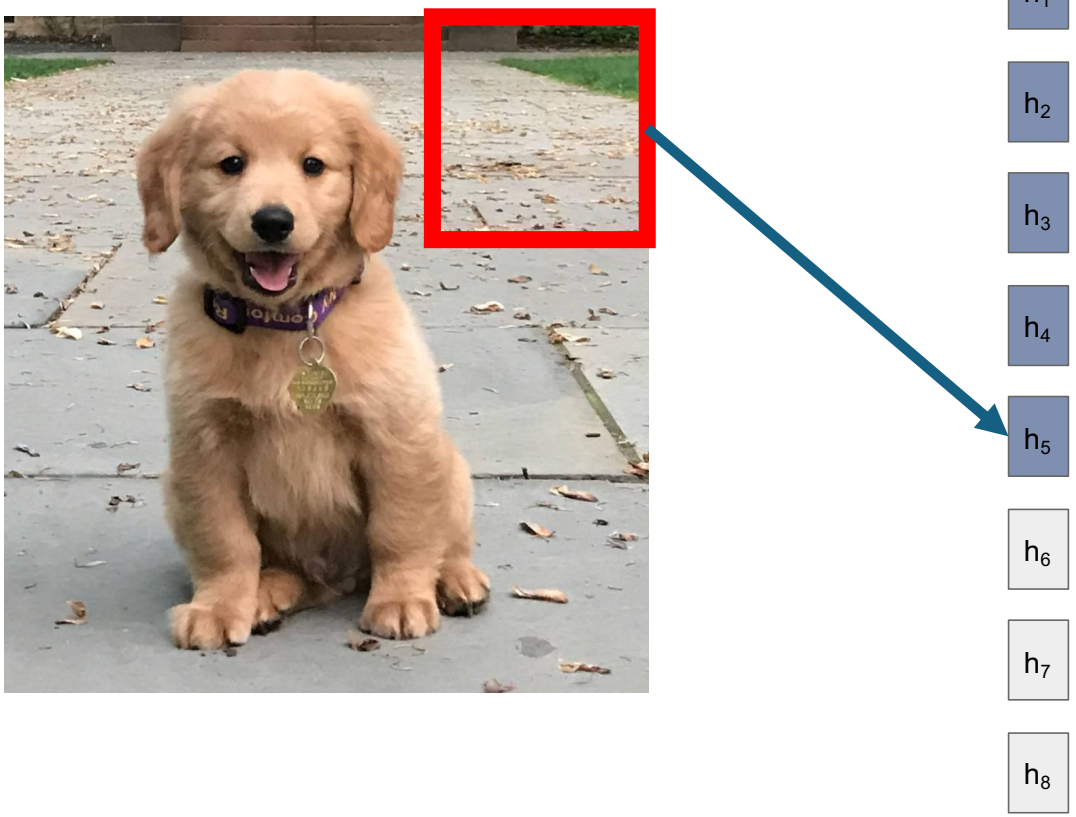
Convolutional επίπεδο για εικόνες



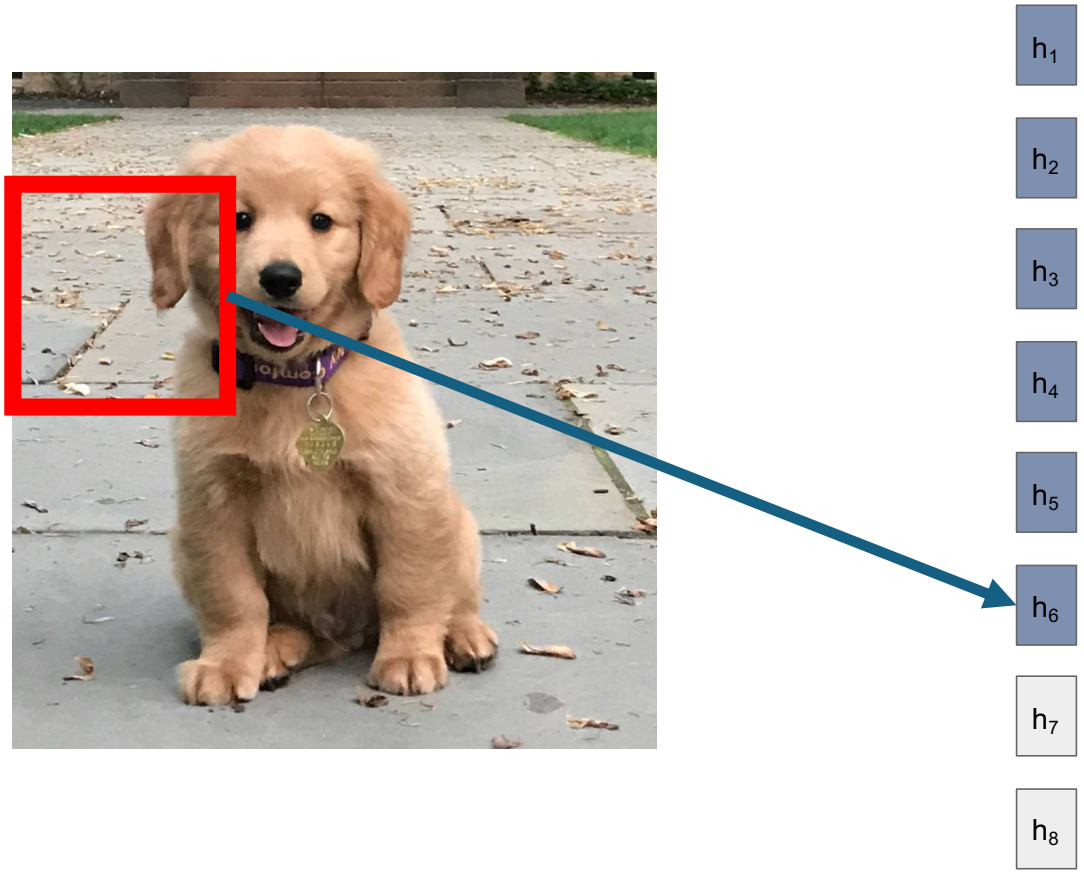
Convolutional επίπεδο για εικόνες



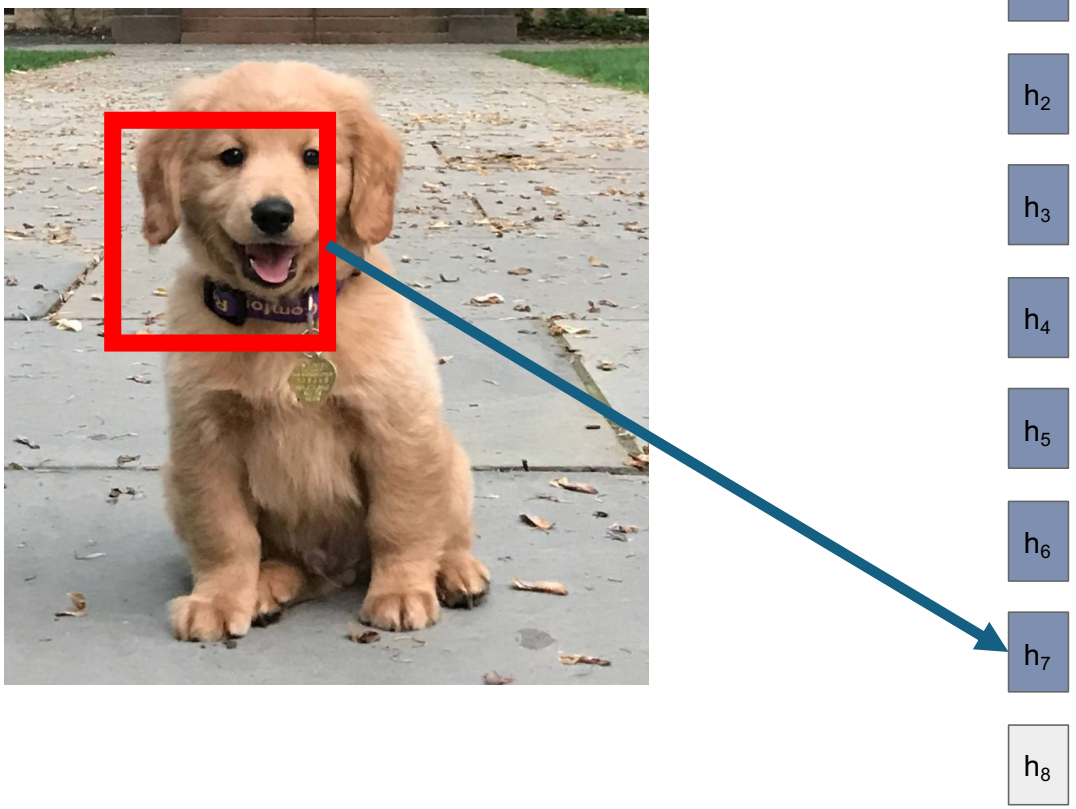
Convolutional επίπεδο για εικόνες



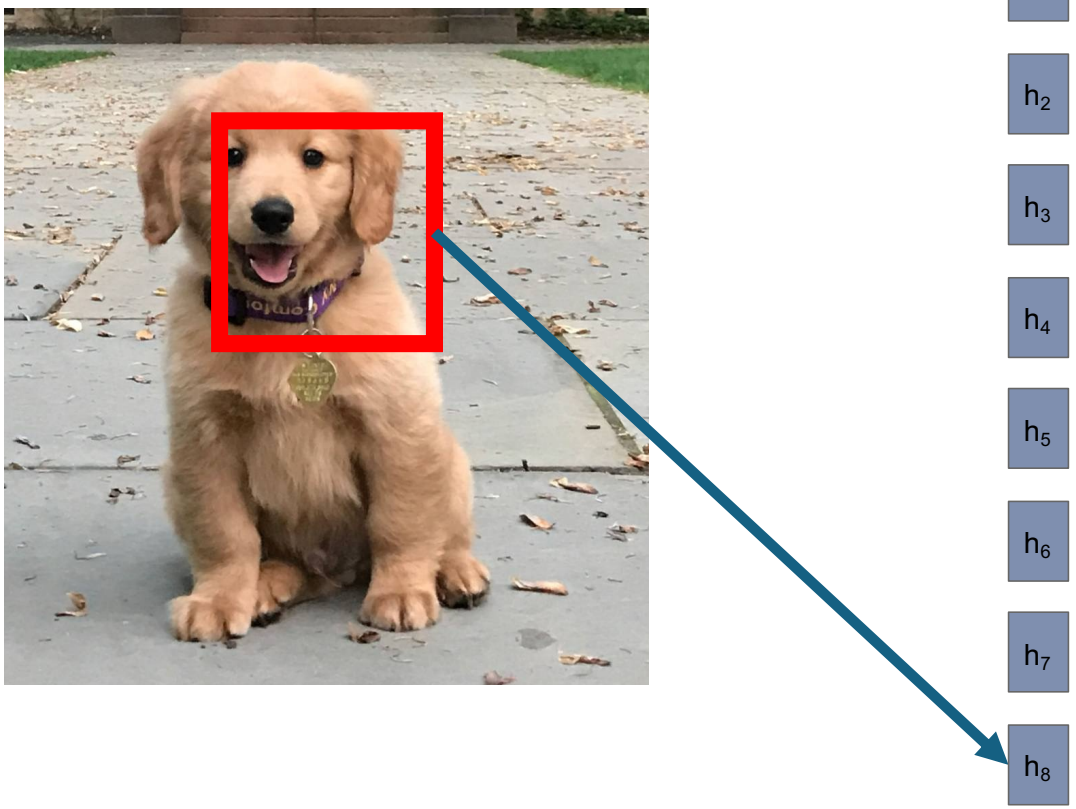
Convolutional επίπεδο για εικόνες



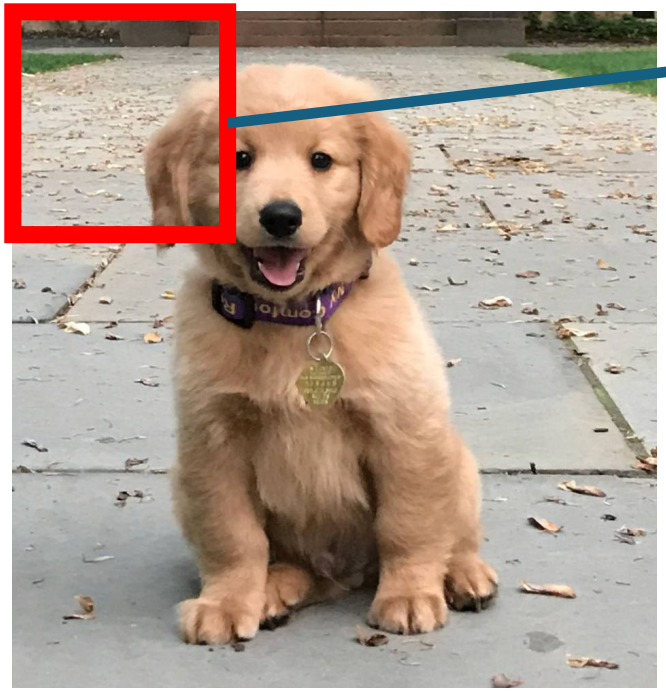
Convolutional επίπεδο για εικόνες



Convolutional επίπεδο για εικόνες



Convolutional επίπεδο για εικόνες

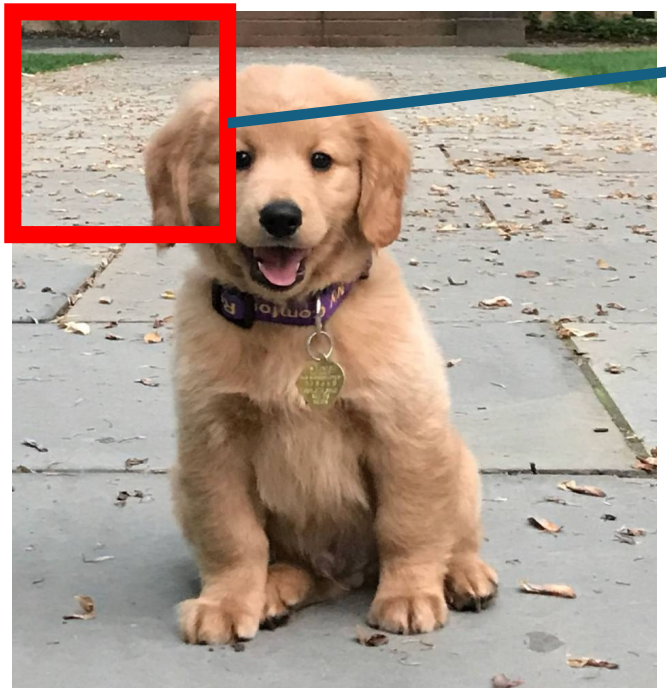


h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
h_{21}	h_{22}	h_{23}	h_{24}	h_{25}
h_{31}	h_{32}	h_{33}	h_{34}	h_{35}
h_{41}	h_{42}	h_{43}	h_{44}	h_{45}
h_{51}	h_{52}	h_{53}	h_{54}	h_{55}

Τί κάνει το convolutional επίπεδο;

Προσπαθεί να βρεί συγκεκριμένα σχήματα σε κάθε κομμάτι της εικόνας.

