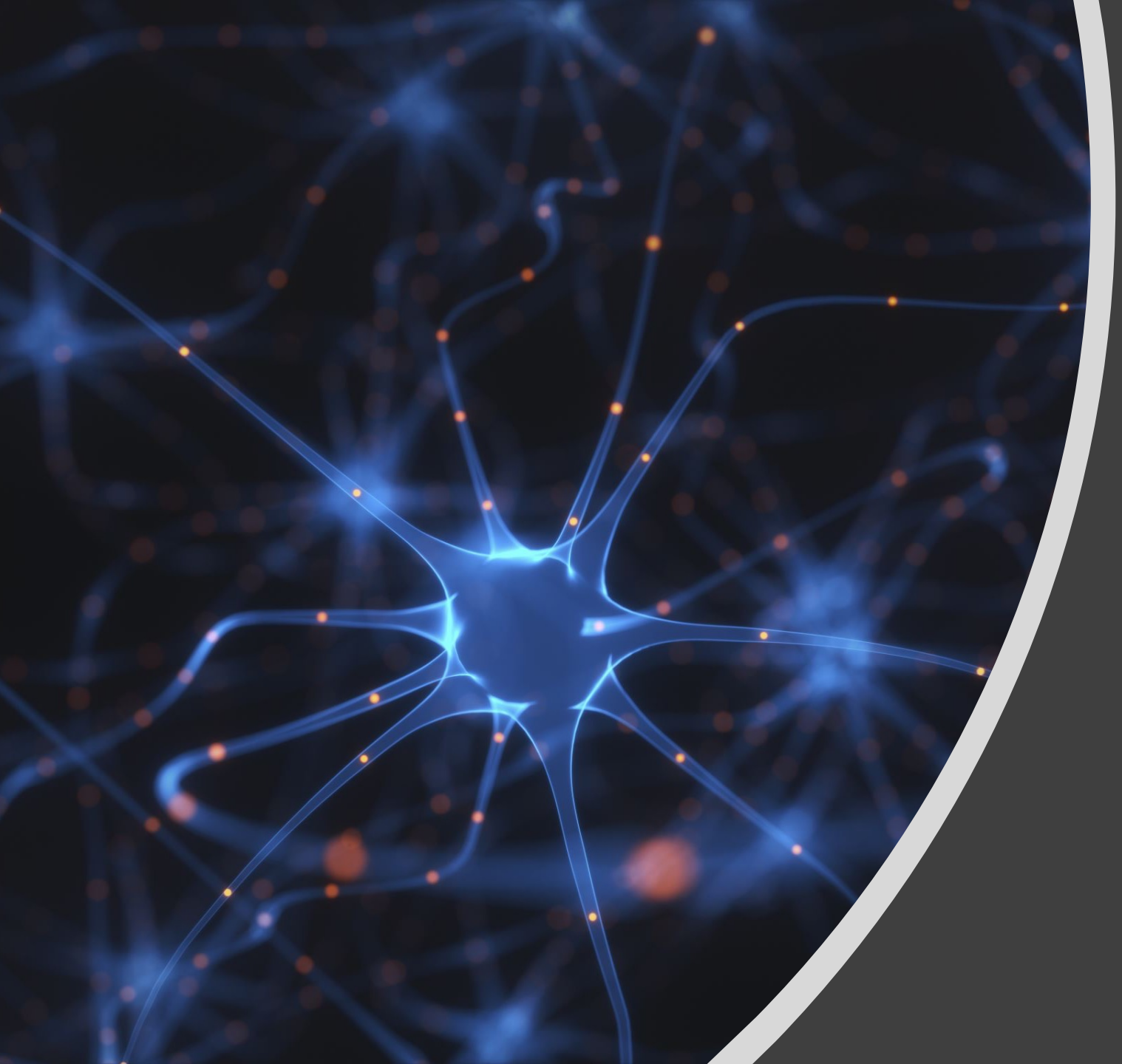
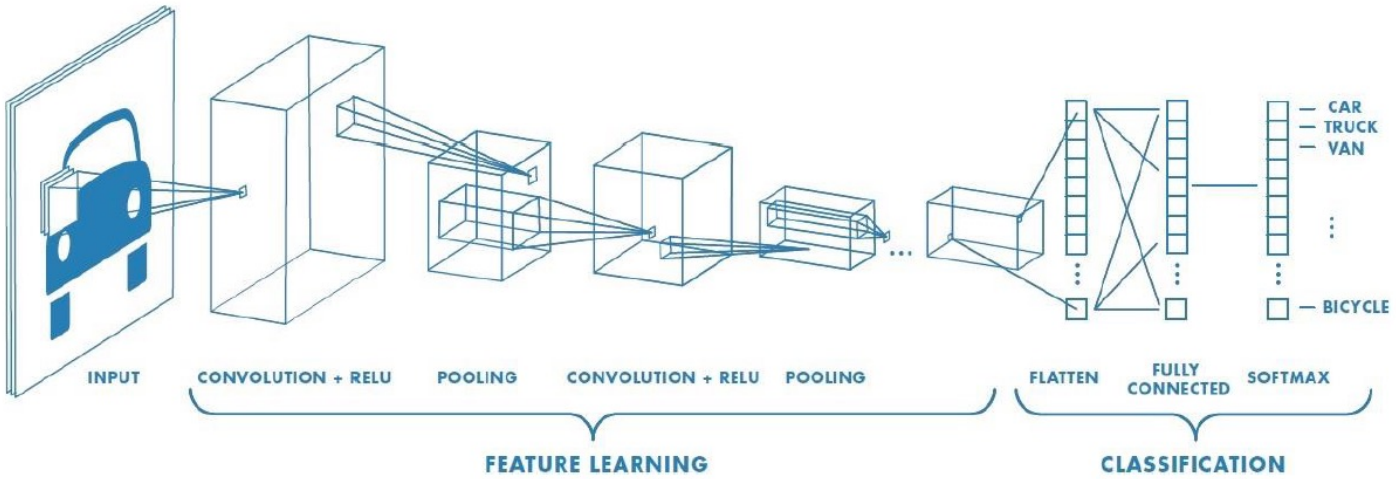


