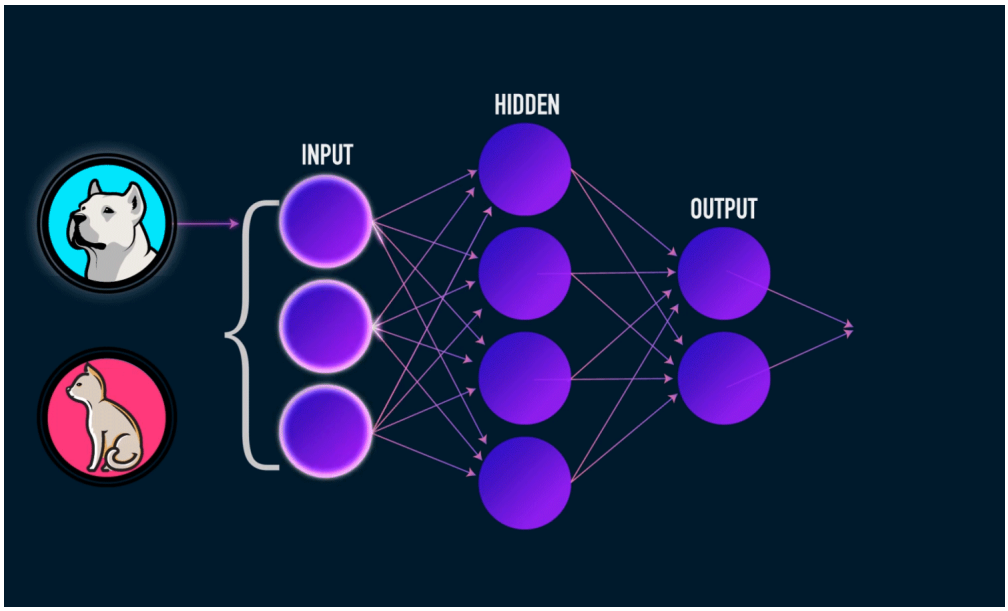
Βαθιά Μάθηση - Deep Learning

Laboratory of Robotics & Automation
Kansizoglou Ioannis, PhD Candidate
ikansizo@pme.duth.gr

Τι είναι ένα (τεχνητό) νευρωνικό δίκτυο ?

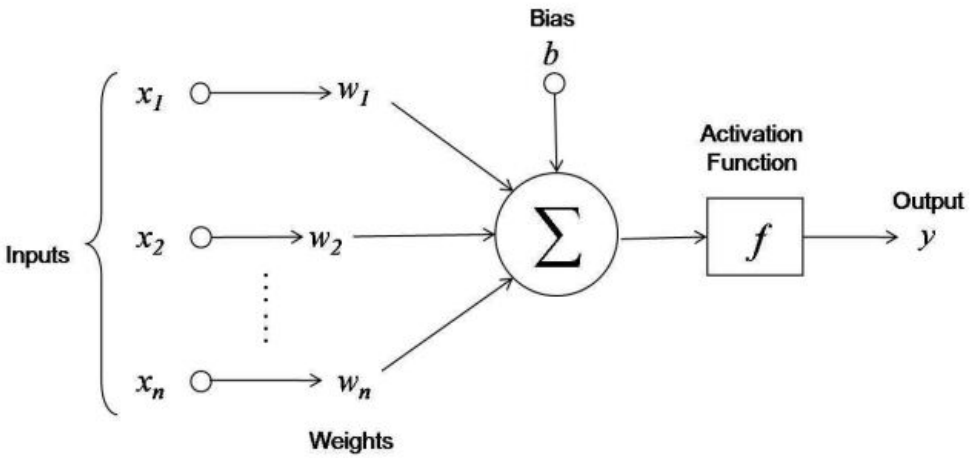


- Είναι ένα δίκτυο από απλούς υπολογιστικούς κόμβους (**νευρώνες**), διασυνδεδεμένους μεταξύ τους.
- Αξιοποιεί μία **απλοποιημένη** μορφή του κεντρικού νευρικού συστήματος (εγκεφάλου).
- Αποτελεί μέθοδο **μηχανικής μάθησης**.

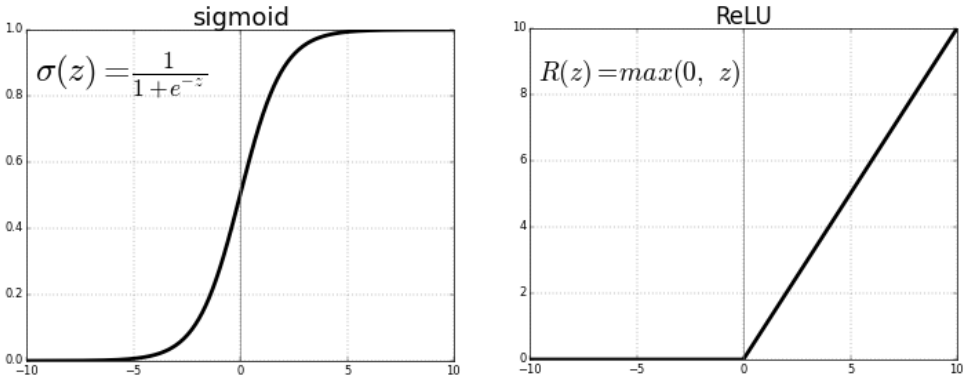
Δομή νευρώνα: μια βαθύτερη ματιά

Κάθε νευρώνας επιστρέφει **μία τιμή**.

$$z = w_1x_1 + w_2x_2 + ... + w_nx_n + b$$



Η συνάρτηση ενεργοποίησης f (activation function)



Στα κρυφά επίπεδα πλέον χρησιμοποιείται σχεδόν αποκλειστικά η **ReLU**.