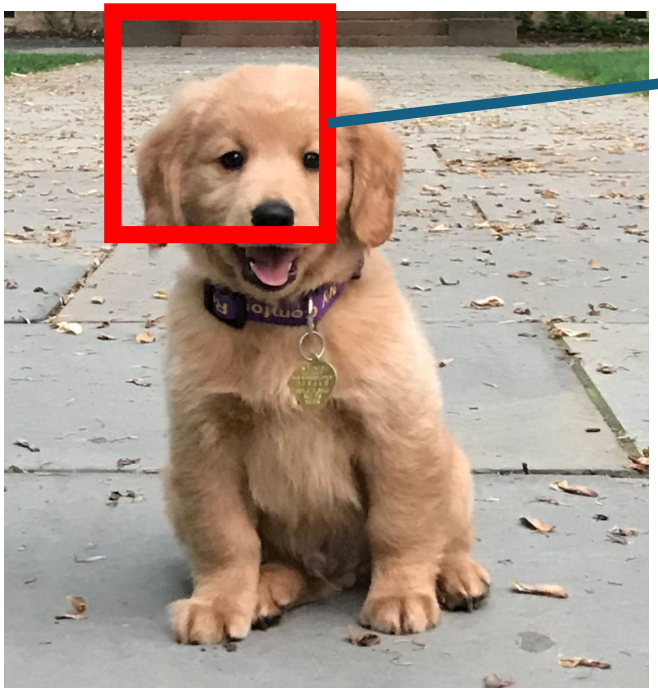
Convolutional επίπεδο για εικόνες

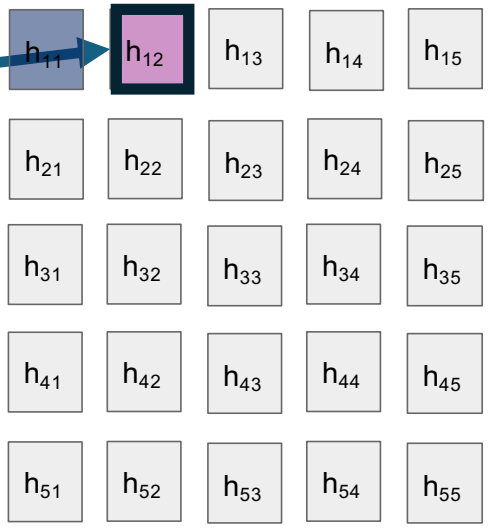


h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
h_{21}	h_{22}	h_{23}	h_{24}	h_{25}
h_{31}	h_{32}	h_{33}	h_{34}	h_{35}
h_{41}	h_{42}	h_{43}	h_{44}	h_{45}
h_{51}	h_{52}	h_{53}	h_{54}	h_{55}

«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

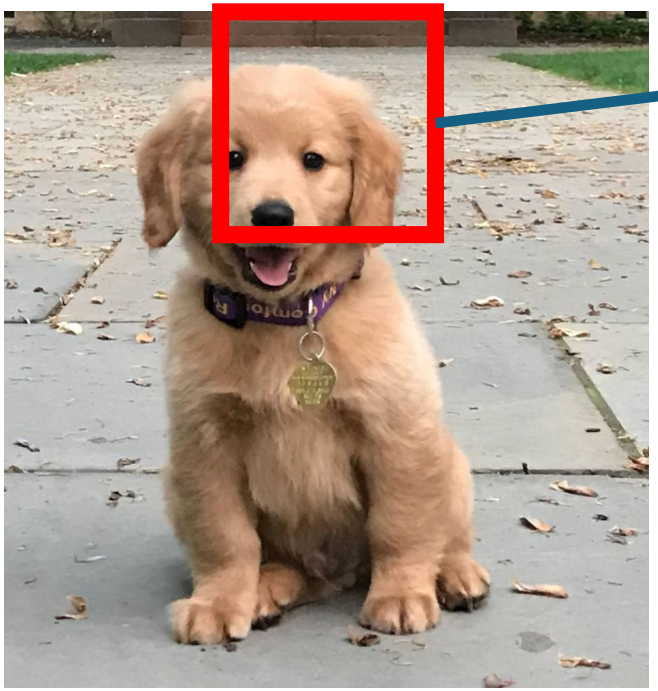
Convolutional επίπεδο για εικόνες

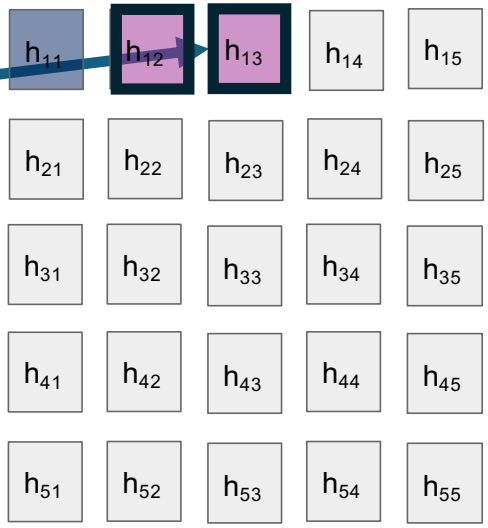




«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

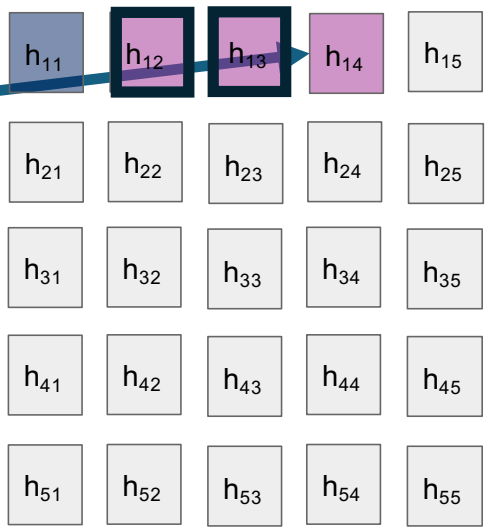
Convolutional επίπεδο για εικόνες





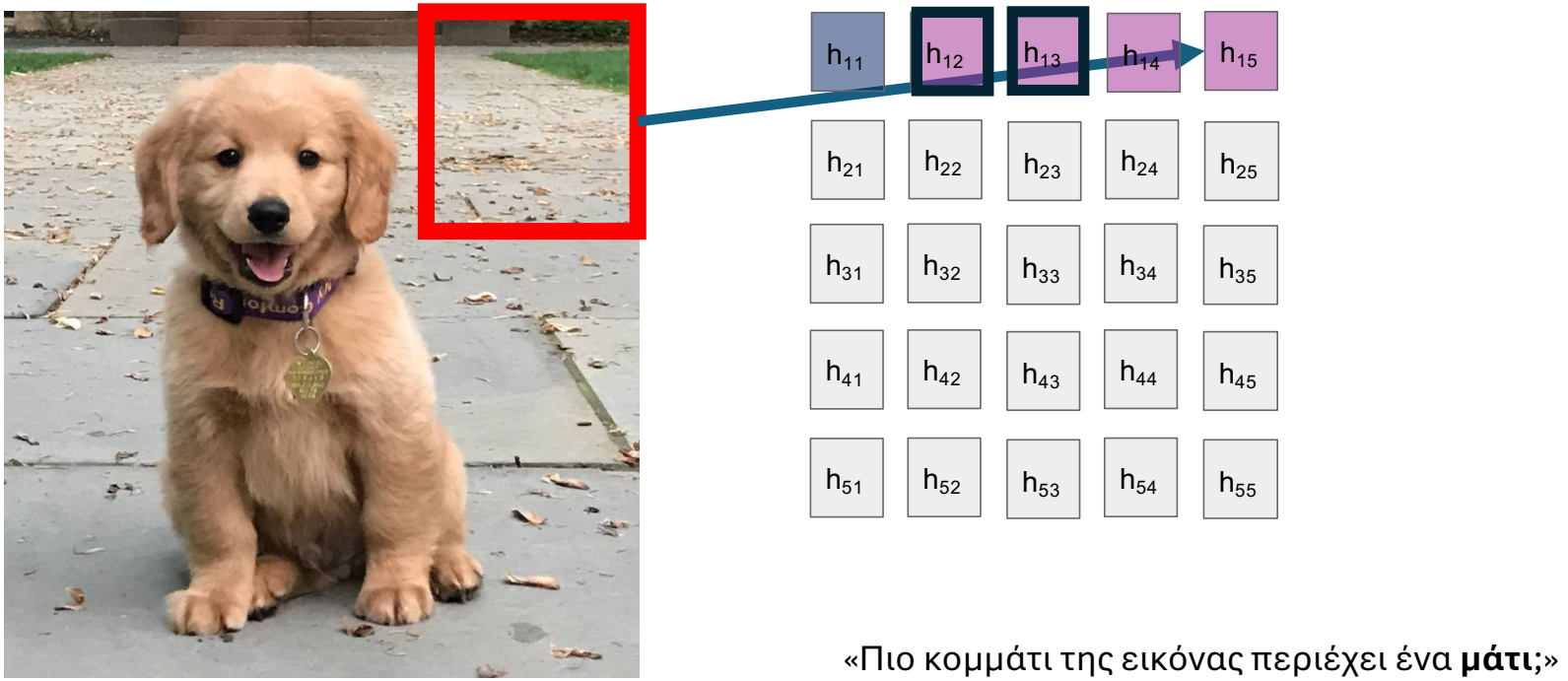
«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

Convolutional επίπεδο για εικόνες



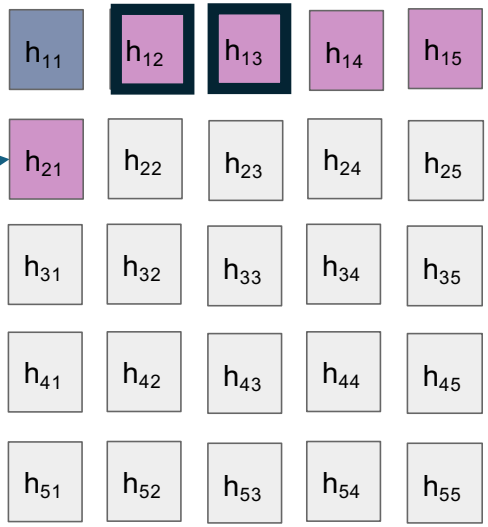
«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

Convolutional επίπεδο για εικόνες



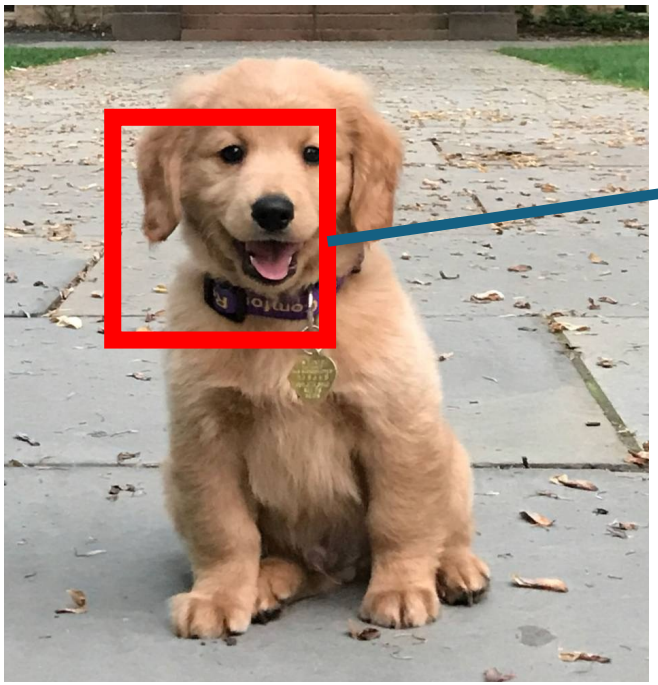
«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