Deep Neural Networks

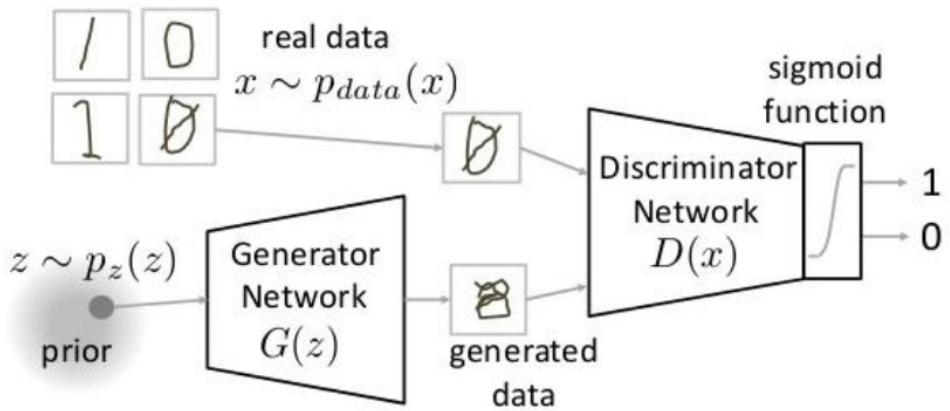
V. Megalooikonomou

Deep Neural Networks: Various Architectures

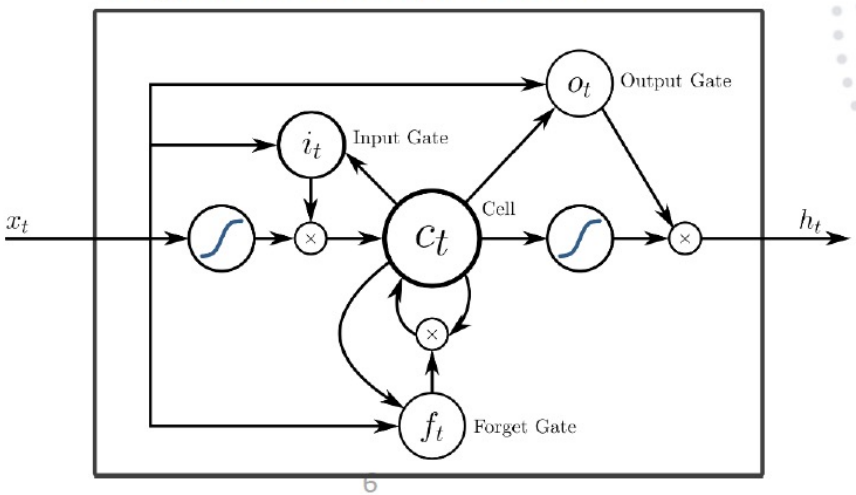
Convolutional Neural Networks



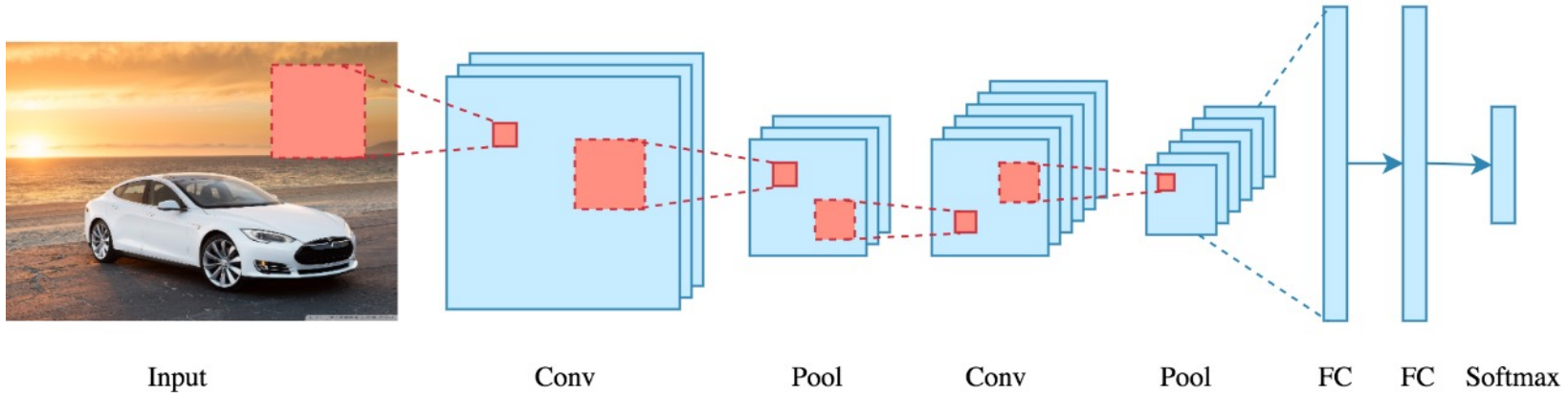
Generative (& Adversarial) Neural Networks



Long Short Memory Networks



Αρχιτεκτονική ενός CNN



Συνέλιξη (convolution)

- Συνήθως εφαρμόζεται στα δεδομένα εισόδου χρησιμοποιώντας ένα φίλτρο συνέλιξης (*convolutional filter*) για την παραγωγή ενός χάρτη χαρακτηριστικών (*feature map*).