Συνεπώς, ολοκληρωμένα η έξοδος του νευρώνα είναι:

$$y = ReLU(w_1x_1 + w_2x_2 + ... + w_nx_n+b)$$

Μηχανική μάθηση

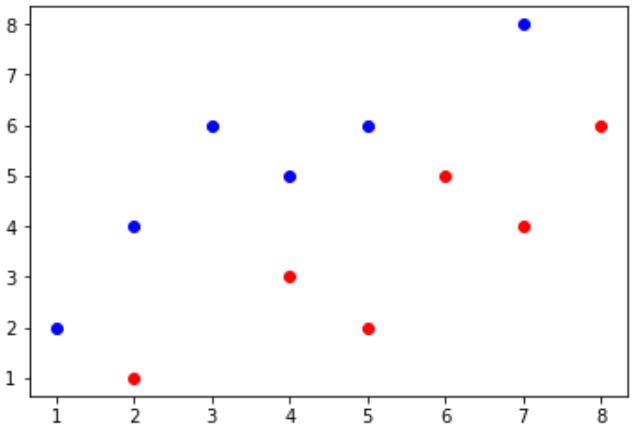
Πώς μαθαίνει ο άνθρωπος ?? ========> Μέσω **εμπειριών !!**

Ορισμός Μηχανικής Μάθησης: Κλάδος που ερευνά μεθόδους για την εκπαίδευση μίας μηχανής ούτως ώστε μέσω εμπειριών (δειγμάτων) να μάθει να διεκπεραιώνει μία συγκεκριμένη εργασία με ορισμένο βαθμό απόδοσης.

Πρακτικά, εκπαιδεύουμε τις παραμέτρους ***w*** μίας συνάρτησης ***F***, με σκοπό όταν δέχεται μια είσοδο ***x*** να παράγει την επιθυμητή έξοδο ***y***. Μαθηματικά μπορούμε να πούμε ότι θέλουμε:

$$y = F(x; w) \approx label$$

Έστω ότι θέλουμε να εκπαιδεύσουμε ένα νευρώνα με συνάρτηση $F(x) = σ(w_0 + w_1x_1 + w_2x_2)$ για να κατηγοριοποιήσουμε τα δεδομένα της παρακάτω εικόνας:



Παρακολουθούμε την τιμή $z = w_0 + w_1x_1 + w_2x_2$:

- Αν z>0 τότε η F>0.5 και ενεργοποιείται ο νευρώνας (κλάση 1)
- Αν z<0 τότε η F<0.5 και δεν ενεργοποιείται ο νευρώνας (κλάση 0)

Τα βήματα της εκπαίδευσης έχουν ως εξής:

1. ΟΡΓΑΝΩΝΟΥΜΕ ΤΑ ΔΕΙΓΜΑΤΑ ΕΚΠΑΙΔΕΥΣΗΣ

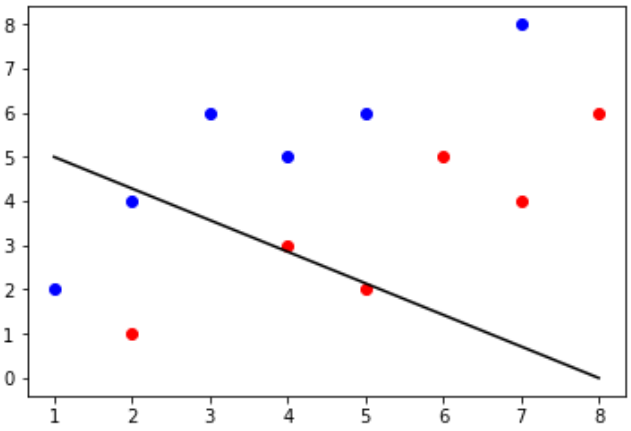
Είναι σημαντικό να ανακατεύουμε τα δεδομένα.

```
[x1,x2,label]
[[2 4 1]
 [5 2 0]
 [1 2 1]
 [4 3 0]
 [7 4 0]
 [5 6 1]
 [3 6 1]
 [4 5 1]
 [7 8 1]
 [8 6 0]
 [6 5 0]
 [2 1 0]]
```

2. ΑΡΧΙΚΟΠΟΙΟΥΜΕ ΤΙΣ ΠΑΡΑΜΕΤΡΟΥΣ (w)

Κακή αρχικοποίηση καθυστερεί, μειώνει την απόδοση ή καθιστά αδύνατη την σύγκλιση.

```
[ w0 ,w1 ,w2 ] = [-8, 1, 1.4]
```

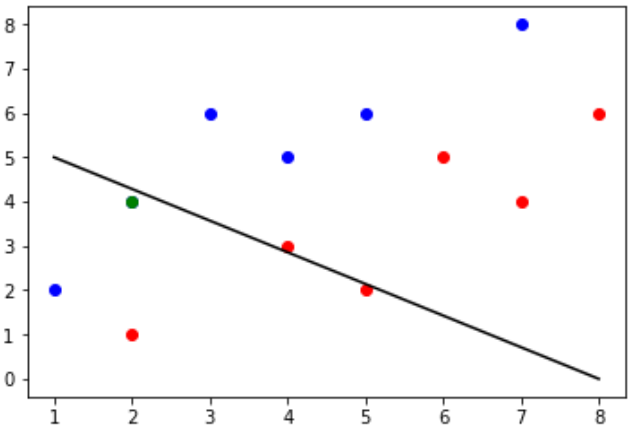


3. ΕΙΣΑΓΟΥΜΕ ΤΟ ΠΡΩΤΟ ΔΕΙΓΜΑ

Υλοποιούμε την πράξη: $w_0 + w_1x_1 + w_2x_2$

Αν το σημείο είναι πάνω από τη γραμμή του νευρώνα το αποτέλεσμα είναι θετικό, διαφορετικά προκύπτει αρνητικό. Στην οριακή περίπτωση που ανήκει στην γραμμή το αποτέλεσμα είναι μηδενικό.

```
Δείγμα 0: [2 4 1]
```




```
z = -0.40000000000000036 και η έξοδος του νευρώνα: y = 0.40131233988754794
Συνεπώς, ο νευρώνας κατηγοριοποιεί το δείγμα 0 στην κόκκινη κλάση.
Λανθασμένη υπόδειξη!
```