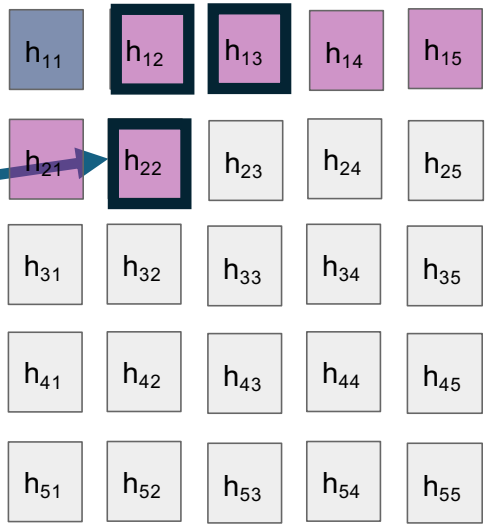
Convolutional επίπεδο για εικόνες



«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

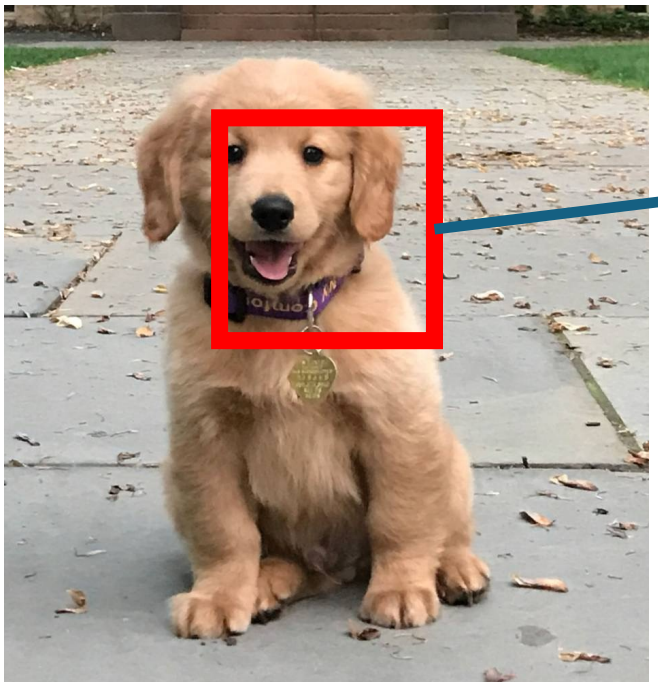
Convolutional επίπεδο για εικόνες

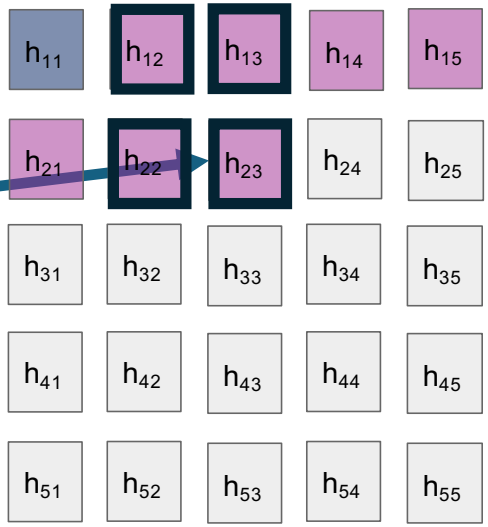




«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

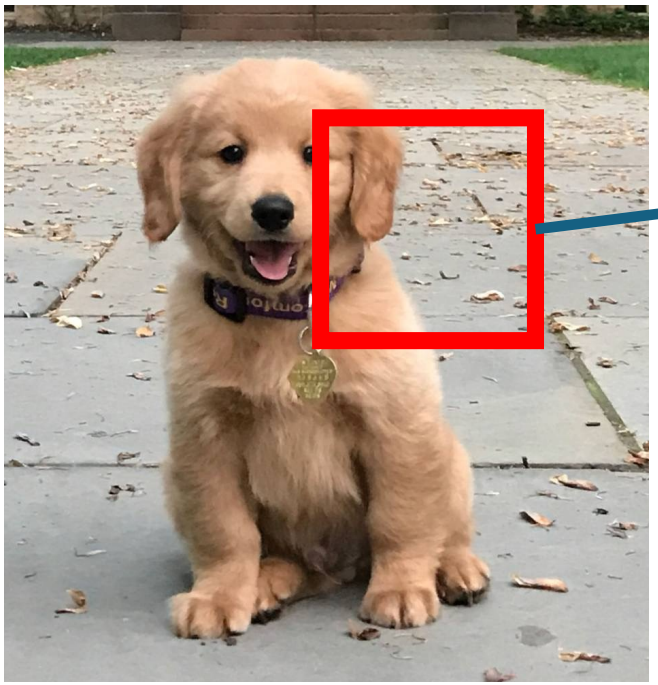
Convolutional επίπεδο για εικόνες

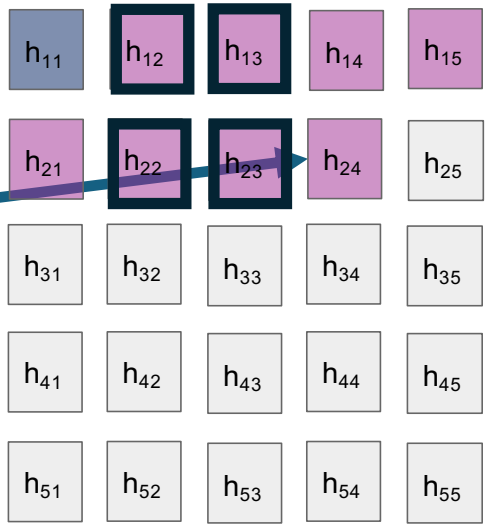




«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

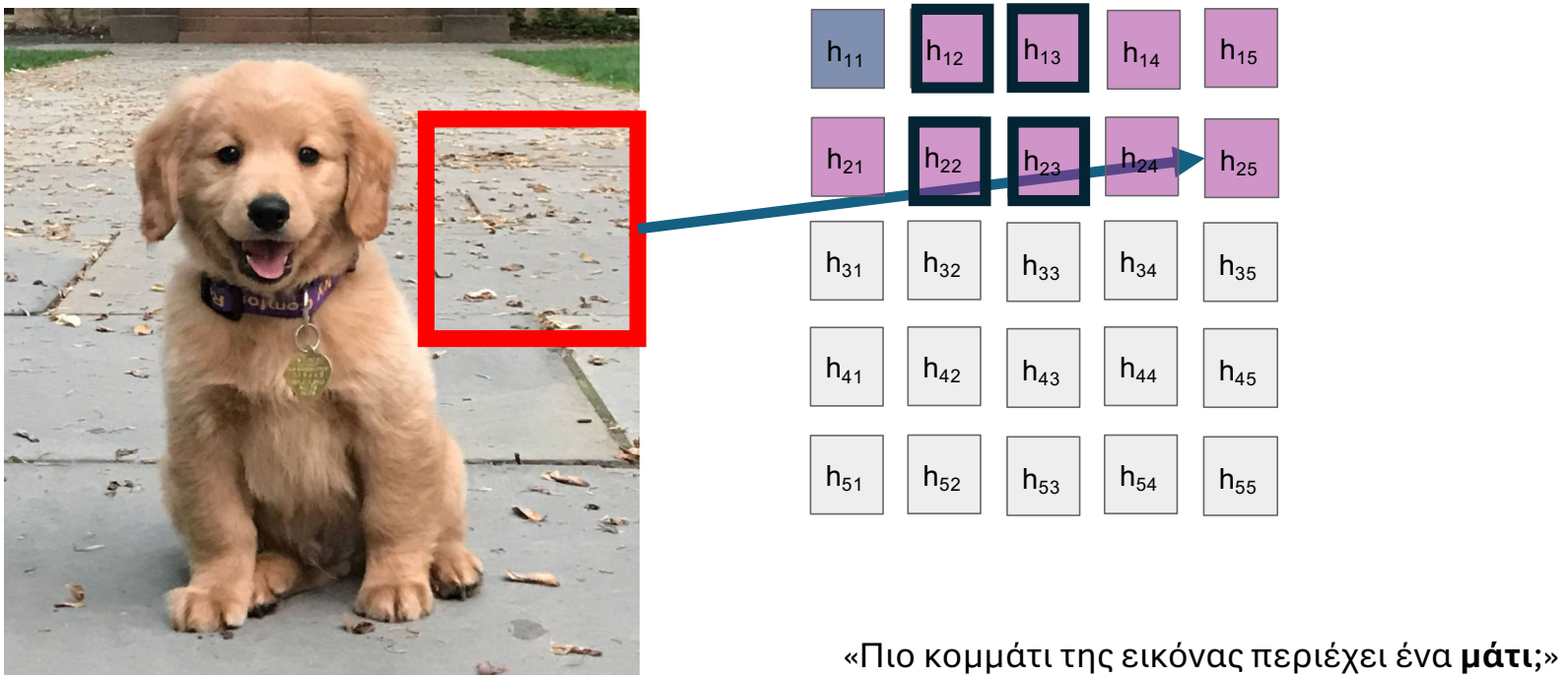
Convolutional επίπεδο για εικόνες





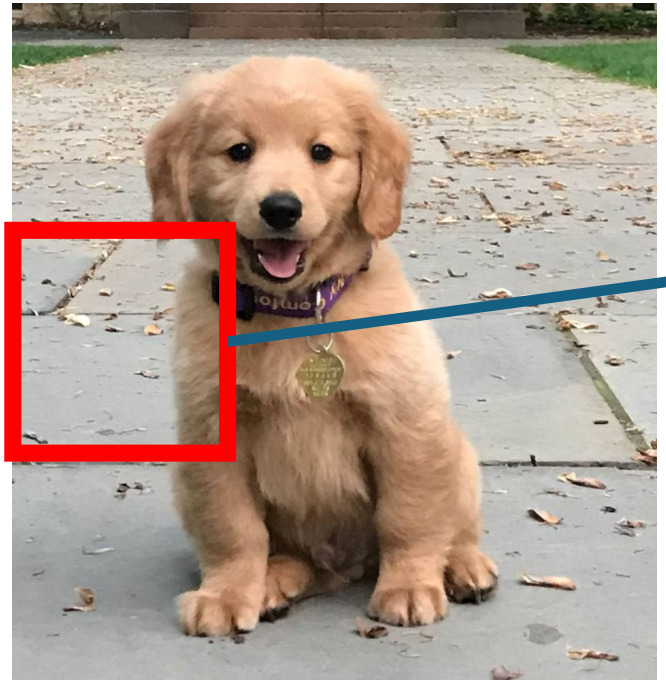
«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

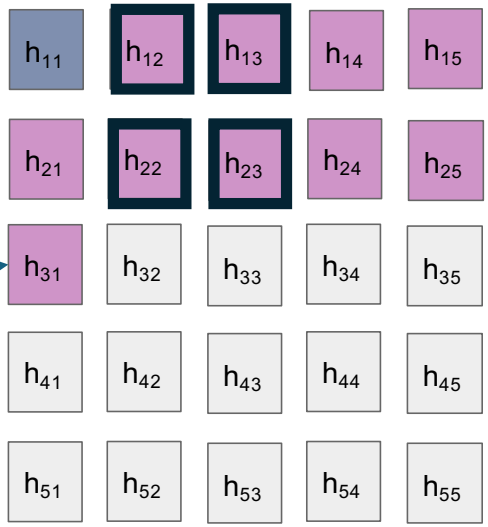
Convolutional επίπεδο για εικόνες



«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

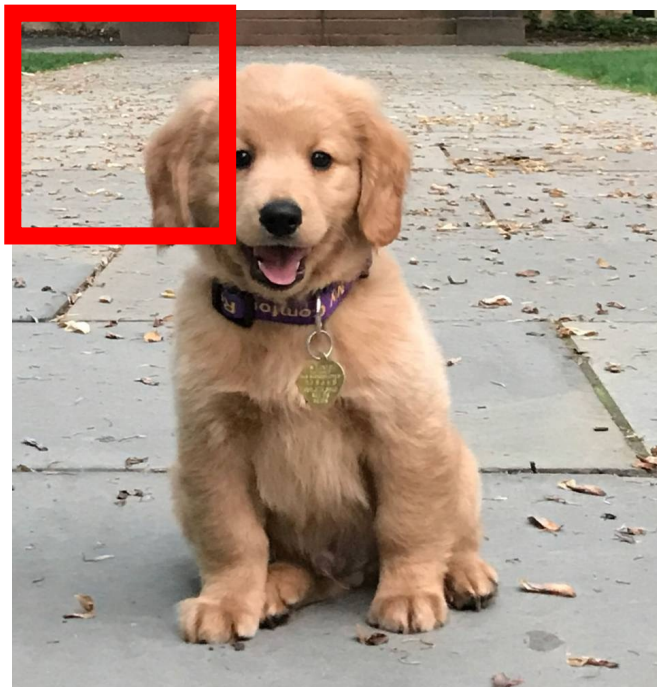
Convolutional επίπεδο για εικόνες

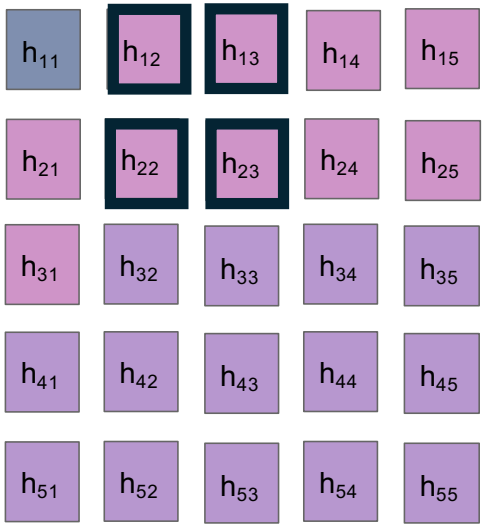




«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

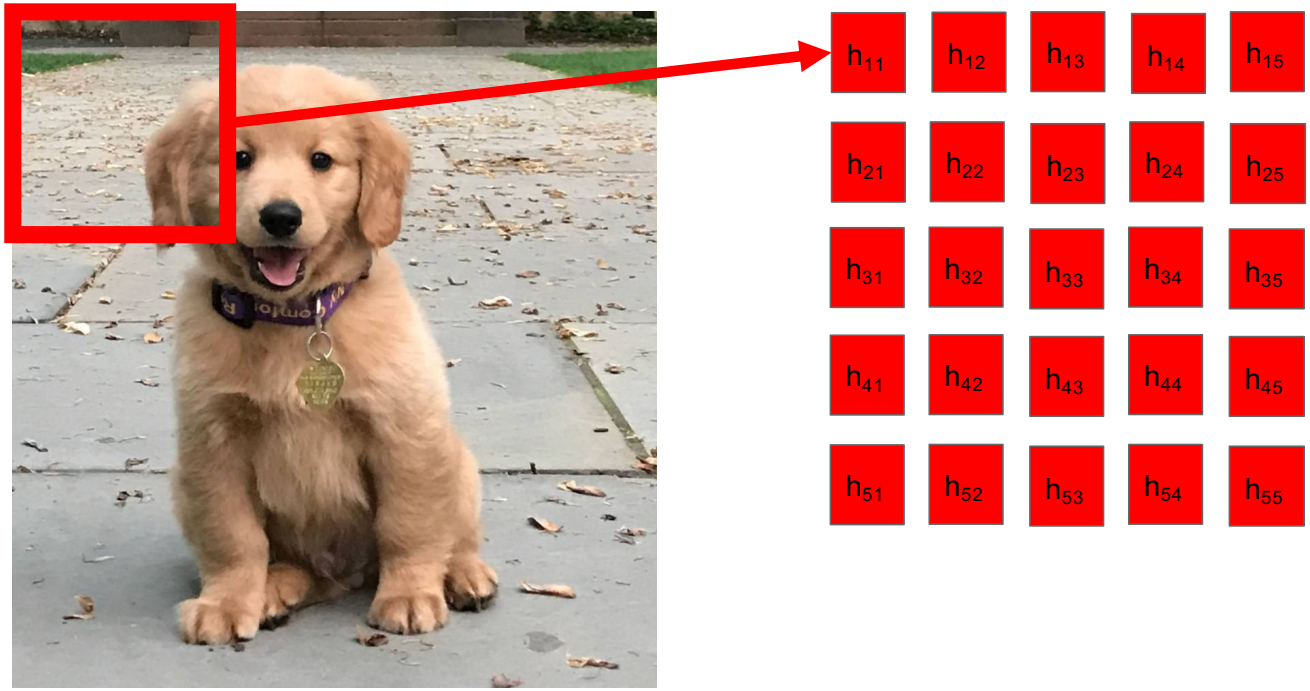
Convolutional επίπεδο για εικόνες



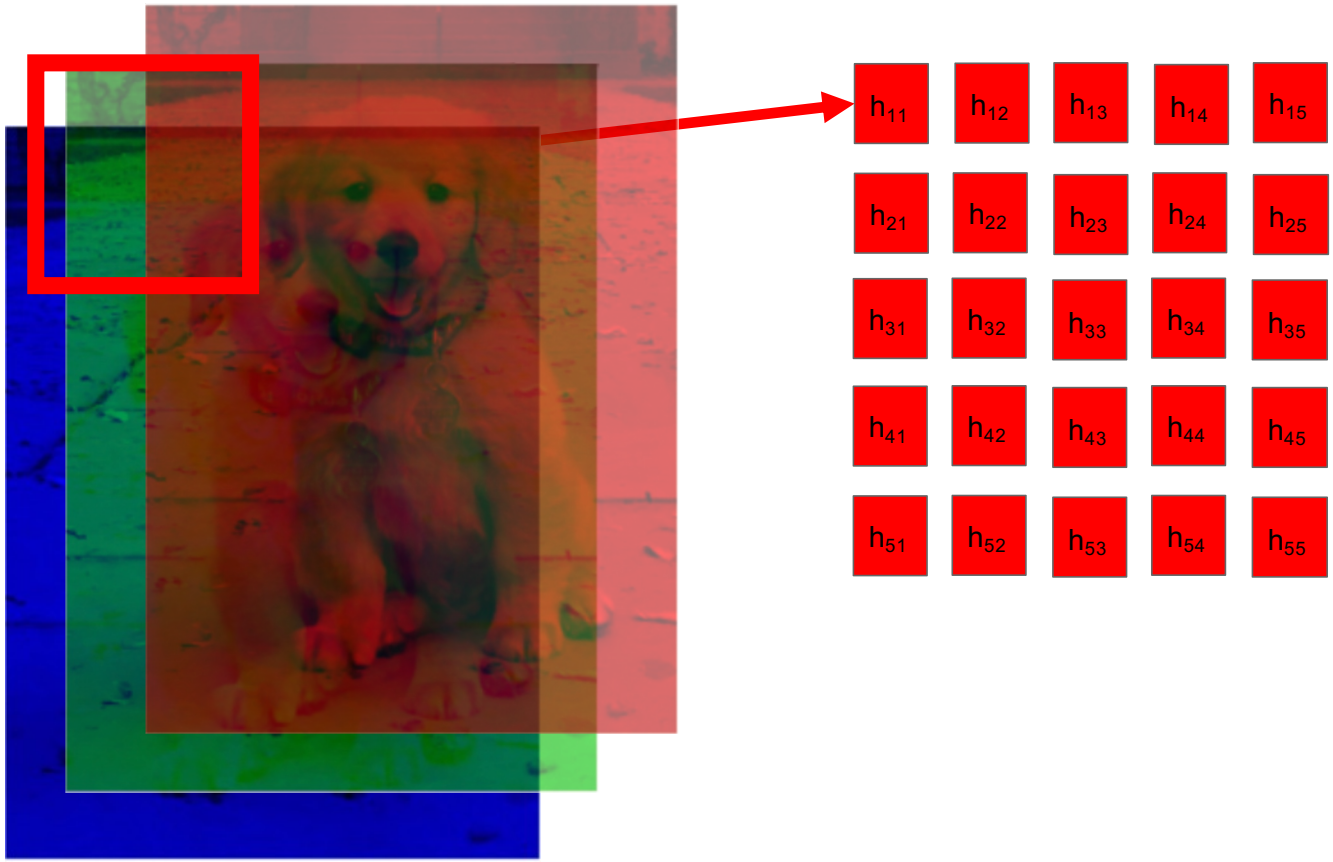


«Πιο κομμάτι της εικόνας περιέχει ένα **μάτι;**»

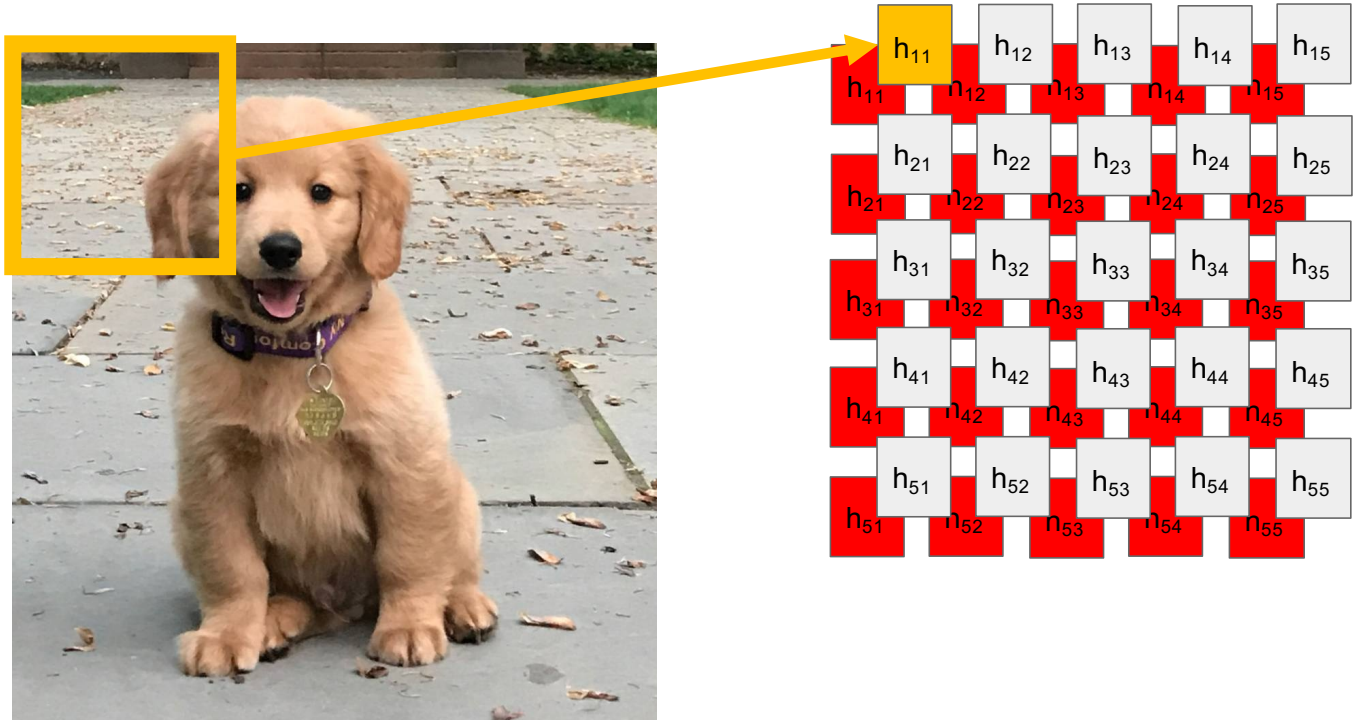
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά ***channels***