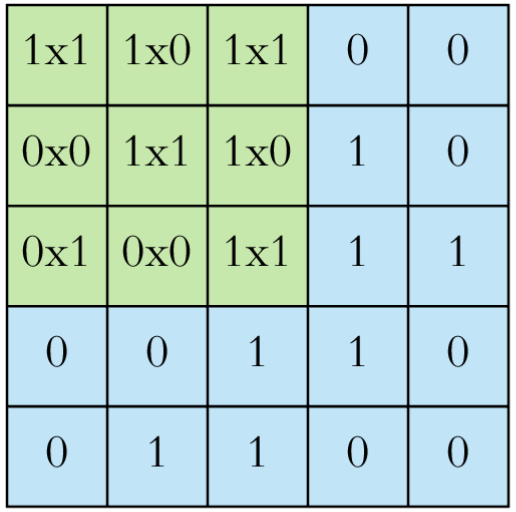
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

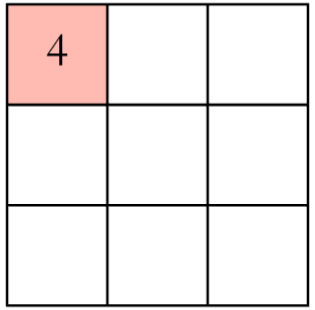
1	0	1
0	1	0
1	0	1

Filter / Kernel

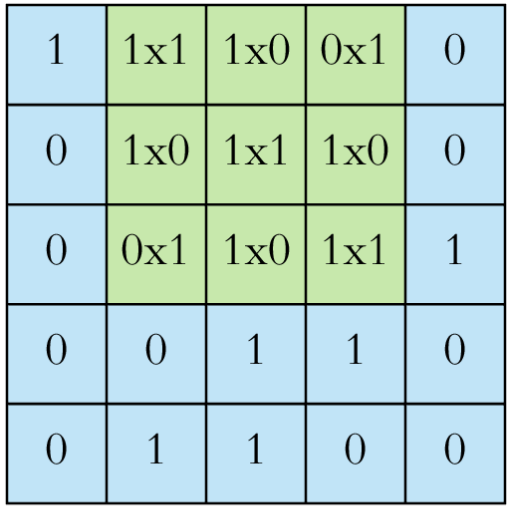
Συνέλιξη (convolution)