4. ΥΠΟΛΟΓΙΖΟΥΜΕ ΤΟ ΣΦΑΛΜΑ

Εισάγουμε την έξοδο του νευρώνα, αλλά και την πραγματική τιμή του δείγματος σε μία συνάρτηση κόστους. Η πιο απλή είναι η τετραγωνική διαφορά:

$$J(y) = \frac{1}{2}(y - label)^2$$

Θέλουμε η έξοδος να προσεγγίζει τις τιμές 0,1 όχι απλά να προβλέπει τις κλάσεις. Στην περίπτωσή μας έχουμε:

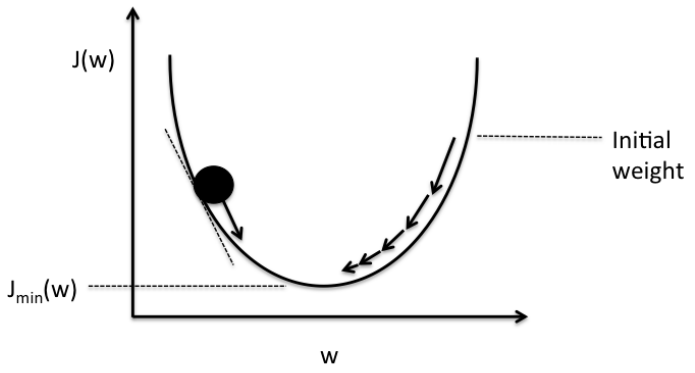
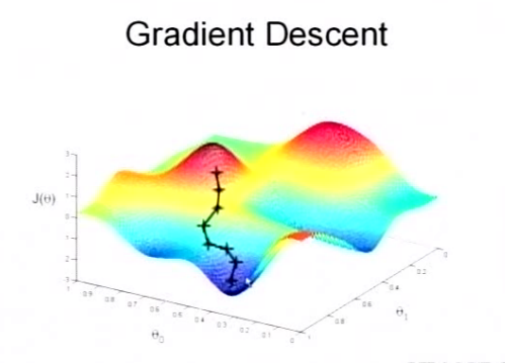
```
Σφάλμα για το δείγμα 0:   loss = 0.17921345718546142
```

5. ΕΠΑΝΑΛΑΜΒΑΝΟΥΜΕ ΤΑ ΒΗΜΑΤΑ 3 & 4 ΓΙΑ ΟΛΑ ΤΑ ΔΕΙΓΜΑΤΑ

Αθροίζουμε τα σφάλματα και υπολογίζουμε το συνολικό σφάλμα του συστήματος

```
Δείγμα 0: [2 4 1]
Έξοδος νευρώνα: y = 0.40131. Λανθασμένη υπόδειξη!
Σφάλμα: loss = 0.1792134572
Δείγμα 1: [5 2 0]
Έξοδος νευρώνα: y = 0.45017. Σωστή υπόδειξη!
Σφάλμα: loss = 0.101324715
Δείγμα 2: [1 2 1]
Έξοδος νευρώνα: y = 0.01477. Λανθασμένη υπόδειξη!
Σφάλμα: loss = 0.4853351043
Δείγμα 3: [4 3 0]
Έξοδος νευρώνα: y = 0.54983. Λανθασμένη υπόδειξη!
Σφάλμα: loss = 0.1511587123
Δείγμα 4: [7 4 0]
Έξοδος νευρώνα: y = 0.99005. Λανθασμένη υπόδειξη!
Σφάλμα: loss = 0.4900977173
Δείγμα 5: [5 6 1]
Έξοδος νευρώνα: y = 0.9955. Σωστή υπόδειξη!
Σφάλμα: loss = 1.01082e-05
Δείγμα 6: [3 6 1]
Έξοδος νευρώνα: y = 0.9677. Σωστή υπόδειξη!
Σφάλμα: loss = 0.0005214985
Δείγμα 7: [4 5 1]
Έξοδος νευρώνα: y = 0.95257. Σωστή υπόδειξη!
Σφάλμα: loss = 0.0011246067
Δείγμα 8: [7 8 1]
Έξοδος νευρώνα: y = 0.99996. Σωστή υπόδειξη!
Σφάλμα: loss = 7e-10
Δείγμα 9: [8 6 0]
Έξοδος νευρώνα: y = 0.99978. Λανθασμένη υπόδειξη!
Σφάλμα: loss = 0.4997752085
Δείγμα 10: [6 5 0]
Έξοδος νευρώνα: y = 0.99331. Λανθασμένη υπόδειξη!
Σφάλμα: loss = 0.4933295462
Δείγμα 11: [2 1 0]
Έξοδος νευρώνα: y = 0.00995. Σωστή υπόδειξη!
Σφάλμα: loss = 4.95192e-05

Συνολικό Σφάλμα: 2.4019401941538843
```

6. ΔΙΟΡΘΩΝΟΥΜΕ ΤΙΣ ΠΑΡΑΜΕΤΡΟΥΣ ΜΑΣ ΜΕΣΗ ΤΗΣ ΜΕΘΟΔΟΥ: GRADIENT DESCENT

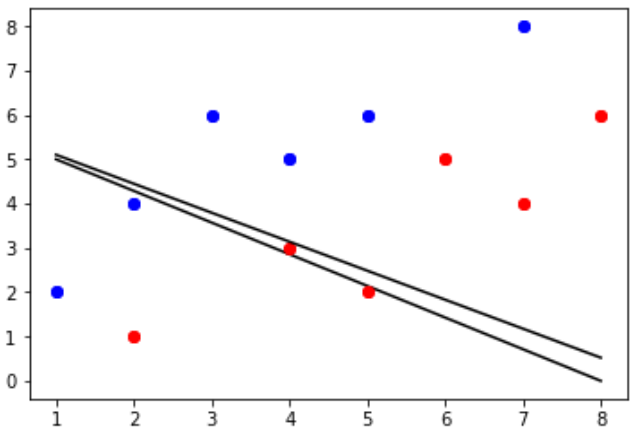
2D	3D
 Schematic of gradient descent.	