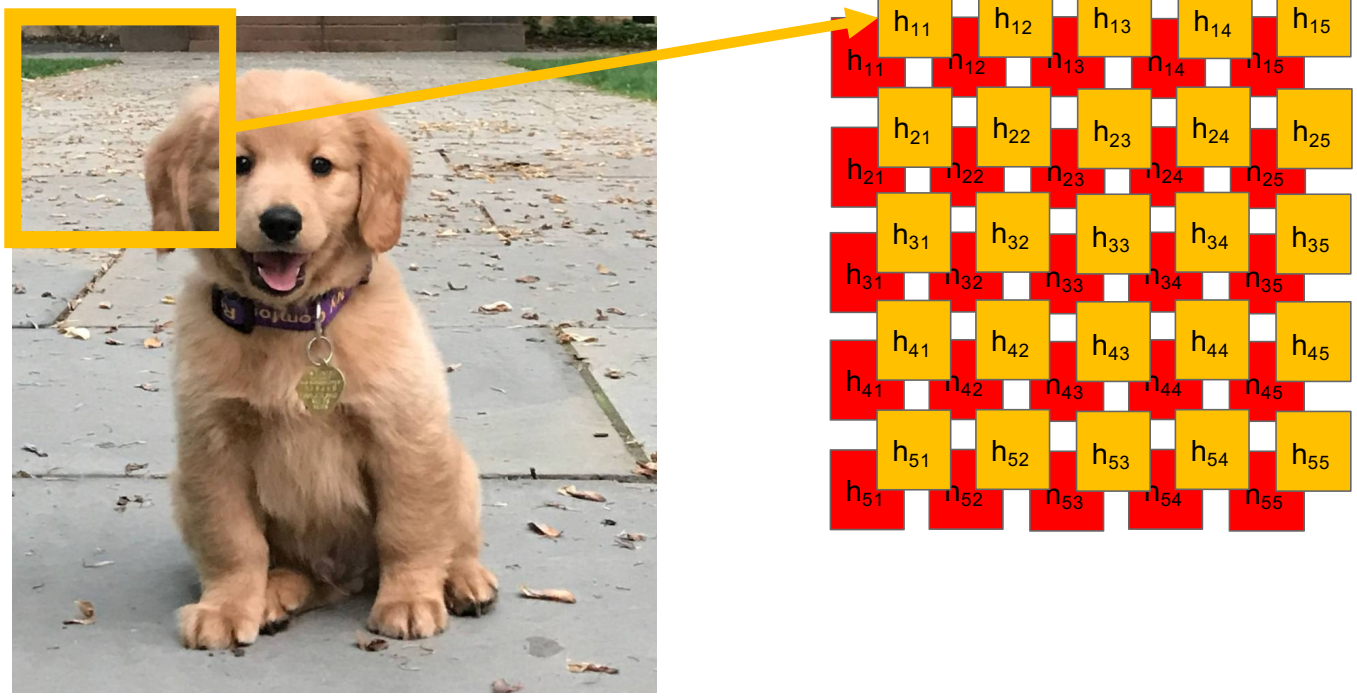
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά ***channels***



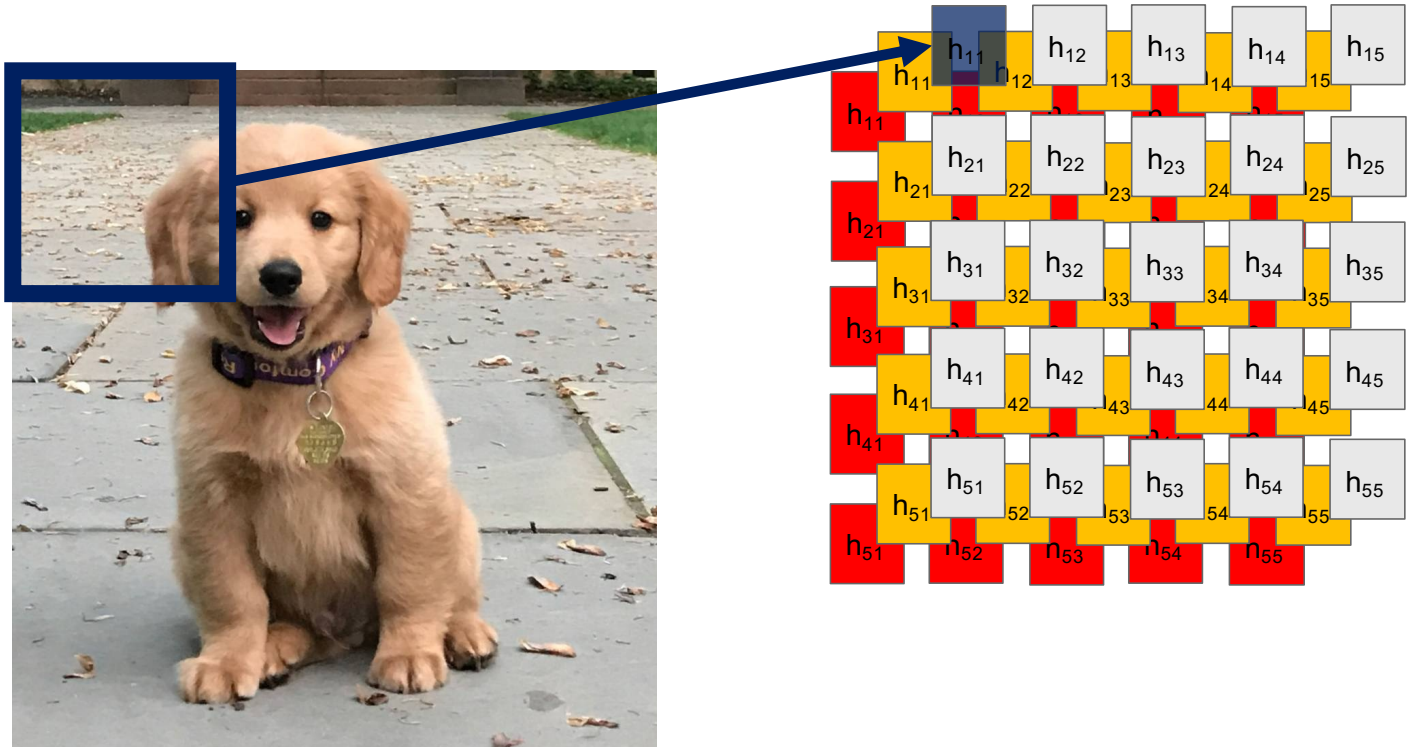
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels



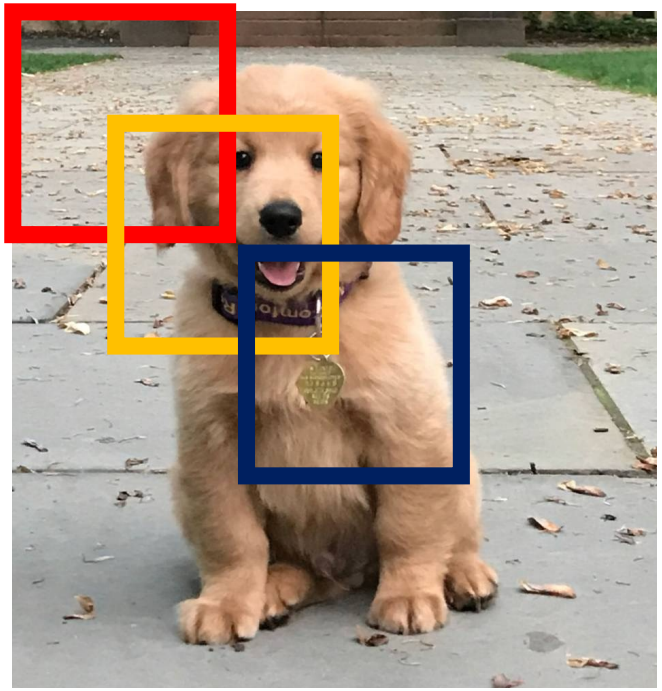
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels

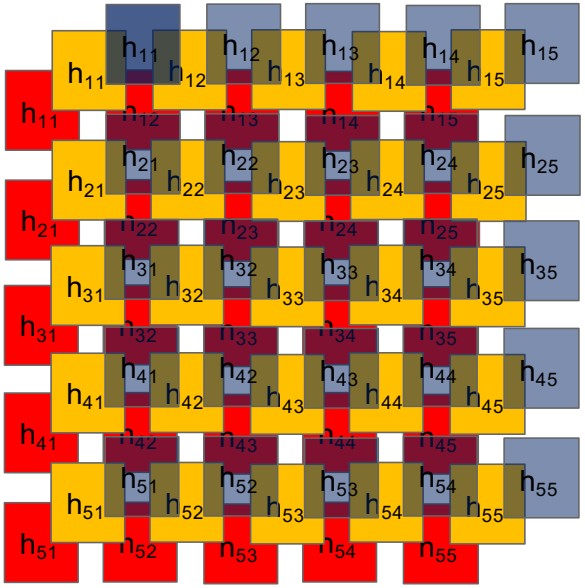


Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels

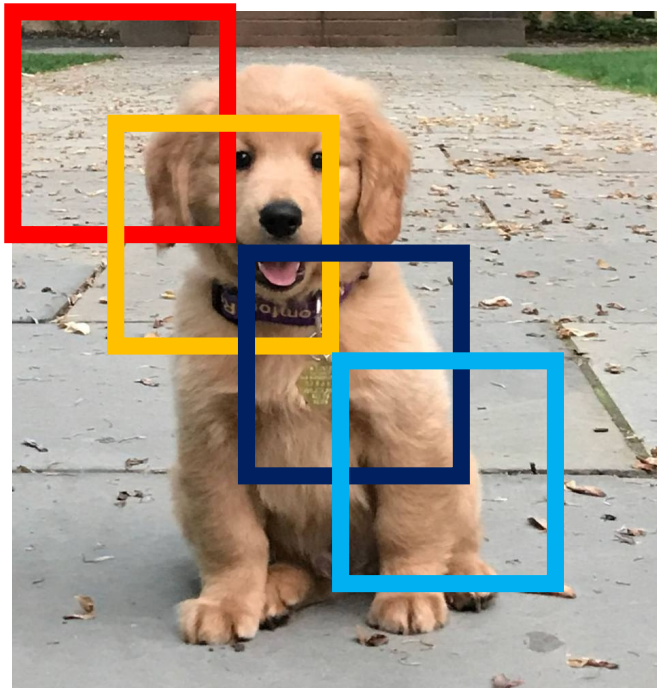


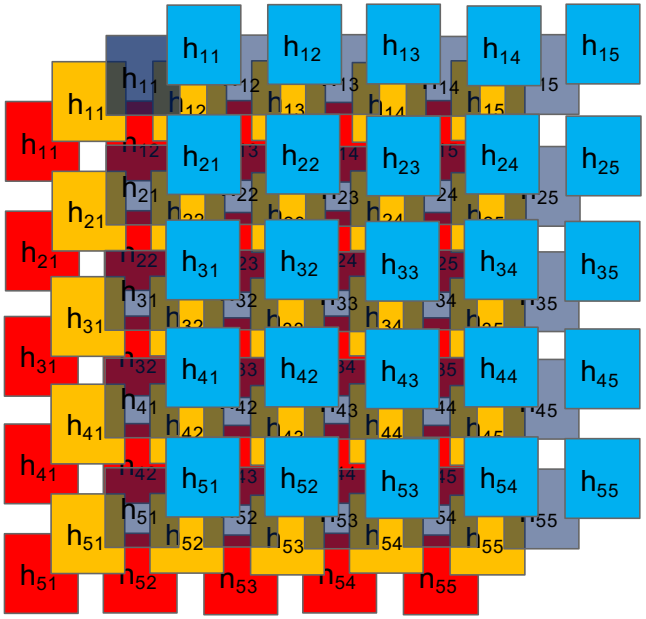
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels



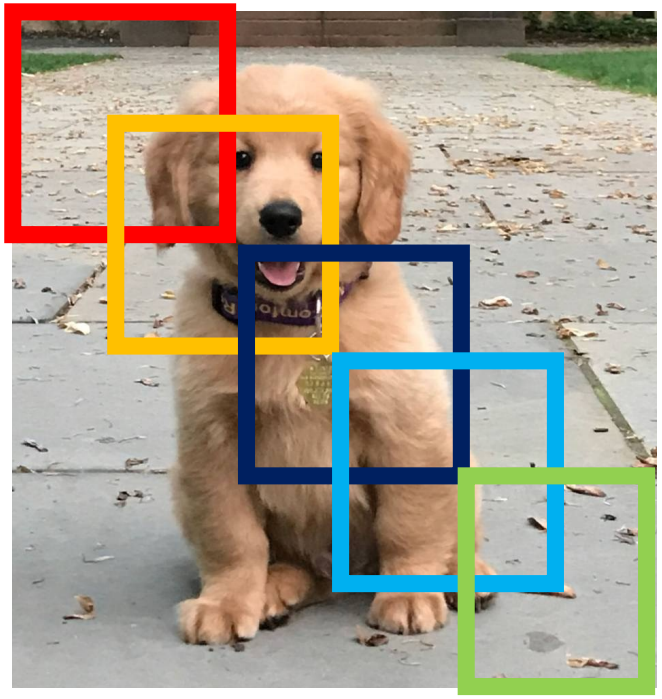


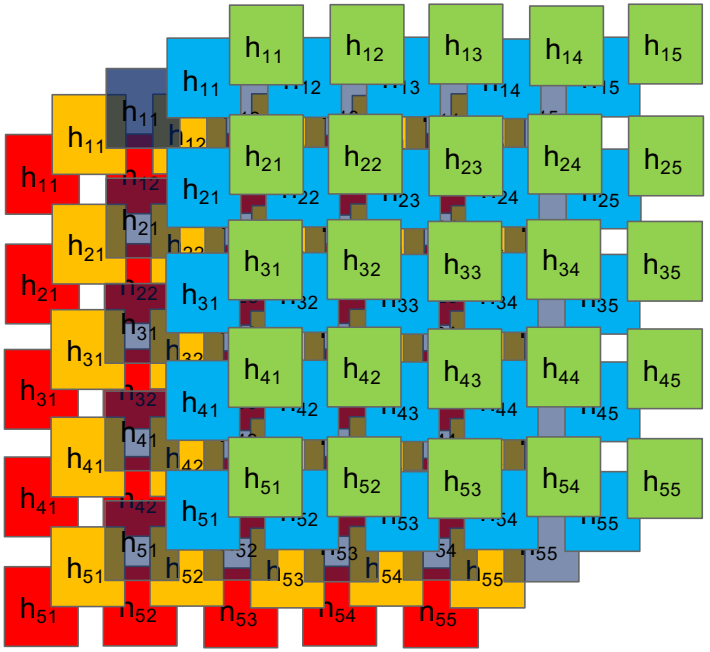
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels



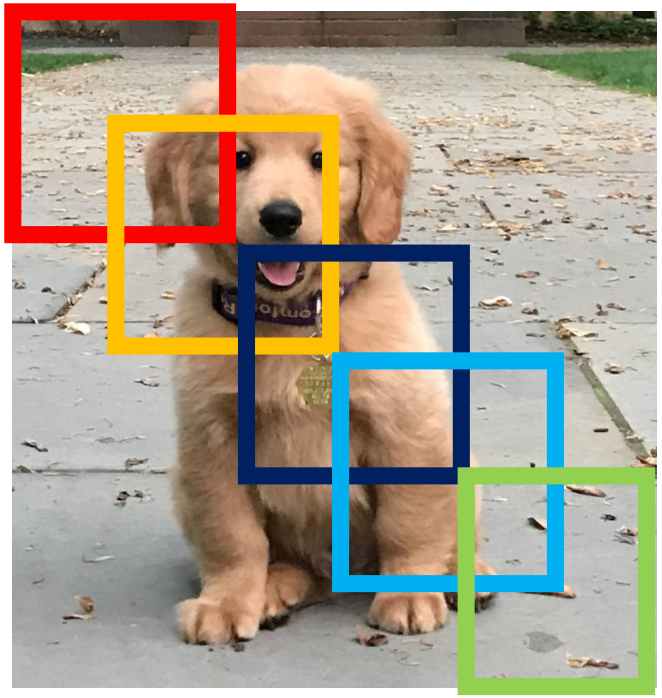


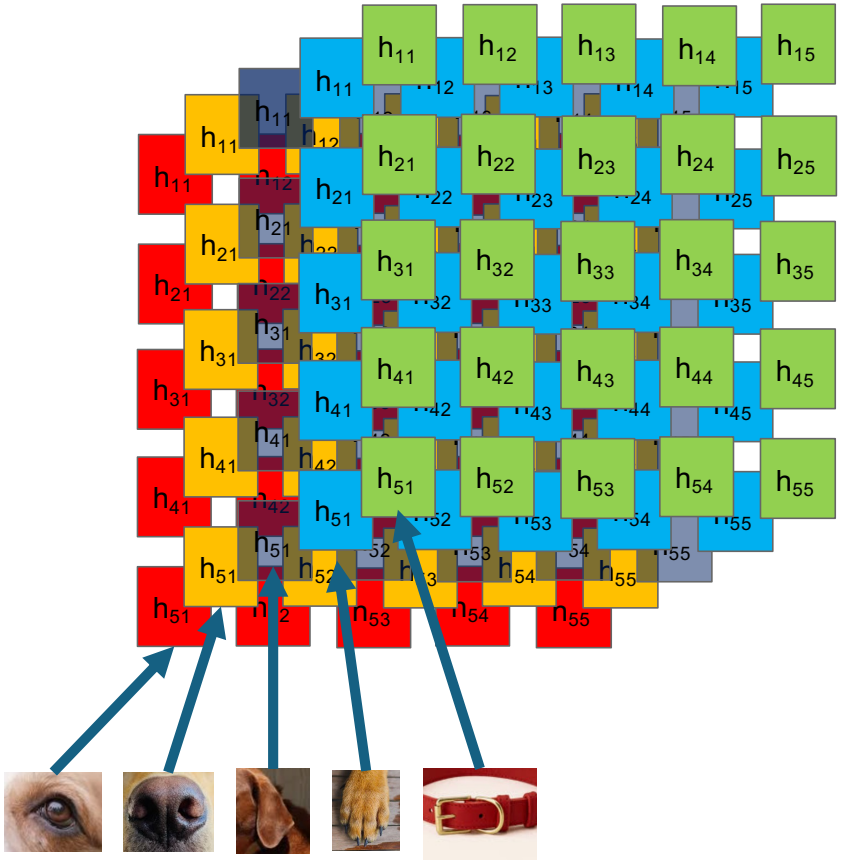
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels