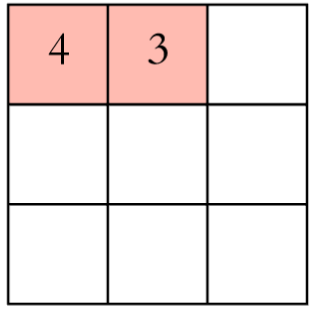
Input x Filter



Feature Map

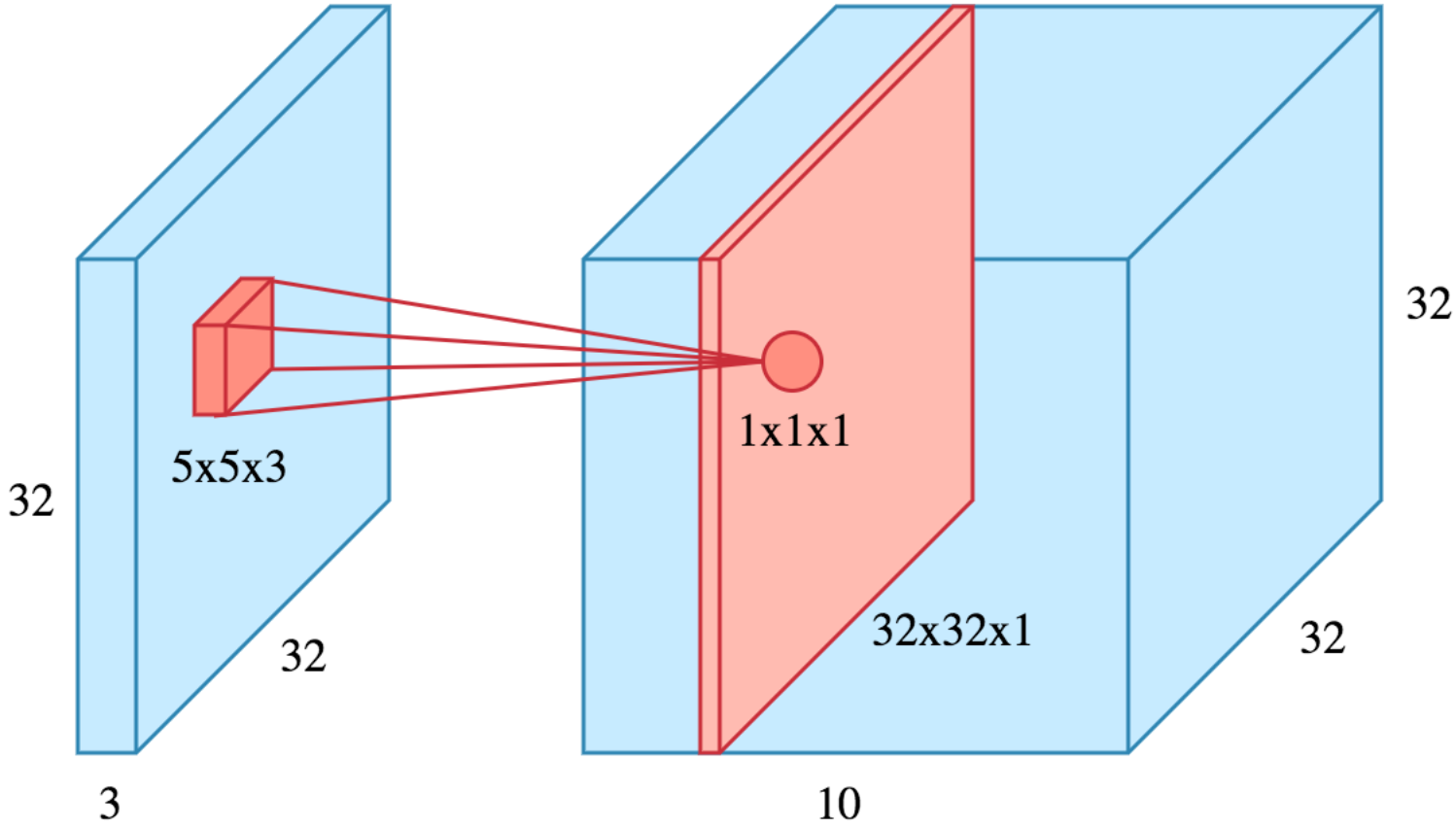


Input x Filter



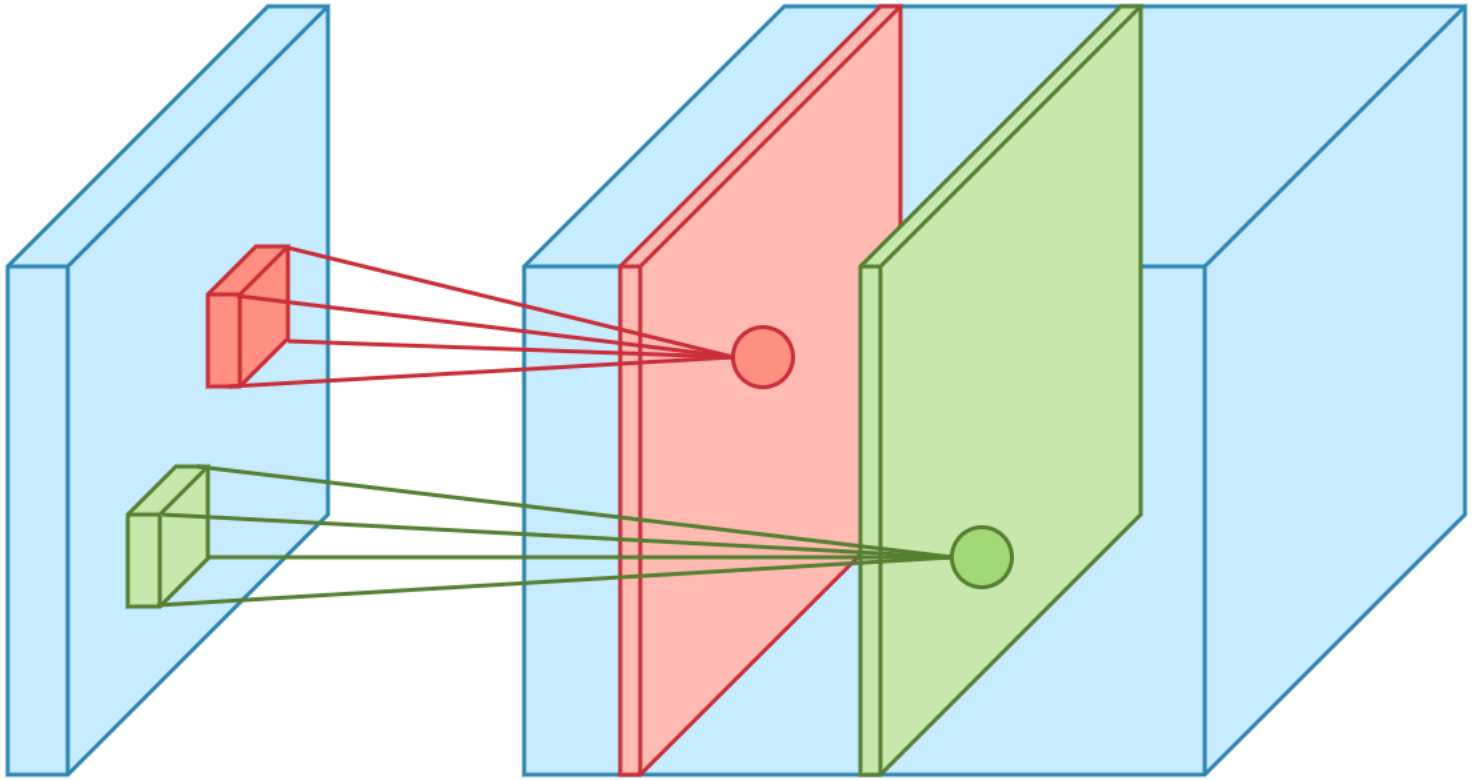
Feature Map

Συνέλιξη (convolution)



Οπτικοποιήση της συνέλιξης σε 3D εικόνα διαστάσεων 32*x32x1*

Συνέλιξη (convolution)



Απεικονίζονται 2 *feature maps κατά το μήκος της διάστασης του βάθους*

ReLU (Rectified linear activation function)

- Παραδοσιακές συναρτήσεις μη γραμμικής ενεργοποίησης
  - συναρτήσεις σιγμοειδούς (sigmoid) και
  - υπερβολικής εφαπτομένης ενεργοποίησης (hyperbolic tangent)
- Γίνονται κορεσμένες: οι μεγάλες τιμές κουμπώνουν στο 1,0 και οι μικρές τιμές στο -1 ή 0 για το tanh και το σιγμοειδές αντίστοιχα.
- Επιπλέον, οι συναρτήσεις είναι πραγματικά ευαίσθητες μόνο σε αλλαγές γύρω από το μέσο σημείο εισόδου τους, όπως 0,5 για σιγμοειδές και 0,0 για tanh.