Τα βάρη διορθώνονται μέσω της:

$$w_i = w_i - \alpha \frac{\partial J}{\partial w_i}$$

Συνεπώς, υπολογίζουμε μερικές παραγώγους της συνάρτησης κόστους.

```
[w0,w1,w2] = [-8, 1, 1.4]
[w0',w1',w2'] = [-8.010284377556651, 0.9102333729710631, 1.391632618150986]
```



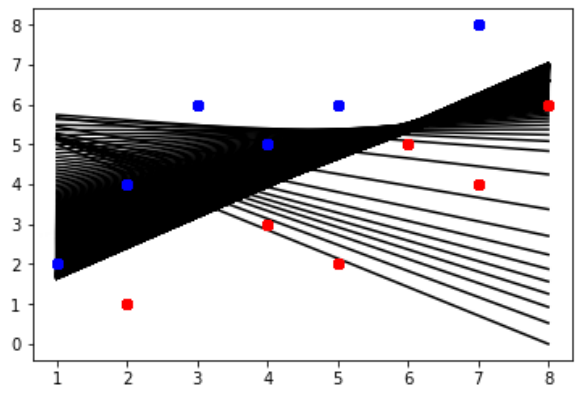
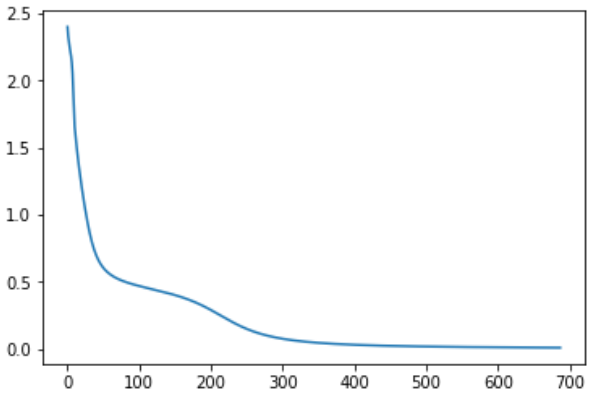
ΑΣ ΕΚΤΕΛΕΣΟΥΜΕ ΟΛΟΚΛΗΡΩΜΕΝΑ ΤΗΝ ΕΚΠΑΙΔΕΥΣΗ

Κάθε επανάληψη, η οποία ορίζεται από την εισαγωγή του πρώτου μέχρι και του τελευταίου δείγματος εκπαίδευσης συμπεριλαμβάνοντας και την εφαρμογή του αλγορίθμου Gradient Descent, ονομάζεται **epoch**.

Εκτελούμε την εκπαίδευση για προκαθορισμένο αριθμό epochs, είτε επαναλαμβάνουμε μέχρι το σφάλμα εκπαίδευσης να ελλατωθεί κάτω από ένα επιθυμητό επίπεδο.

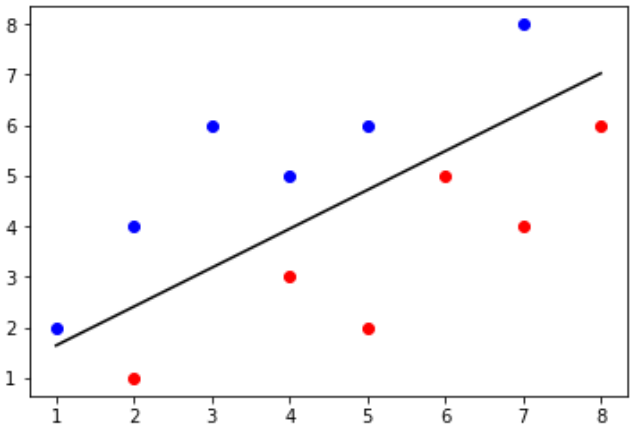
Έστω ότι θέλουμε το σφάλμα να γίνει μικρότερο από 0.01

```
Epoch_0 Συνολικό Σφάλμα: 2.4019401941538843

Epoch_1 Συνολικό Σφάλμα: 2.329520702681356

Epoch_2 Συνολικό Σφάλμα: 2.284788997951755

Epoch_3 Συνολικό Σφάλμα: 2.2504521846307064

Epoch_4 Συνολικό Σφάλμα: 2.218260766279002

Epoch_5 Συνολικό Σφάλμα: 2.1825972045368642

Epoch_6 Συνολικό Σφάλμα: 2.135130764910242

Epoch_7 Συνολικό Σφάλμα: 2.0569776607674903

...

Epoch_685 Συνολικό Σφάλμα: 0.010042452159097071

Epoch_686 Συνολικό Σφάλμα: 0.010016644423407106

Epoch_687 Συνολικό Σφάλμα: 0.009990960773269853
```

Μετατόπιση της γραμμής του νευρώνα	Συνολικό σφάλμα κατά την εκπαίδευση
	

Και το αποτέλεσμα είναι...

```
[ w0 ,w1 ,w2 ] = [-4.7664440430506545, -4.193920379177114, 5.452505839107012]
```



Τι σχέση έχουν όλα αυτά με εμάς...

Συνελικτικά νερωνικά δίκτυα (Convolutional Neural Networks - CNNs)

- Με την χρήση **φίλτρων** και την πράξη της συνέλιξης εντοπίζουμε χρήσιμες πληροφορίες (features maps) απευθείας πάνω στην εικόνα.

  image matrix size: (576, 768)

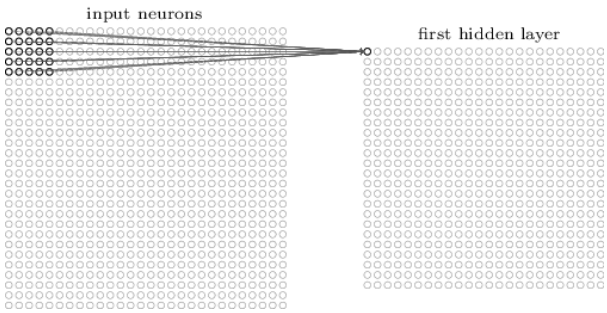
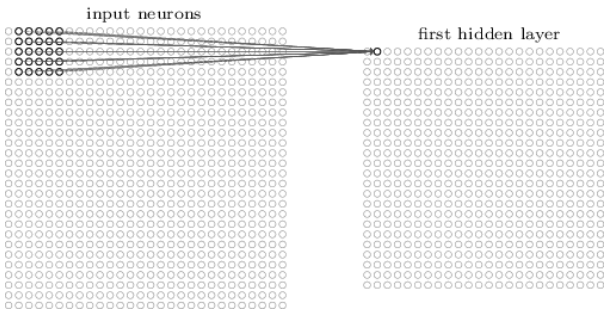
Για παράδειγμα εφαρμόζουμε **Edge Detection** για να τονίσουμε τις **κάθετες** ακμές της εικόνας.