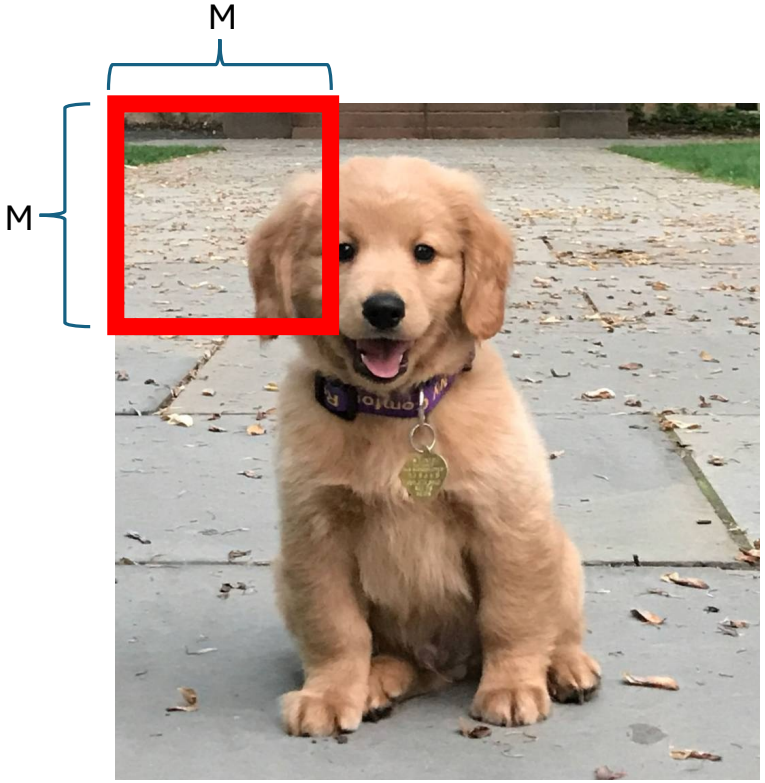


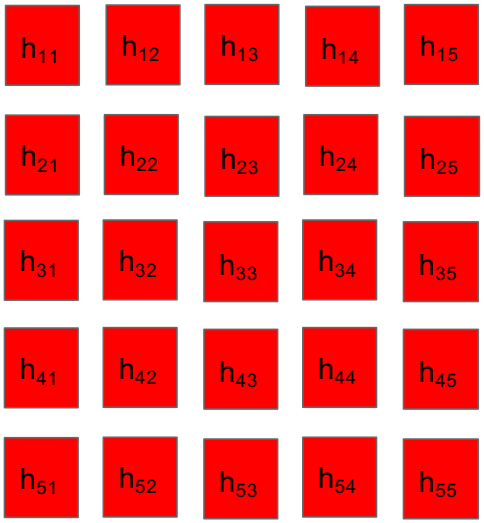
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels





Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels





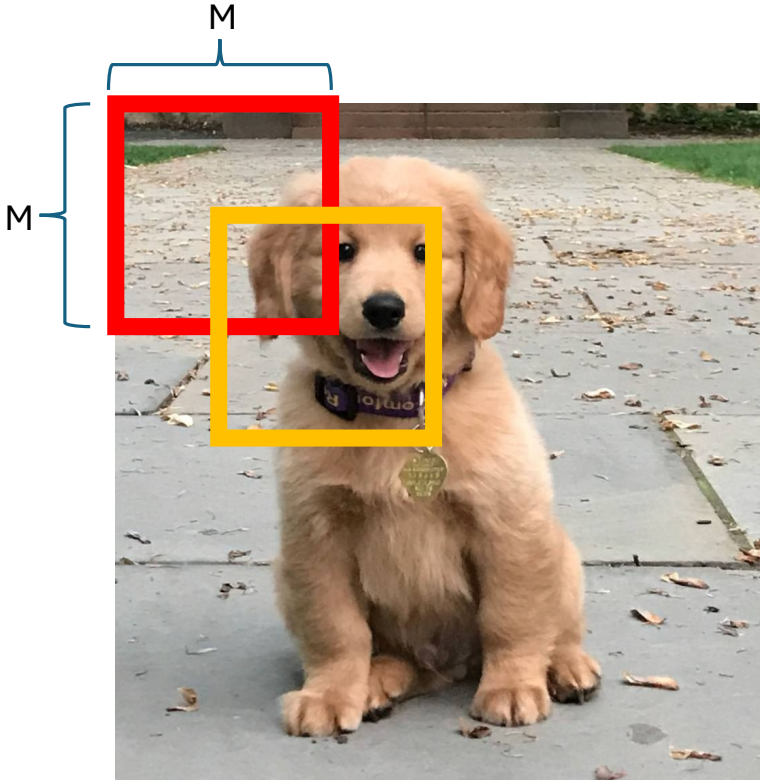
size of filter: M

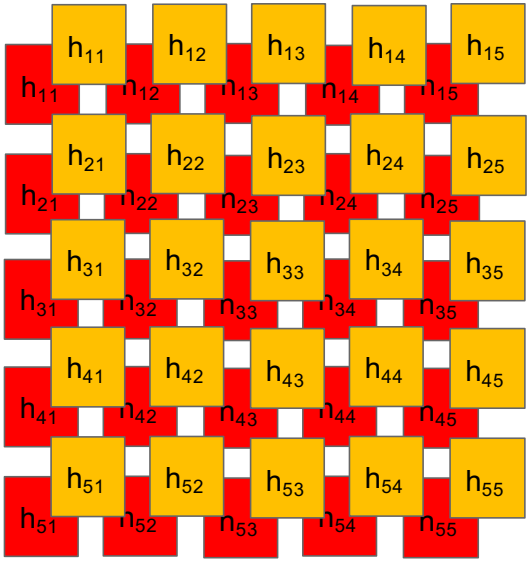
input channels (3 = RGB)

```
self.conv1 = nn.Conv2d(3,1,M)
```

output channels

Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels





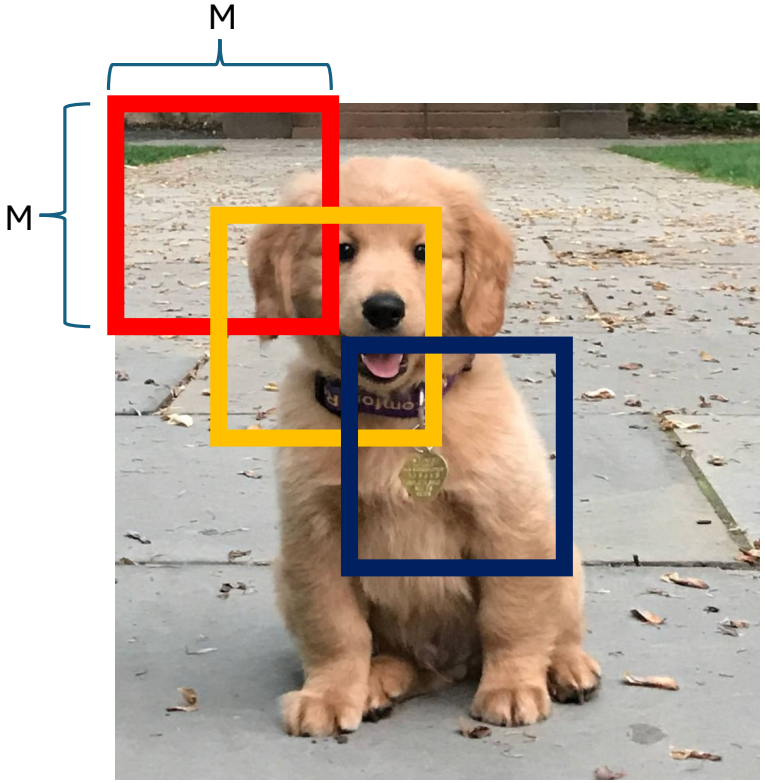
size of filter: M

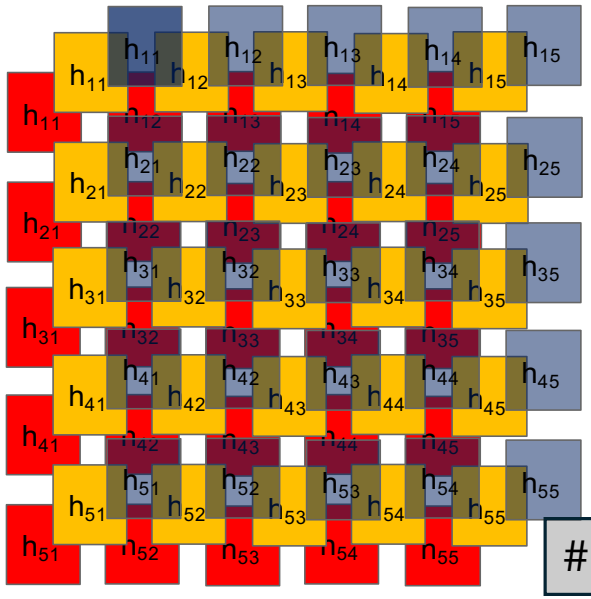
input channels (3 = RGB)

```
self.conv1 = nn.Conv2d(3,2,M)
```

output channels

Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels





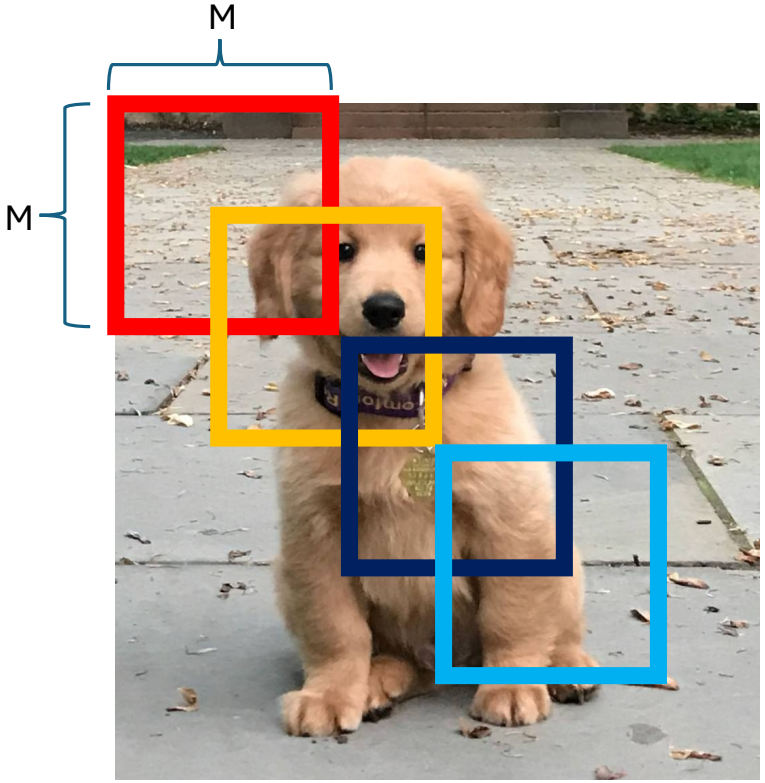
size of filter: M

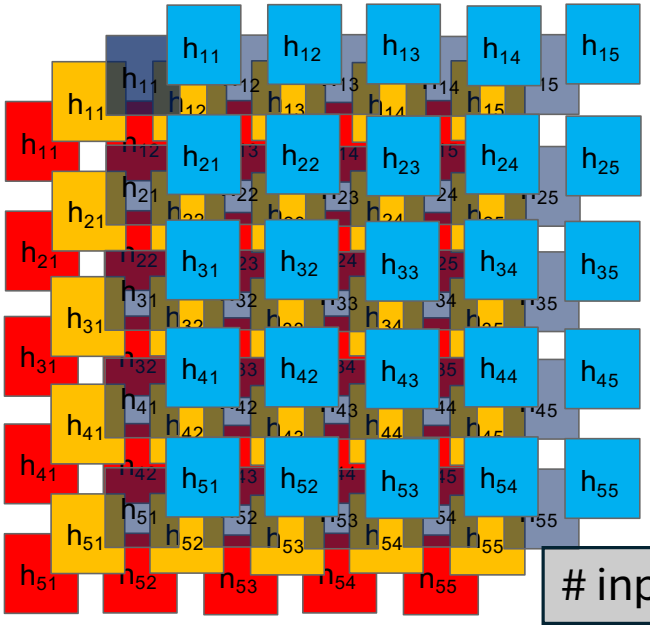
input channels (3 = RGB)

```
self.conv1 = nn.Conv2d(3,3,M)
```

output channels

Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels





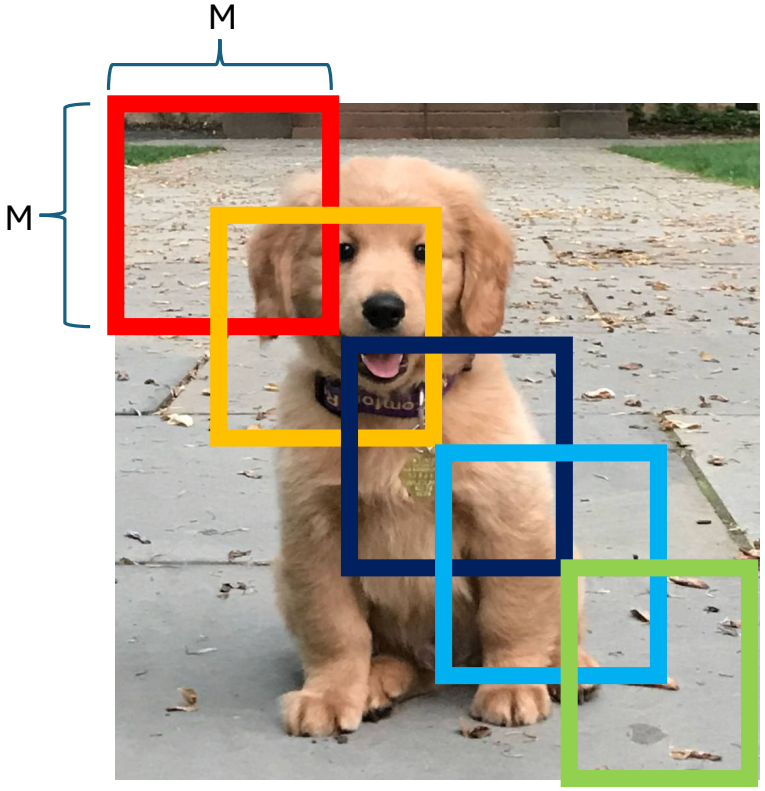
size of filter: M

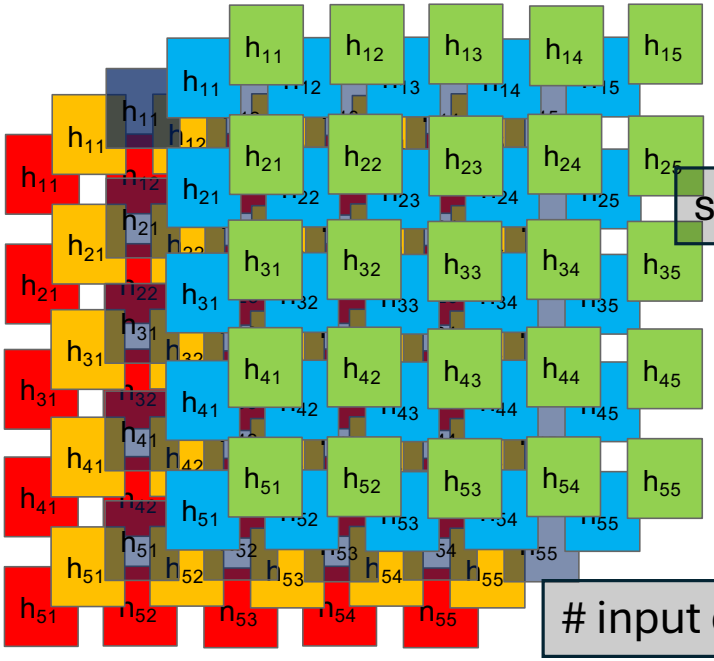
input channels (3 = RGB)

```
self.conv1 = nn.Conv2d(3,4,M)
```