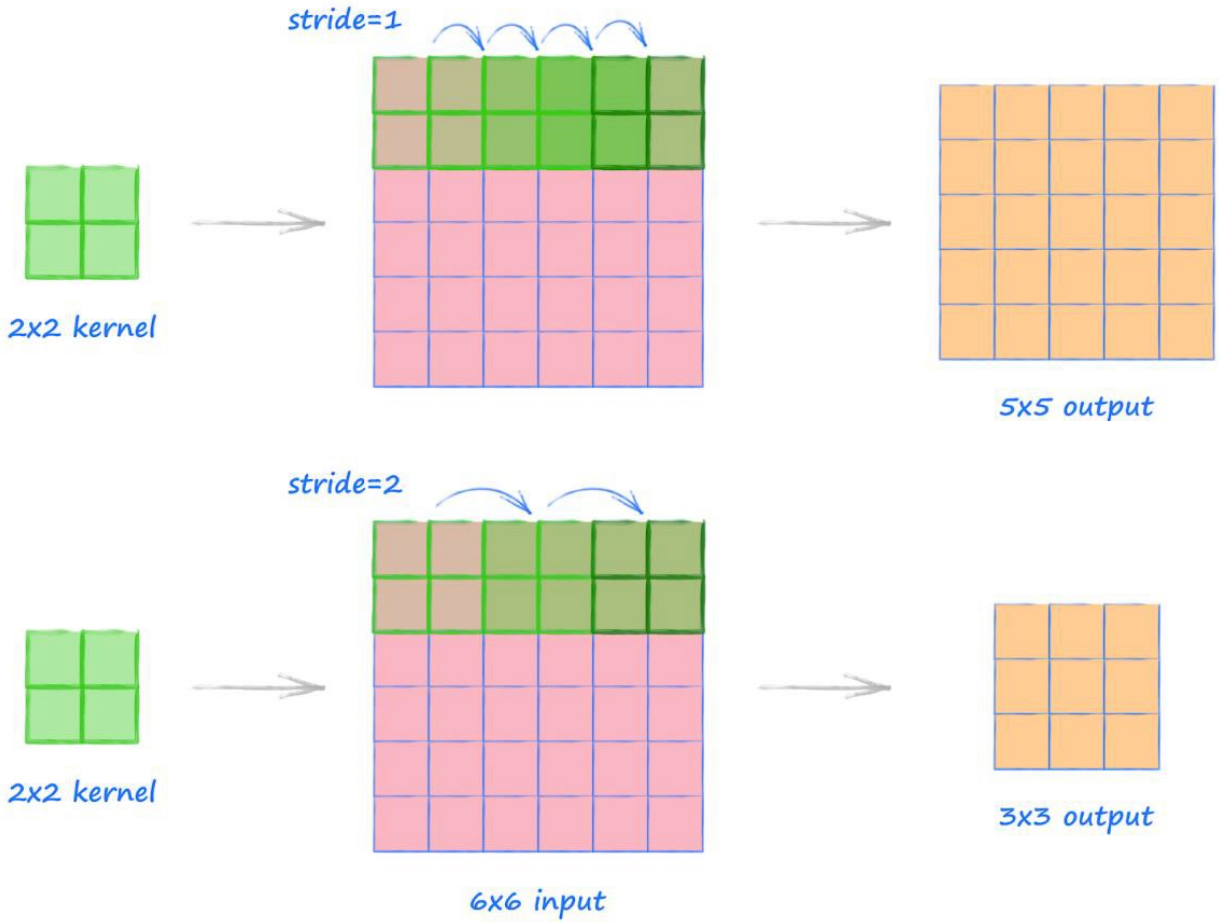
- Για να χρησιμοποιηθεί η stochastic gradient descent με αντίστροφη διάδοση σφαλμάτων για την εκπαίδευση των DNNs απαιτείται συνάρτηση ενεργοποίησης που μοιάζει και λειτουργεί ως γραμμική συνάρτηση αλλά στην πραγματικότητα είναι μη γραμμική συνάρτηση που επιτρέπει την εκμάθηση πολύπλοκων σχέσεων στα δεδομένα

ReLU (Rectified linear activation function)

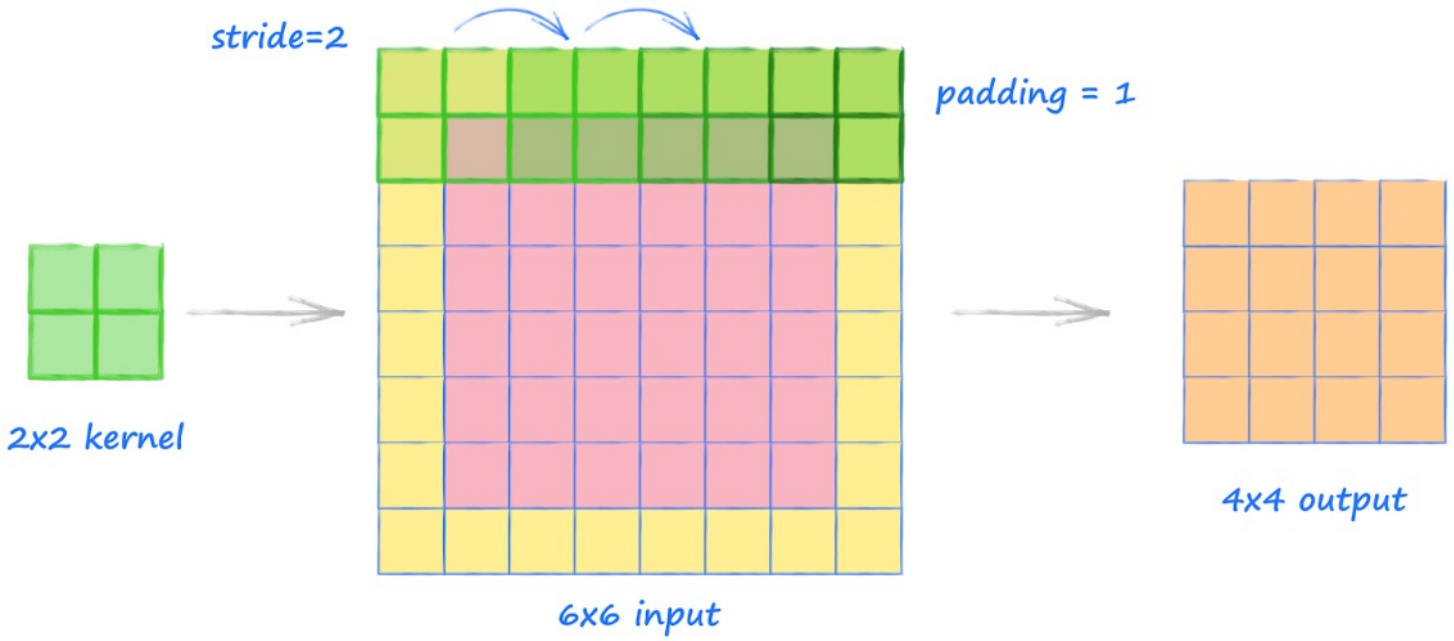
- Μια τμηματικά γραμμική συνάρτηση ενεργοποίησης
  $f(x) = max(0,x)$