Τι γίνεται όμως στην περίπτωση που δεν γνωρίζουμε τι είδους ακμές πρέπει να εντοπιστούν για να επιλυθεί ένα πρόβλημα?

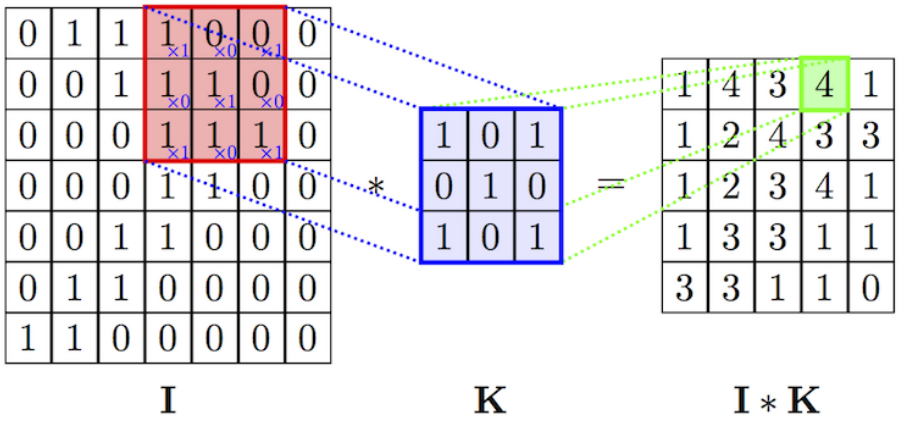
- ***ΛΥΣΗ:*** Εκπαιδεύουμε τα φίλτρα για να αποφασίσουνε μόνα τους να εντοπίζουν τα κατάλληλα στοιχεία στην εικόνα.

CONVOLUTIONAL LAYER

Σε ένα ConvLayer μπορούμε να φανταστούμε τους νευρώνες να έχουν την παρακάτω διάταξη.

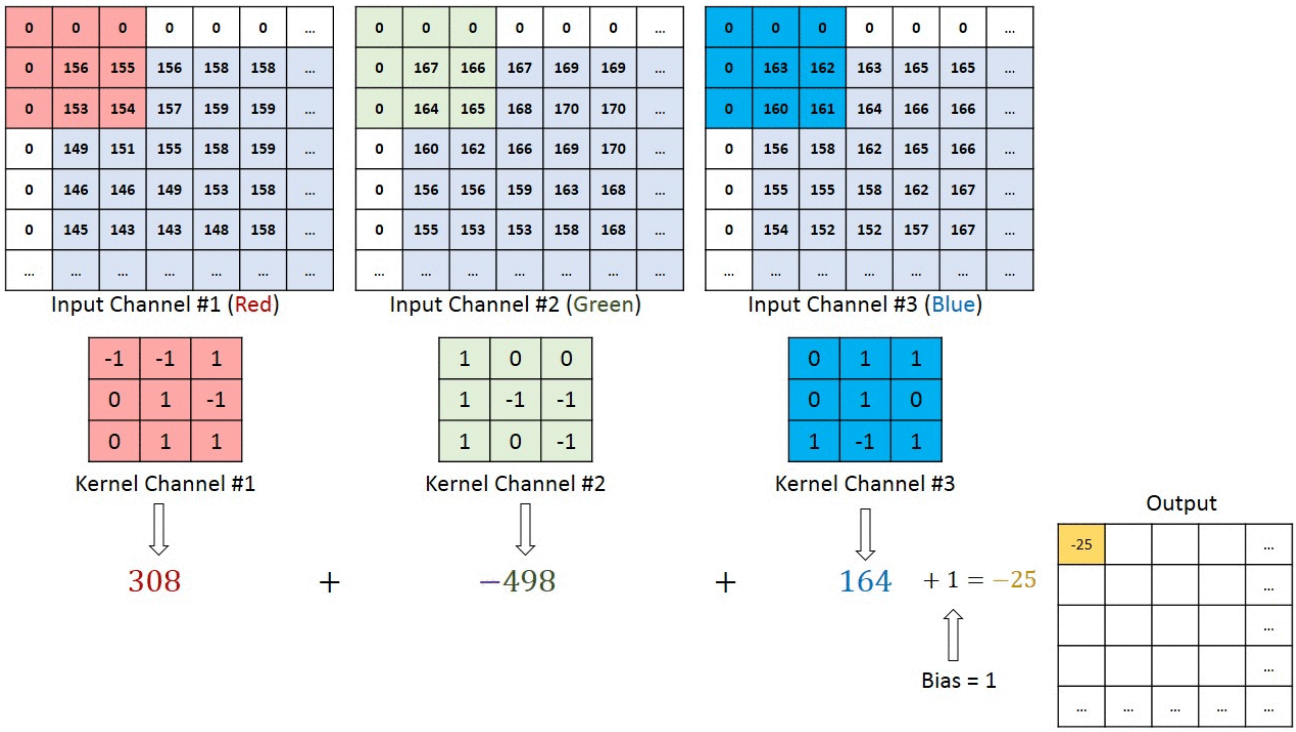
A	B
	

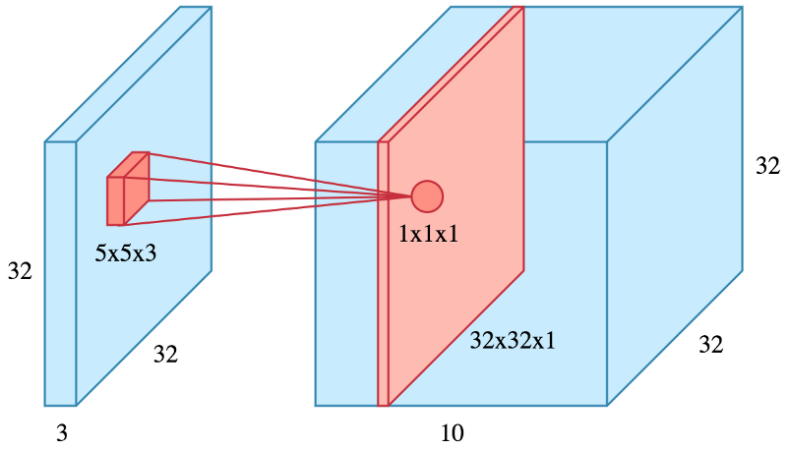
Η τιμή κάθε νευρώνα σε αυτά τα επίπεδα προκύπτει από μία τετραγωνική **περιοχή** των νευρώνων του προηγούμενου επιπέδου. Περιέχει δηλαδή χωρική πληροφορία, που χρειάζεται στις εικόνες.