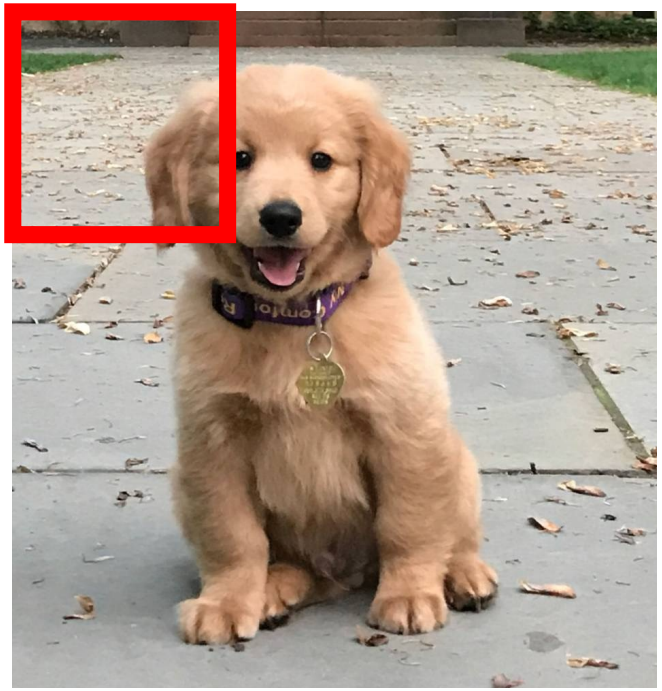
output channels

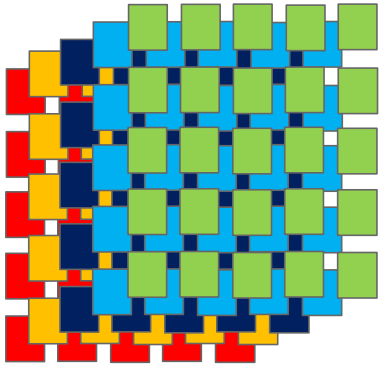
Το ίδιο επίπεδο μπορεί να έχει πολλαπλά channels





Conv2d: # input channels

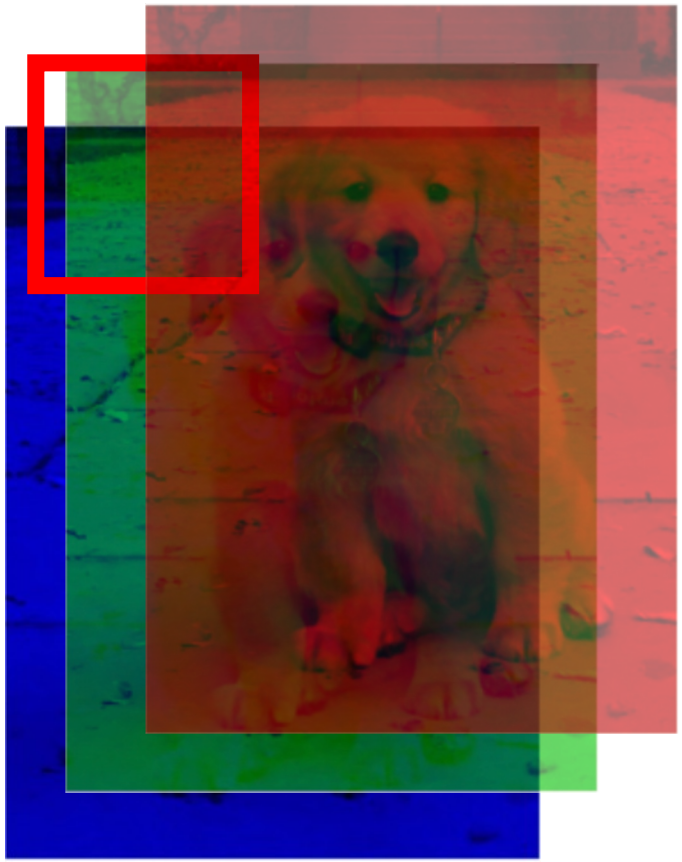


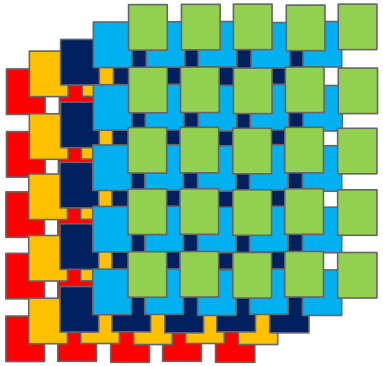


```
self.conv1 = nn.Conv2d(3,5,M,s,p)
```

input channels (3 = RGB)

Conv2d: # input channels

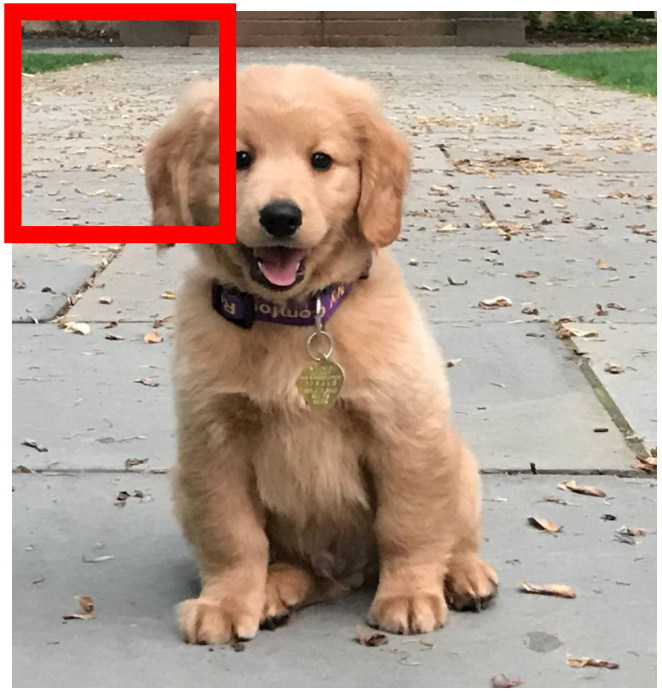


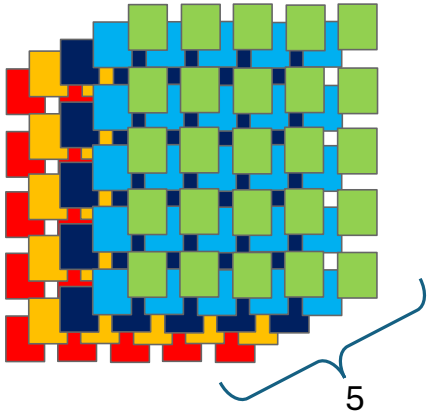


```
self.conv1 = nn.Conv2d(3,5,M,s,p)
```

input channels (3 = RGB)

Conv2d: # output channels

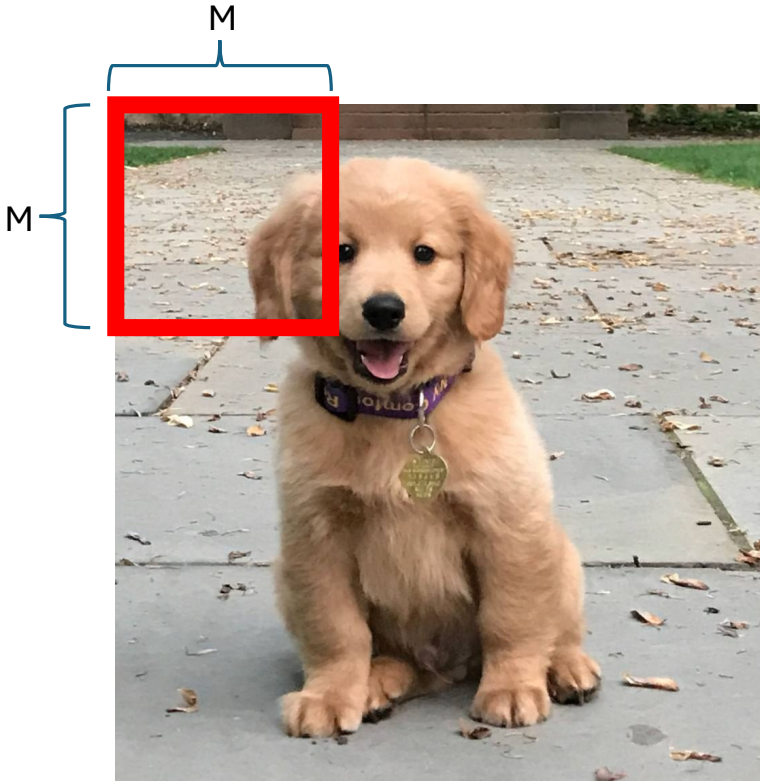


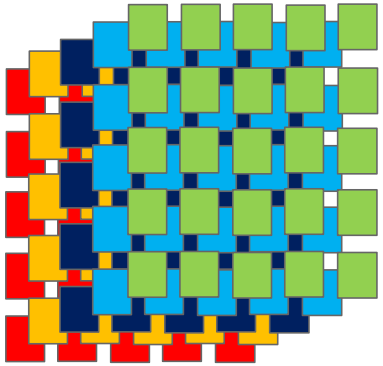


output channels

```
self.conv1 = nn.Conv2d(3,5,M,s,p)
```

Conv2d: size of the filter

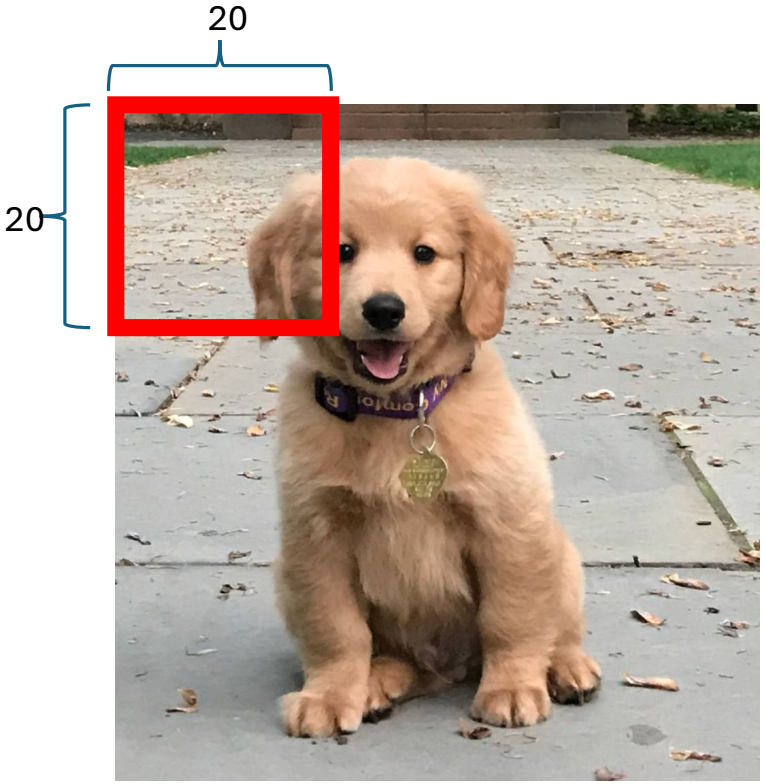


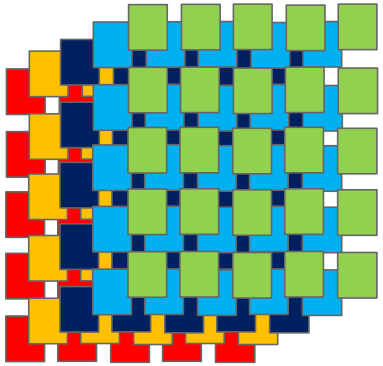


size of filter: M

```
self.conv1 = nn.Conv2d(3,5,M,s,p)
```

Conv2d: size of the filter

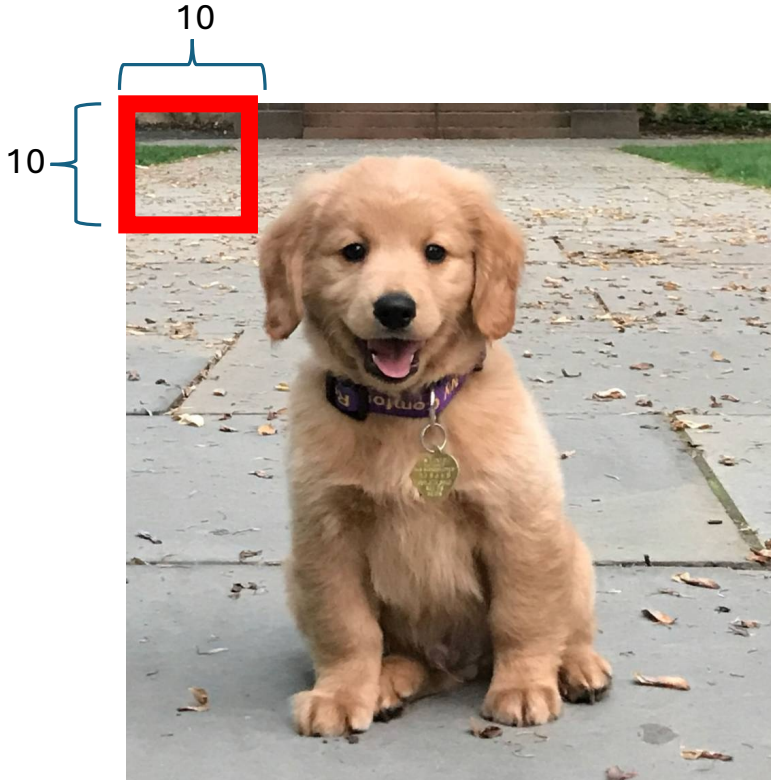


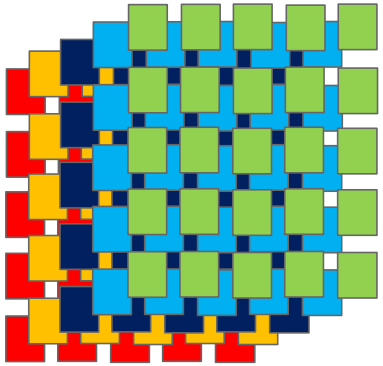


size of filter: M=20

```
self.conv1 = nn.Conv2d(3,5,20,s,p)
```

Conv2d: size of the filter

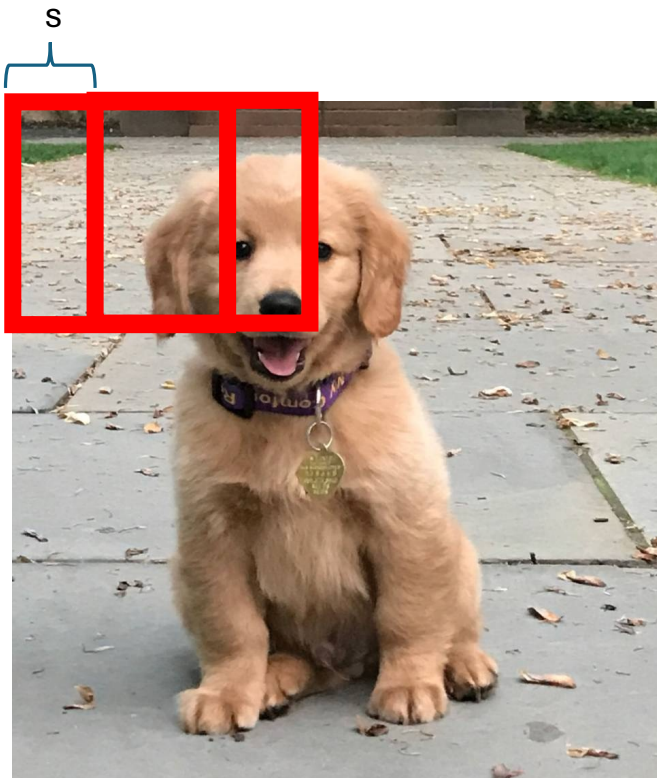


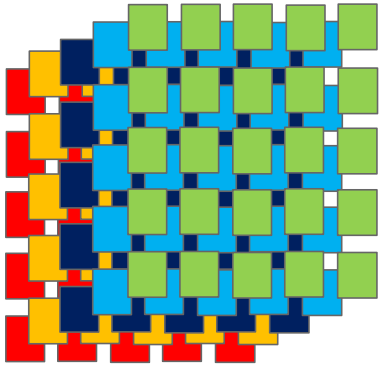


size of filter: M=10

```
self.conv1 = nn.Conv2d(3,5,10,s,p)
```

Conv2d: stride

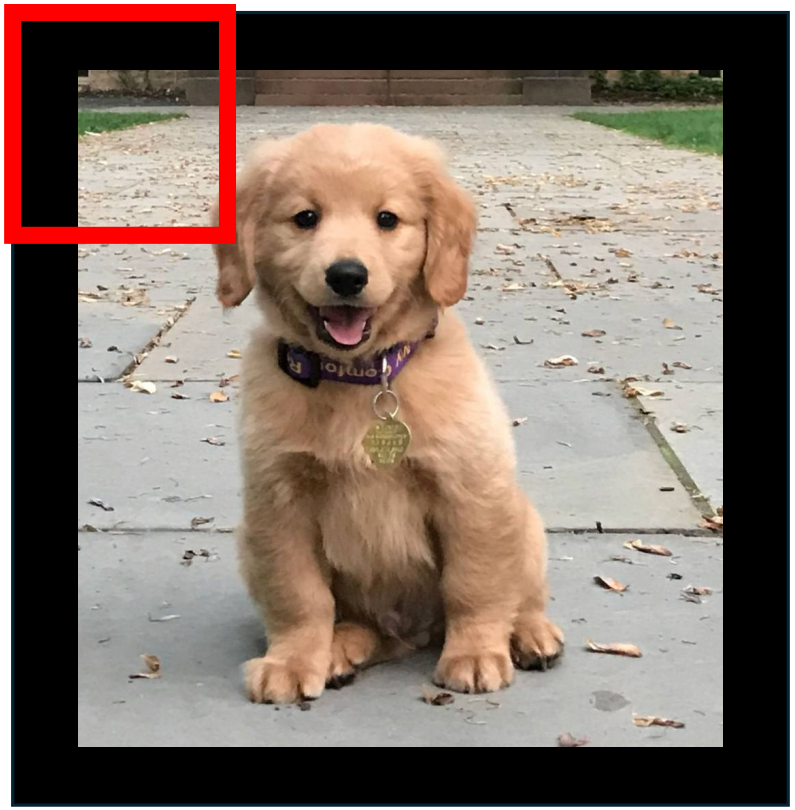


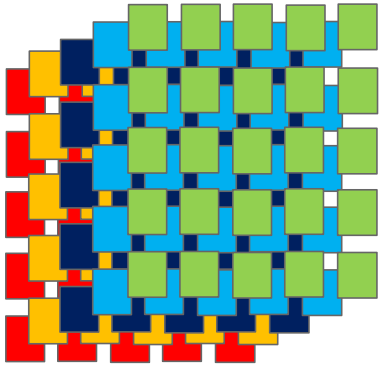


```
self.conv1 = nn.Conv2d(3,5,20,s,p)
```