- Εξάγει απευθείας την είσοδο αν είναι θετική, διαφορετικά μηδέν.
- Είναι η προεπιλεγμένη συνάρτηση ενεργοποίησης για πολλούς τύπους NNs
- Ένα μοντέλο που την χρησιμοποιεί είναι πιο εύκολο στην εκπαίδευση και συχνά επιτυγχάνει καλύτερη απόδοση

Stride & padding



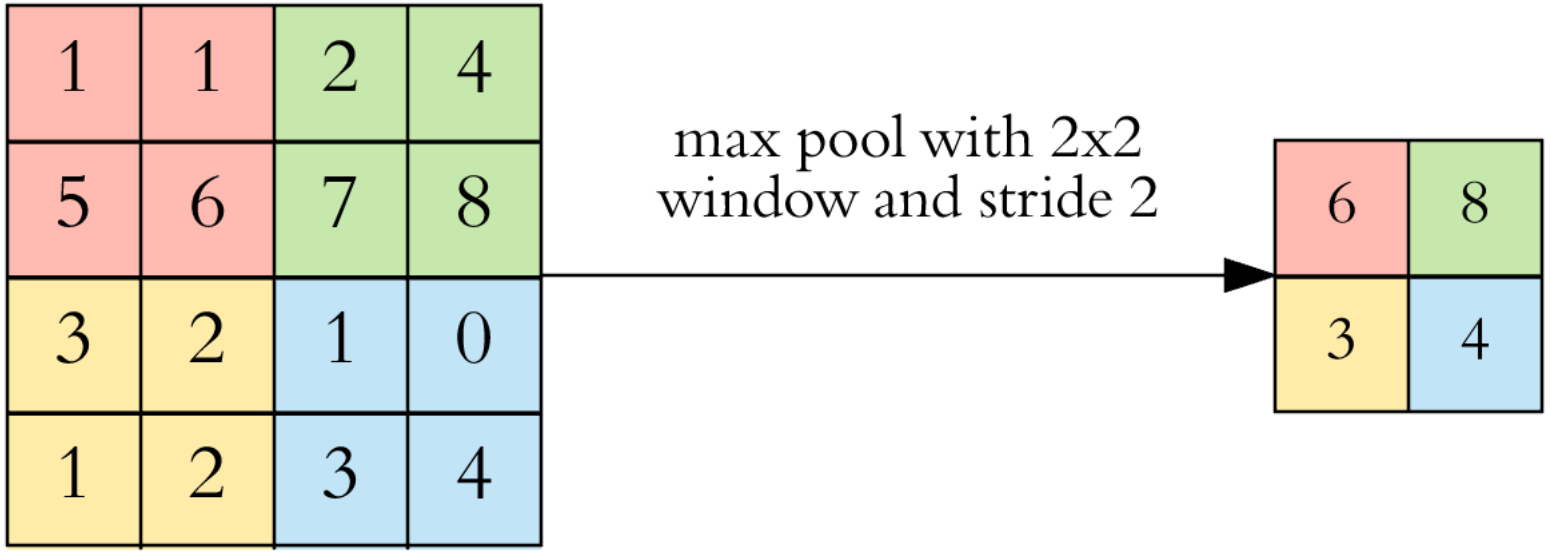
Stride & padding



Αν θέλουμε να διατηρήσουμε την ίδια διαστατικότητα, μπορούμε να χρησιμοποιήσουμε padding για να περιβάλλουμε την είσοδο με μηδενικά

Συγκέντρωση - Pooling

- Μείωση του αριθμού των παραμέτρων
- Συντόμευση του χρόνου εκπαίδευσης
- Καταπολέμηση του overfitting
- Πιο συνηθισμένος τύπος συγκέντρωσης: max pooling



Υπερ-παράμετροι

- **Μέγεθος φίλτρου:** συνήθως χρησιμοποιούμε φίλτρα 3x3, αλλά χρησιμοποιούνται επίσης 5x5 ή 7x7 ανάλογα με την εφαρμογή

- **Αριθμός φίλτρων:** δύναμη του δύο οπουδήποτε μεταξύ 32 και 1024.
  - Η χρήση περισσότερων φίλτρων έχει ως αποτέλεσμα ένα πιο ισχυρό μοντέλο, αλλά κινδυνεύουμε με *overfitting* λόγω του αυξημένου αριθμού παραμέτρων.
  - Συνήθως ξεκινάμε με μικρό αριθμό φίλτρων στα αρχικά στρώματα και αυξάνουμε σταδιακά τον αριθμό καθώς προχωράμε βαθύτερα στο δίκτυο.