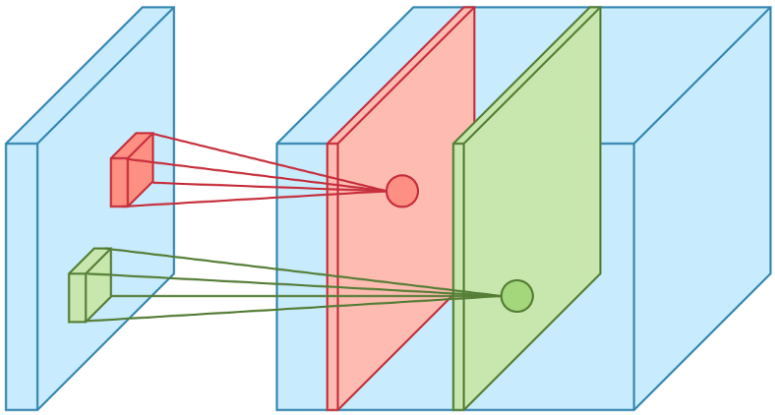
Η περιοχή αυτή συνελίσσεται με τα φίλτρα του επιπέδου (μπλε τετράγωνο) που περιέχουν τις εκπαιδευόμενες παραμέτρους και παράγουν ορισμένους χάρτες χαρακτηριστικών (***feature maps***). Το μέγεθος του φίλτρου (K) καθορίζει και το μέγεθος της περιοχής.

Στην παραπάνω εικόνα έχουμε σαν είσοδο εικόνα ενός καναλιού π.χ. *grayscale*. Για *RGB* εικόνες χρειαζόμαστε φίλτρο τριών καναλιών ένα για κάθε χρώμα, όπως φαίνεται παρακάτω:



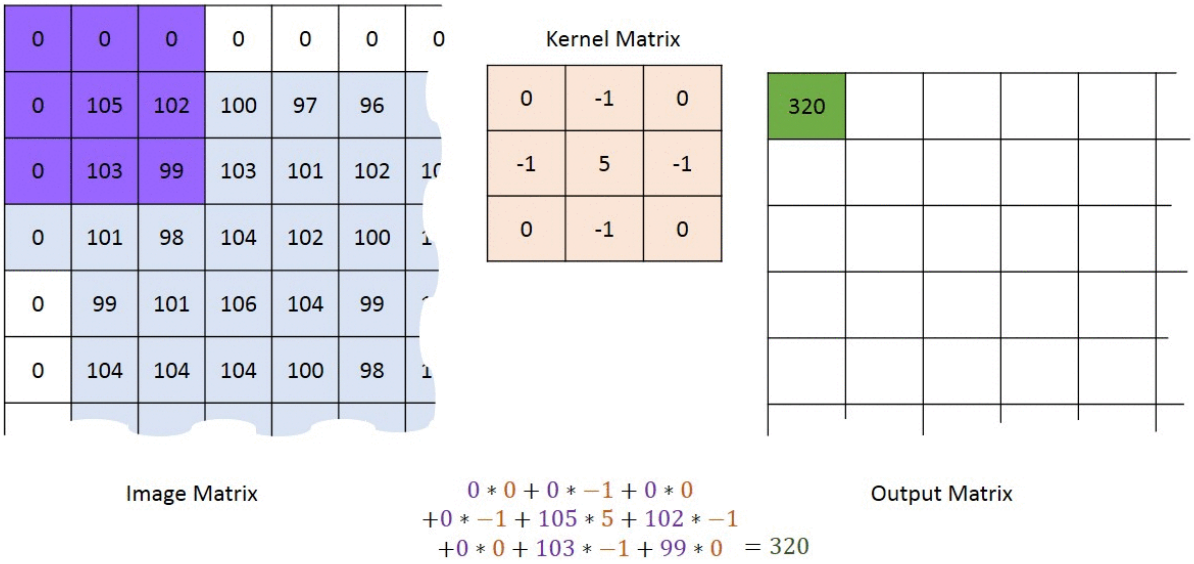


Επίσης θέλουμε να αναζητούμε όχι ένα αλλά περισσότερα στοιχεία σε μια εικόνα. Συνεπώς χρησιμοποιούμε πολλαπλά φίλτρα που παράγουν διαφορετικούς feature maps. Τοποθετώντας τους τον ένα πίσω από τον άλλο ορίζουμε το βάθος (**depth**) του επιπέδου.



Βασικά χαρακτηριστικά συνέλιξης εικόνας:

- **Stride (S):** Ο αριθμός των pixels που μετατοπίζεται το φίλτρο στον οριζόντιο ή τον κατακόρυφο άξονα. Παρακάτω S=2 ή [2,2].
- **Padding (P):** Ο αριθμός των pixels που προσθέτουμε γύρω-γύρω από την είσοδο στην κάθε διεύθυνση. Παρακάτω P=1 ή [1,1].