stride

Conv2d: zero padding

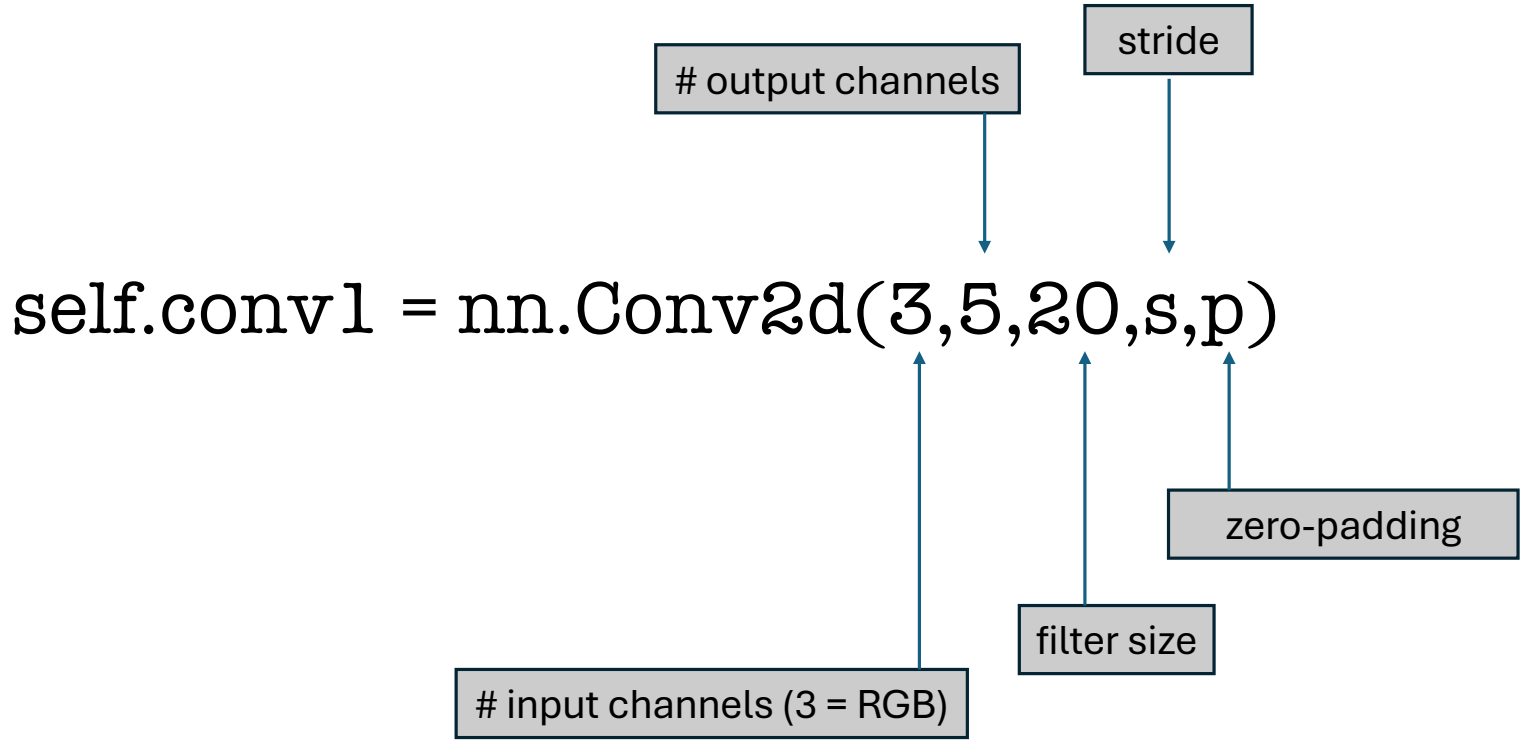




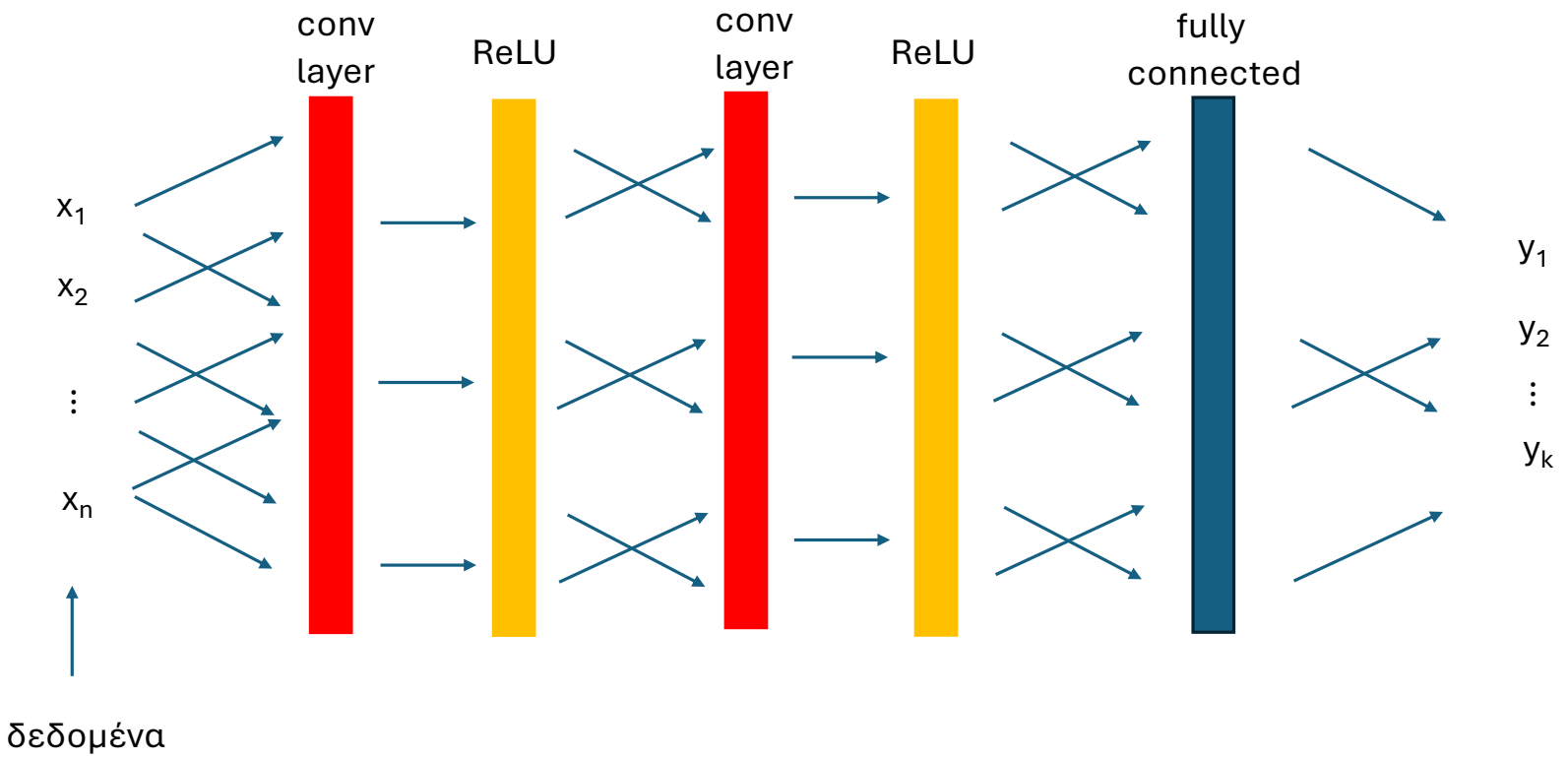
zero padding

```
self.conv1 = nn.Conv2d(3,5,20,s,p)
```

Conv2d

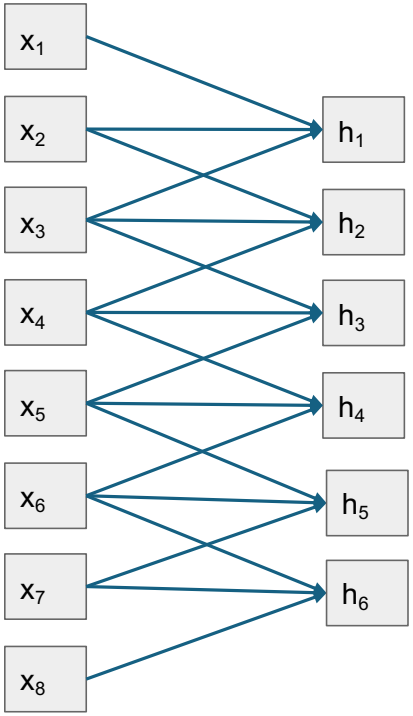

stride
# output channels
self.conv1 = nn.Conv2d(3,5,20,s,p)
zero-padding
filter size
input channels (3 = RGB)

Convolutional + ReLU + Fully Connected



Άλλα χρήσιμα επίπεδα: max & average pooling

Max Pooling



$h_1 = \max(x_1,x_2,x_3)$

$h_2 = \max(x_2,x_3,x_4)$

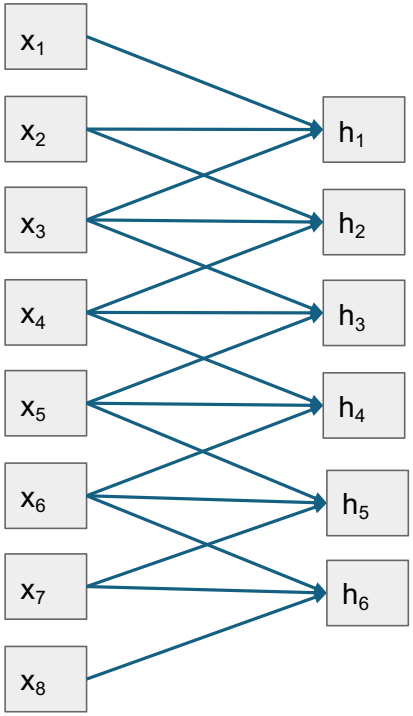
$h_3 = \max(x_3,x_4,x_5)$

$h_4 = \max(x_4,x_5,x_6)$

$h_5 = \max(x_5,x_6,x_7)$

$h_6 = \max(x_6,x_7,x_8)$

Average Pooling



$h_1 = (1/3)(x_1+x_2+x_3)$

$h_2 = (1/3)(x_2+x_3+x_4)$

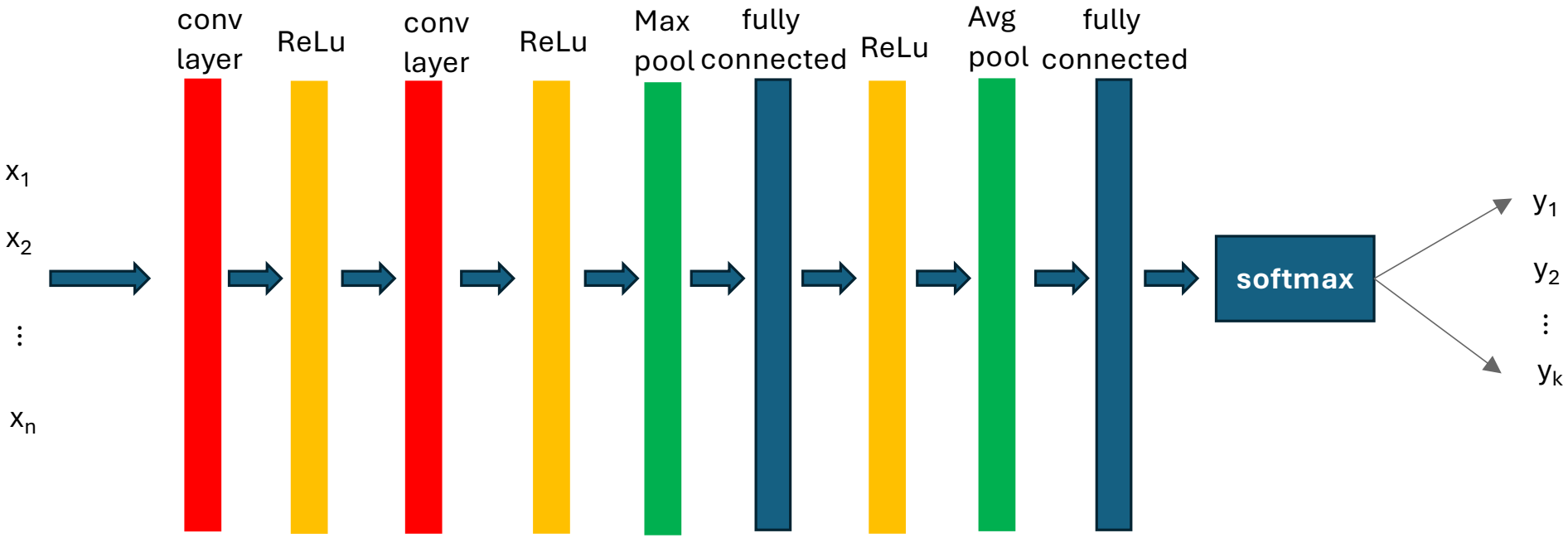
$h_3 = (1/3)\ (x_3+x_4+x_5)$

$h_4 = (1/3)\ (x_4+x_5+x_6)$

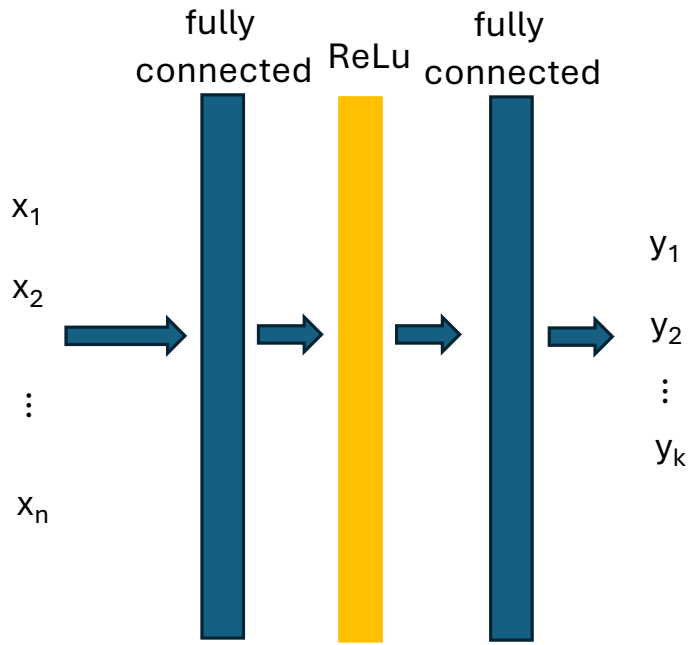
$h_5 = (1/3)\ (x_5+x_6+x_7)$

$h_6 = (1/3)\ (x_6+x_7+x_8)$

Convolutional + ReLU + Fully Connected



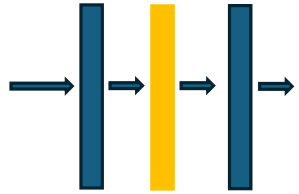
Fully Connected + ReLU




```python
# Define the neural network
class SimpleNet(nn.Module):
    def __init__(self):
        super(SimpleNet, self).__init__()
        self.fc1 = nn.Linear(3, 2)  # 3 inputs to 2 outputs
        self.fc2 = nn.Linear(2, 1)  # 2 inputs to 1 output
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

Fully Connected + ReLU



```
def __init__(self):
    super(SimpleNet, self).__init__()
    self.fc1 = nn.Linear(3, 2)
    self.fc2 = nn.Linear(2, 1)
    self.relu = nn.ReLU()
```