- **Stride:** προεπιλεγμένη τιμή 1

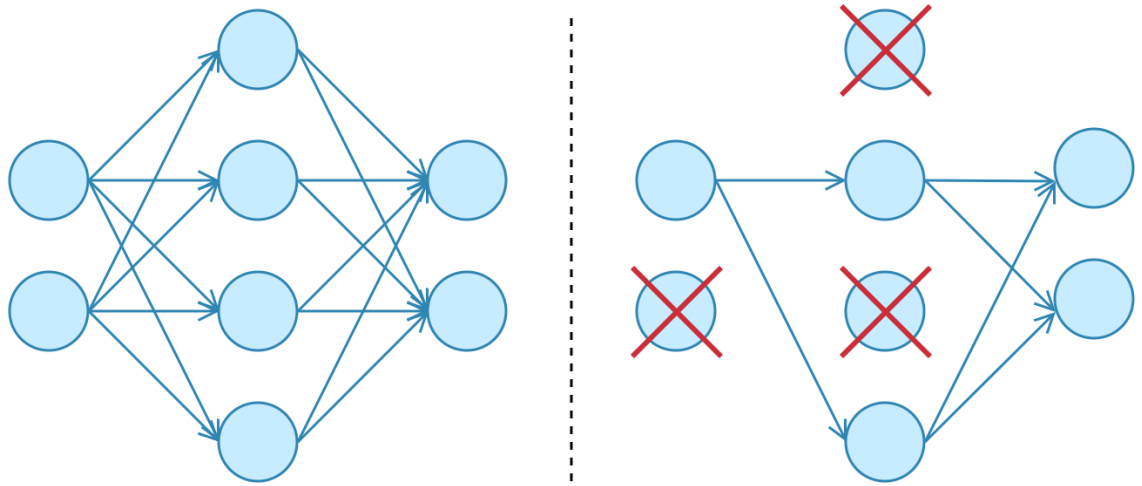
- **Padding:** συνήθως χρησιμοποιούμε padding

Fully Connected (FC) layers

- Μετά τα επίπεδα συνέλιξης & συγκέντρωσης προσθέτουμε μερικά FC επίπεδα για να ολοκληρώσουμε την αρχιτεκτονική του CNN.
- Πρόκειται για την ίδια αρχιτεκτονική πλήρως συνδεδεμένου ANN
- Η έξοδος τόσο των επιπέδων συνέλιξης όσο και των επιπέδων συγκέντρωσης είναι τρισδιάστατοι όγκοι, αλλά ένα πλήρως συνδεδεμένο επίπεδο αναμένει ένα διάνυσμα αριθμών 1D:
  - Εξισώνουμε την έξοδο του τελευταίου στρώματος συγκέντρωσης σε ένα διάνυσμα
  - Αυτό γίνεται η είσοδος στο πλήρως συνδεδεμένο στρώμα.
  - Η ισοπέδωση είναι απλώς η διευθέτηση του τρισδιάστατου όγκου αριθμών σε ένα διάνυσμα 1D

Dropout

- Για την αποφυγή της υπερβολικής προσαρμογής (*overfitting*)
- Κατά τη διάρκεια της εκπαίδευσης, σε κάθε επανάληψη, ένας νευρώνας "πέφτει" προσωρινά ή απενεργοποιείται με πιθανότητα p
  - Η υπερπαράμετρος p ονομάζεται *ρυθμός αποχώρησης (*είναι συνήθως ένας αριθμός γύρω στο 0,5 που αντιστοιχεί στο 50% των νευρώνων που αποχωρούν

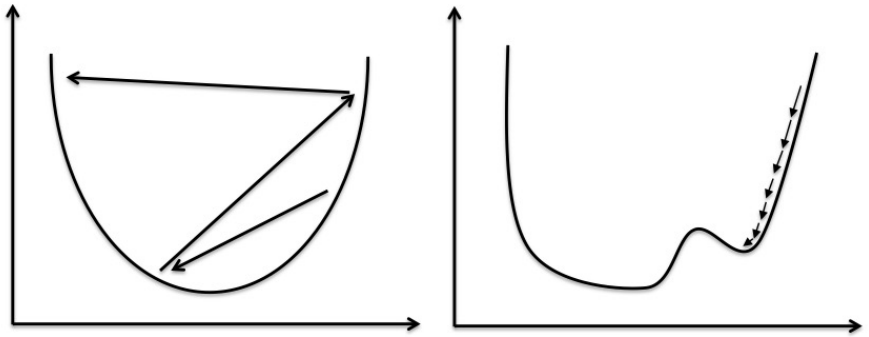


Εκπαίδευση - Training

- Το CNN εκπαιδεύεται με τον ίδιο τρόπο όπως και το ANN: backpropagation με gradient descent.
- Οι εξισώσεις προκύπτουν με παρόμοια διαδικασία όπως στα πολυδιάστατα Perceptron (ANNs).
- Στην εκπαίδευση του CNN έχουμε και πάλι την εμπρόσθια ενεργοποίηση (*Forward activation*) και την οπισθοδρομική διάδοση του σφάλματος (*Error backpropagation*).