Convolution with horizontal and vertical strides = 2

Το μέγεθος του feature map έχει 3 διαστάσεις [$width \times height \times depth$] και αντιστοιχεί στον **αριθμό των νευρώνων του επιπέδου**. Οι διαστάσεις καθορίζονται ως:

$$width = \frac{I_w - K_w + 2P}{S} + 1$$

$$height = \frac{I_h - K_h + 2P}{S} + 1$$

$$depth = Number\_of\_filters$$

Αντίστοιχα οι διαστάσεις των παραμέτρων του επιπέδου έχουν ως εξής:

$$weights = [K_w \times K_h \times channels \times depth]$$

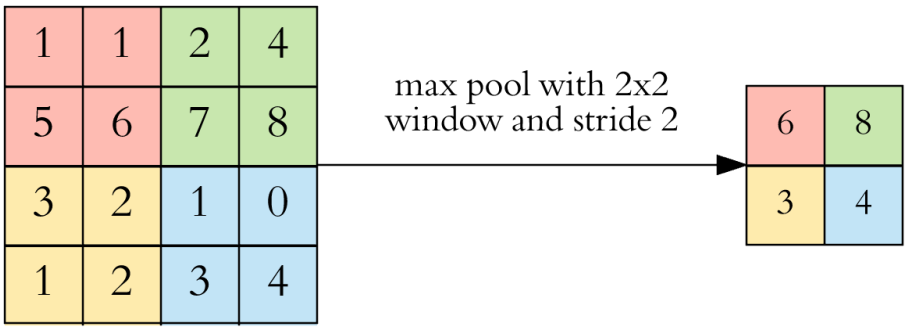
$$bias = [depth]$$

ΠΡΟΣΟΧΗ: Τα κανάλια (channels) καθορίζονται από τον αριθμό φίλτρων (άρα το depth) του προηγούμενου επιπέδου.

POOLING LAYER

Σε ένα PoolLayer απλά κάνουμε μείωση μεγέθους (**downsample**) της εικόνας. Βασικές μέθοδοι pooling είναι: max, min, average. Συνήθως, στα δίκτυα εφαρμόζουμε **max pool**.

Max Pool: Ένα φίλτρο μεγέθους [$K_w \times K_h$] σαρώνει **ανά μονάδα βάθους (depthwise)** το εισαγόμενο επίπεδο και διατηρεί μόνο την μέγιστη τιμή της κάθε περιοχής.



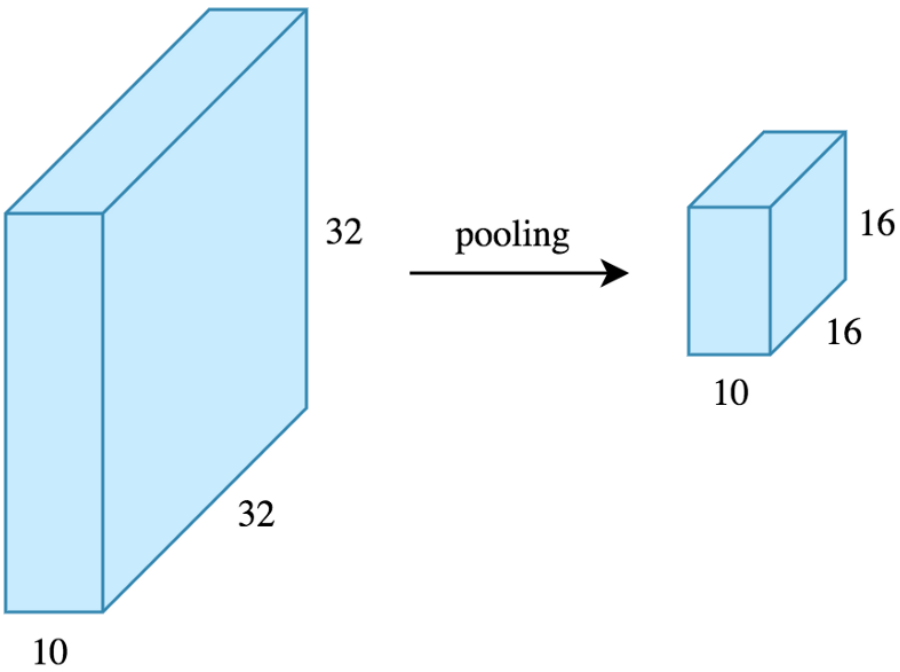
Παρόμοια έχουμε:

- **Stride (S)**: Συνήθως S=K ή $[K_w, K_h]$
- **Padding (P)**: Συνήθως P=0 ή [0,0]

Υπό αυτές τις συνθήκες στην έξοδο προκύπτει feature map μεγέθους:

$$[(\frac{I_w - K_w}{S} + 1) \times (\frac{I_h - K_h}{S} + 1) \times depth] = [\frac{I_w}{K_w} \times \frac{I_h}{K_h} \times depth]$$

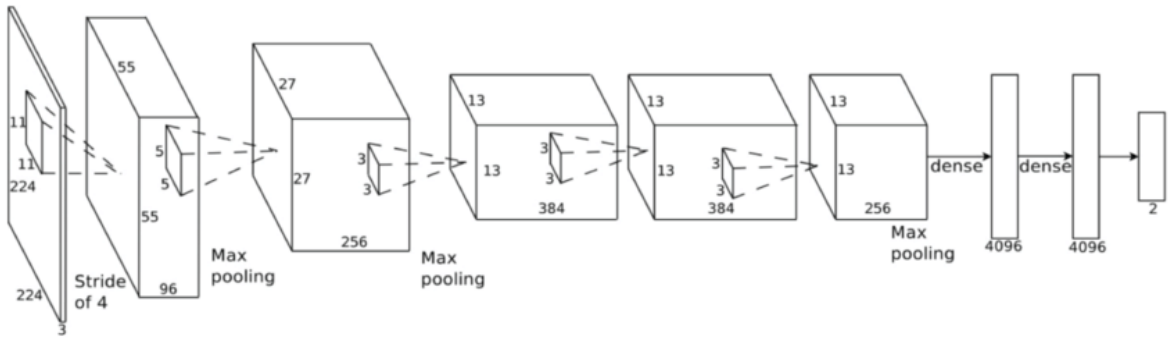
Για παράδειγμα η παραπάνω εικόνα από $[4 \times 4 \times 1]$ συμπιέστηκε σε $[2 \times 2 \times 1]$ pixels. Η μορφή του στις 3 διαστάσεις έχει ως εξής:



ΠΡΟΣΟΧΗ: Το PoolLayer **δεν** εχει εκπαιδευόμενες παραμέτρους. Μειώνει το μέγεθος του επιπέδου της εισόδου και διατηρεί το βάθος της.