```
def forward(self, x):
    x = self.fc1(x)
    x = self.relu(x)
    x = self.fc2(x)
    return x
```

x_1

x_2

$\vdots$

x_n

y_2

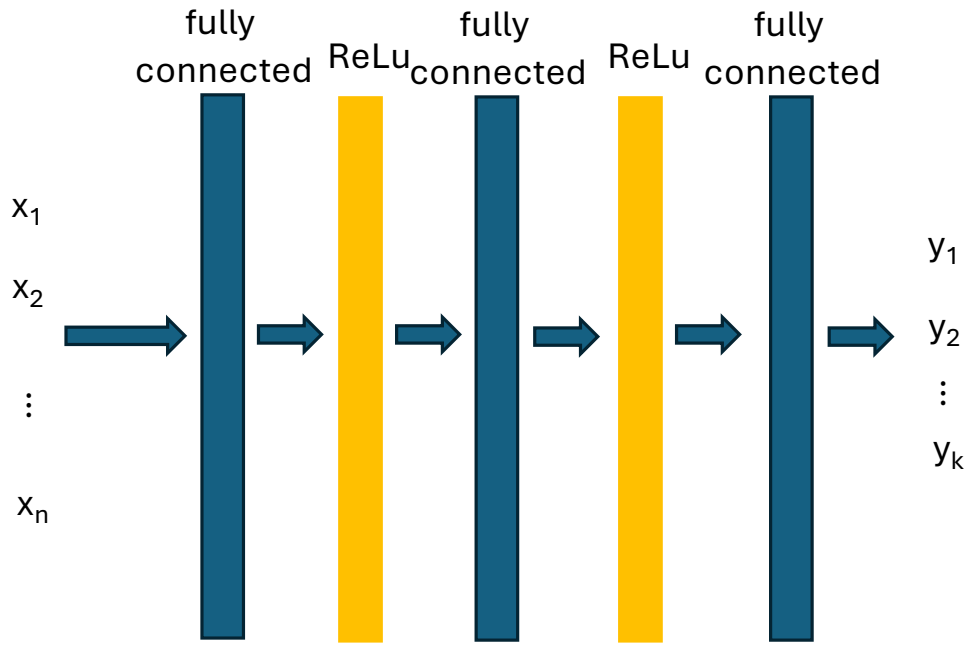
self.fc1 → self.relu → self.fc2

self.fc1

self.relu

self.fc2

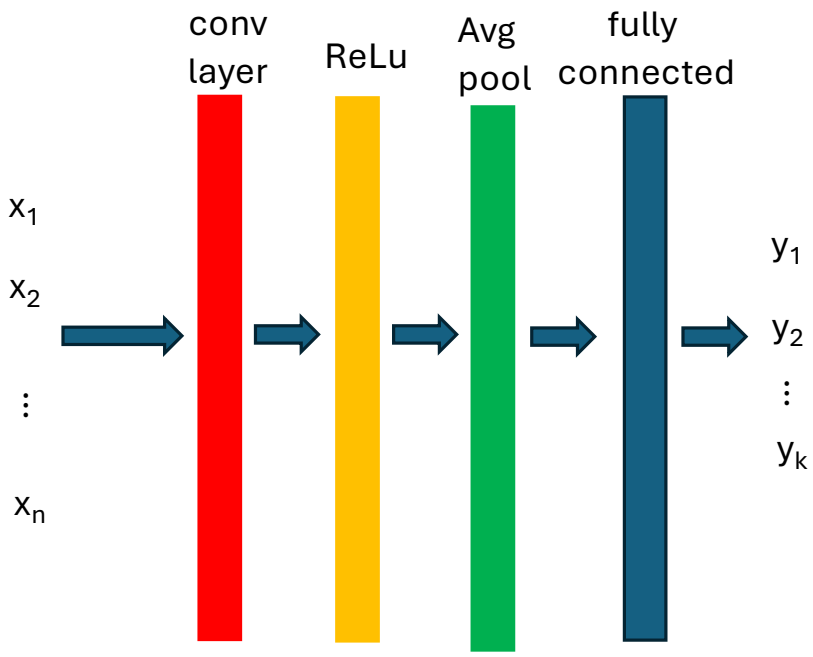
Fully Connected + ReLU




```python
# Define the neural network
class SimpleNet(nn.Module):
    def __init__(self):
        super(SimpleNet, self).__init__()
        self.fc1 = nn.Linear(3, 2)  # 3 inputs to 2 outputs
        self.fc2 = nn.Linear(2, 2)  # 2 inputs to 2 outputs
        self.fc3 = nn.Linear(2, 1)  # 2 inputs to 1 output
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        return x
```

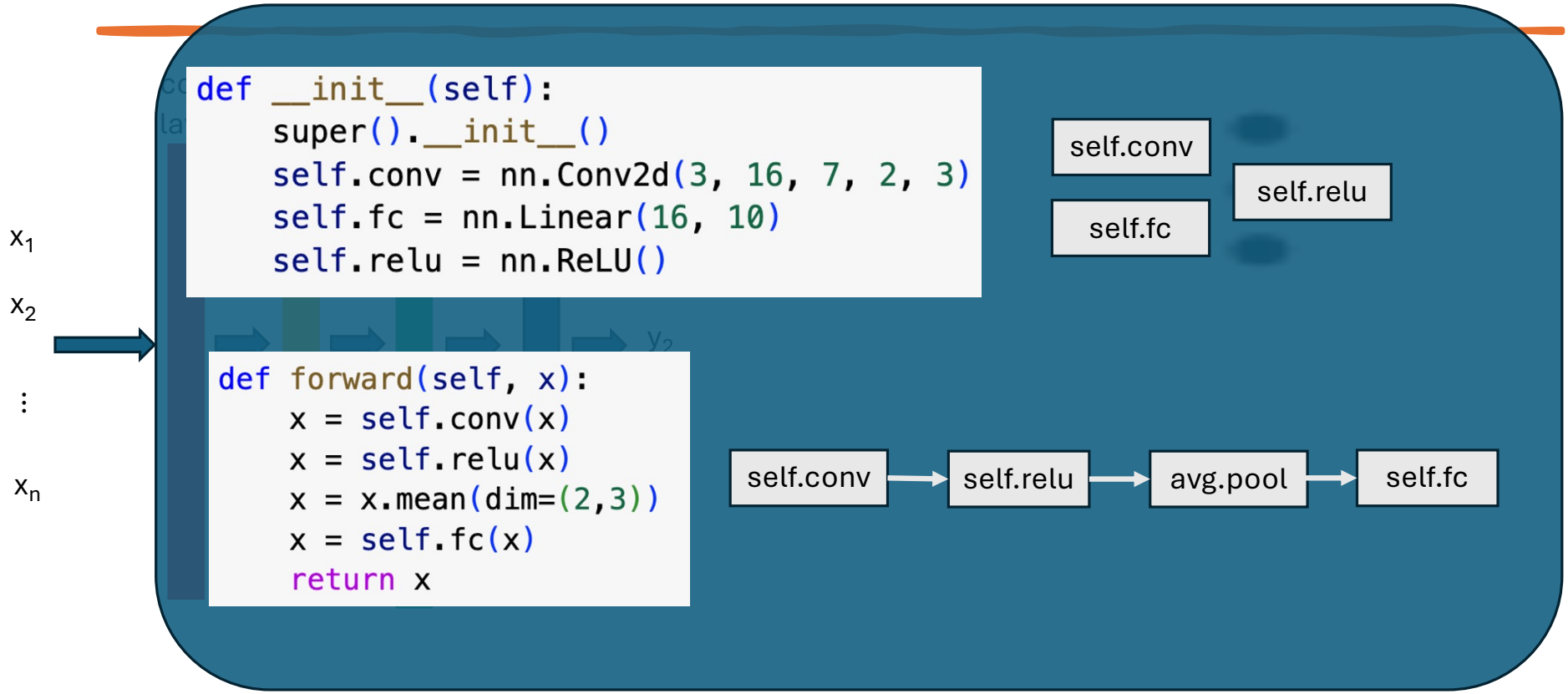
Convolutional + ReLU + Avg Pool + Fully Connected



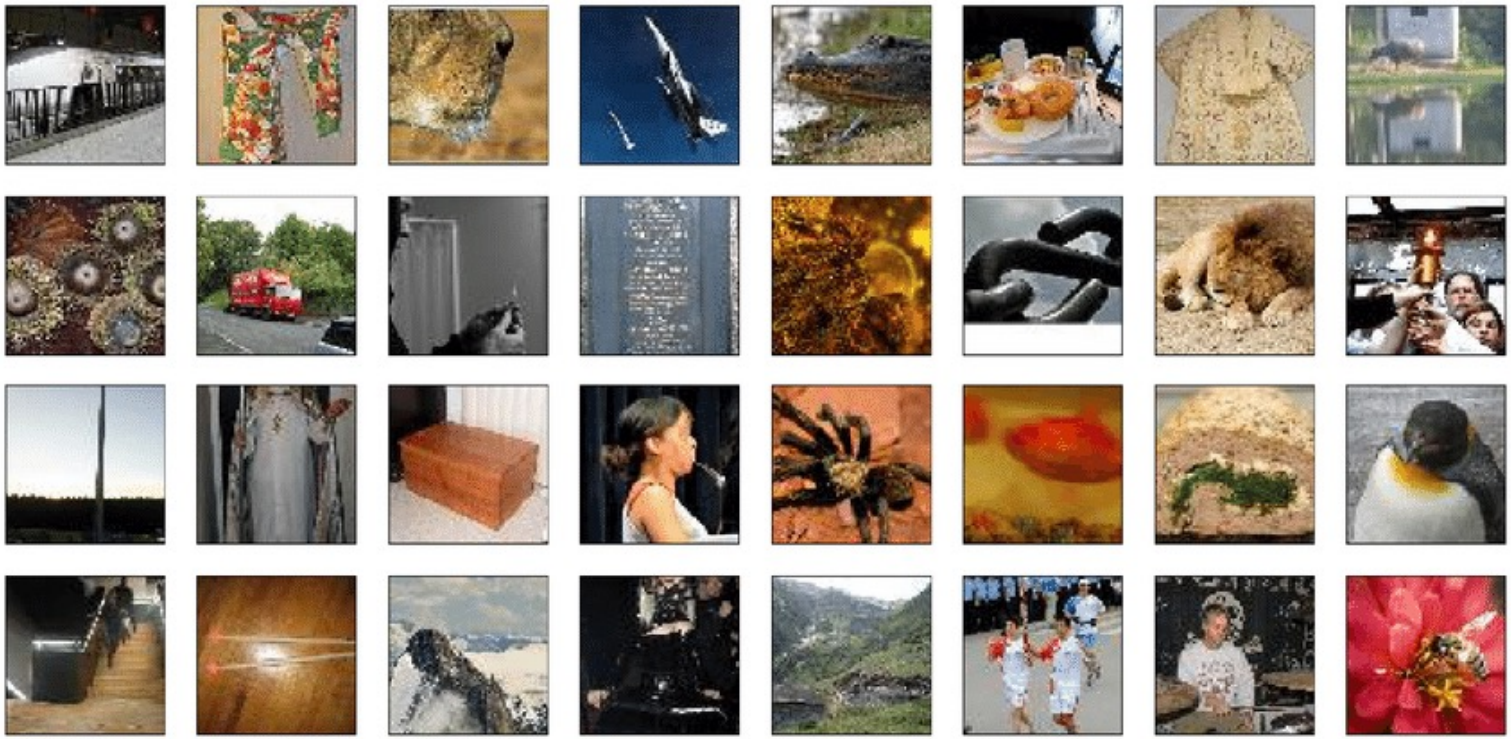

```
class CNNClassifier(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.conv = nn.Conv2d(3, 16, 7, 2, 3)
        self.fc = nn.Linear(16, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.conv(x)
        x = self.relu(x)
        x = x.mean(dim=(2,3))
        x = self.fc(x)
        return x
```

Convolutional + ReLU + Avg Pool + Fully Connected



ImageNet: Σημείο Καμπής στην Ιστορία της Μηχανικής Όρασης



15.000.000 έγχρωμες εικόνες

Μεγέθους 500 x 500

1.000 κατηγορίες

ImageNet

Σημείο Καμπής στην Ιστορία της Μηχανικής Όρασης

Class ID	Class Name
0	tench, Tinca tinca
1	goldfish, Carassius auratus
2	great white shark, white shark, man-eater, man-eating shark, Carcharodon caharias',
3	tiger shark, Galeocerdo cuvieri
4	hammerhead, hammerhead shark
5	electric ray, crampfish, numbfish, torpedo
6	stingray
7	cock
8	hen
9	ostrich, Struthio camelus
10	brambling, Fringilla montifringilla

ImageNet: Σημείο Καμπής στην Ιστορία της Μηχανικής Όρασης

500 x 500 έγχρωμη εικόνα: 500 x 500 x 3 = 750.000 features

Πόσες παραμέτρους έχει ένα fully-connected επίπεδο που έχει τον ίδιο αριθμό νευρώνες με features;

ImageNet: Σημείο Καμπής στην Ιστορία της Μηχανικής Όρασης

500 x 500 έγχρωμη εικόνα: 500 x 500 x 3 = 750.000 features

Πόσες παραμέτρους έχει ένα fully-connected επίπεδο που έχει τον ίδιο αριθμό νευρώνες με features;

500 x 500 x 3 x 500 x 500 x 3 = **$5{,}62 \times 10^{11}$**

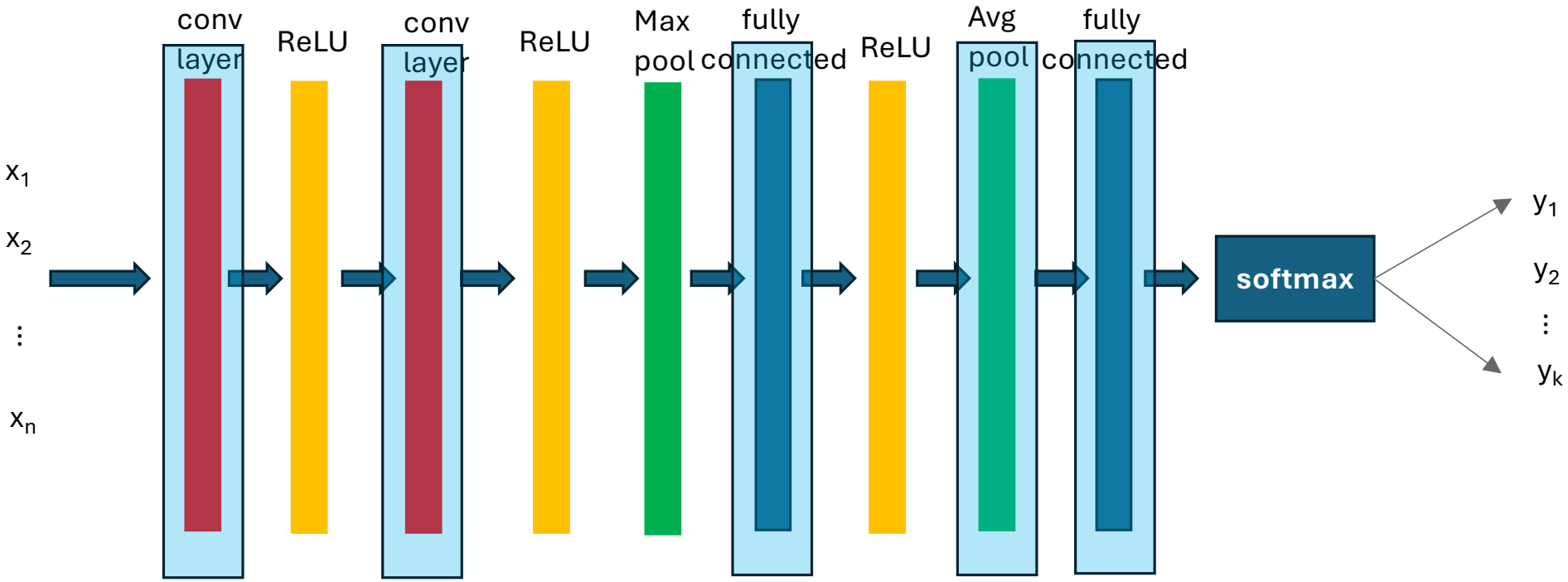
3 φορές το GPT3, 1/3 του GPT4

ImageNet: Σημείο Καμπής στην Ιστορία της Μηχανικής Όρασης

- Τα Convolutional επίπεδα προσφέρουν πάρα πολύ μεγάλη οικονομία παραμέτρων: VGG16 – 165.000.000 παραμέτρους (3.000X λιγότερα από ένα fully connected επίπεδο).
- Ακρίβεια: 92%

(Λείπουν τα ReLU από αυτήν την εικόνα)

Convolutional + ReLU + Fully Connected



Αυτά τα πέντε επίπεδα όλα υπολογίζονται με πρόσθεση και πολλαπλασιασμό

Νευρωνικά Δίκτυα

Πως τα χρησιμοποιούμε

1. Ορίζουμε τα επίπεδα: MyNeuralNetwork χρησιμοποιώντας Conv-layer, Fully-connected-layer, ReLU, MaxPool, AvgPool, SoftMax
2. mymodel = MyNeuralNetwork() – ορίζουμε την οικογένεια
3. train(mymodel,data,optimizer,epochs) – βρίσκουμε παραμέτρους που συμφωνούν με (X,y)
4. mymodel(x) – υπολογίζουμε προβλέψεις για τα x