AlexNet

Σχεδιάστηκε από τον *Alex Krizhevsky* και εκπαιδεύτηκε στο *ImageNet*.

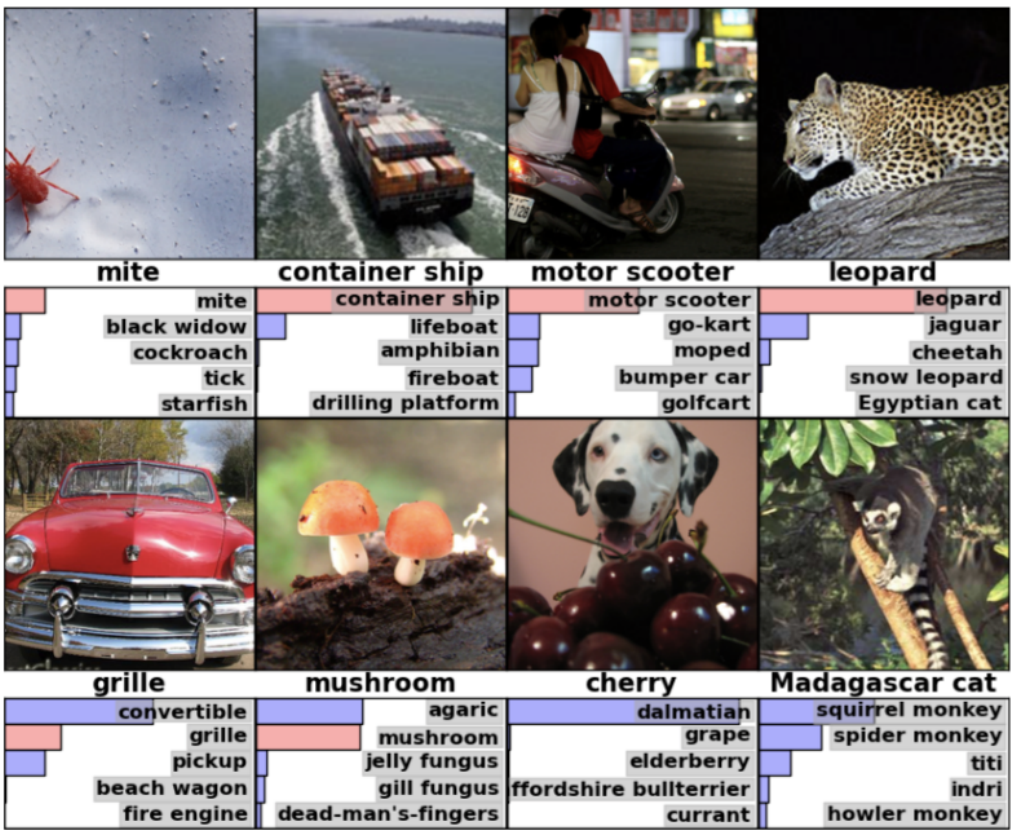


Paper: Imagenet classification with deep convolutional neural networks, *Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E, Advances in neural information processing systems, 2012*

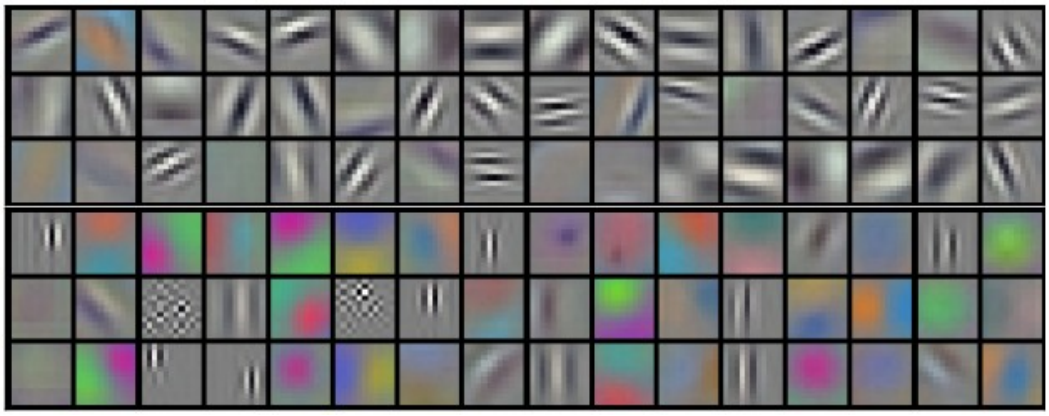
IMAGENET

~ 14 εκατομμύρια εικόνες

- 1.2 M, εικόνες εκπαίδευσης (training)
- 100 K, εικόνες επαλήθευσης (test)
- 1000 κλάσεις



Και τι υπάρχει μέσα σε ένα CNN ?



Size/Operation	Filter	Stride	Padding
227x227x3	-	-	-
Conv1+ReLU	11x11x3x96	4	0
55x55x96	-	-	-
Max Pool	3x3x96	2	0
27x27x96	-	-	-
Conv2+ReLU	5x5x96x256	1	2
27x27x256	-	-	-
Max Pool	3x3x256	2	0
13x13x256	-	-	-
Conv3+ReLU	3x3x256x384	1	1
13x13x384	-	-	-
Conv4+ReLU	3x3x384x384	1	1
13x13x384	-	-	-
Conv5+ReLU	3x3x384x256	1	1
13x13x256	-	-	-
Max Pool	3x3x256	2	0
6x6x256	-	-	-
FC6+ReLU	1x1x4096	-	-
4096	-	-	-
FC7+ReLU	1x1x4096	-	-
4096	-	-	-
FC8+ReLU	1x1x4096	-	-
1000 *classes*	-	-	-

References - Sites

1] http://neuralnetworksanddeeplearning.com/

2] http://cs231n.github.io/convolutional-networks/

3] https://anhvnn.wordpress.com/

4] http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

5] https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

6] https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637

7] http://www.image-net.org/