Gradient Descent

- Ελαχιστοποίηση της loss function
- Χρήση της παραγώγου της loss function – ακριβή πράξη
- Stochastic gradient descent (SGD):
  - επιλέγονται τυχαία δείγματα από το training set ώστε να υπολογιστεί μια προσέγγιση της παραγώγου

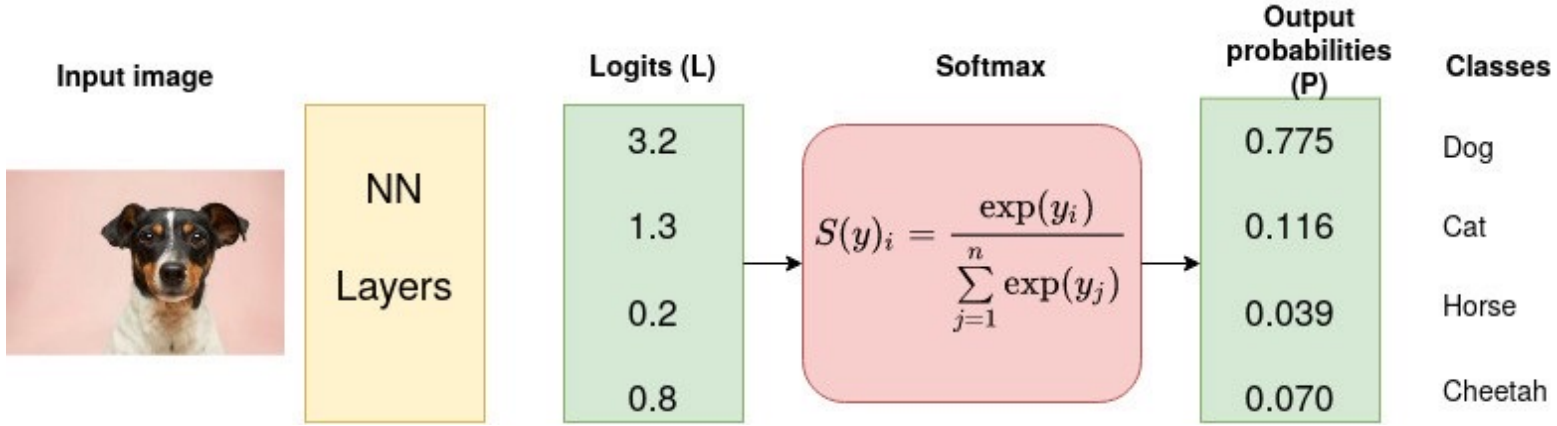


Αριστερά: για μεγάλο α (learning rate) γίνονται μεγάλα άλματα ξεπερνώντας το ολικό ελάχιστο

Δεξιά: μικρότερες διορθώσεις στα βάρη οδηγούμενοι σε ελάχιστο

Softmax Activation Function

- Η softmax function χρησιμοποιείται ως η τελευταία activation function ενός NN:
  - για την κανονικοποίηση της έξοδου του δικτύου σε μία κατανομή πιθανότητας πάνω στις προβλεπόμενες κατηγορίες εξόδου (output classes)
- Πριν την εφαρμογή της softmax, κάποια μέρη του διανύσματος μπορεί να είναι αρνητικά ή μεγαλύτερα του 1, και μπορεί να μην αθροίζουν σε 1.
- Μετά την εφαρμογή της, κάθε αριθμός θα είναι στο (0,1), και θα αθροίζουν σε 1, έτσι ώστε να μπορούν να ληφθούν ως πιθανότητες



Softmax Activation Function

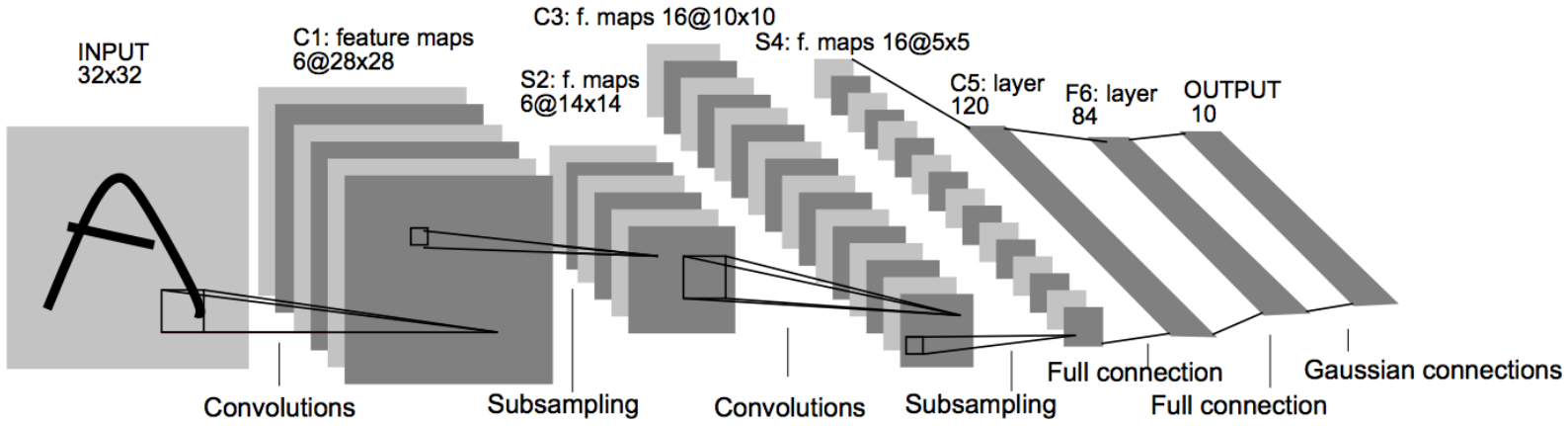
$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}$$

where,

y	is an input vector to a softmax function, S. It consist of n elements for n classes (possible outcomes)
y_i	the i-th element of the input vector. It can take any value between -inf and +inf
$exp(y_i)$	standard exponential function applied on y_i. The result is a small value (close to 0 but never 0) if $y_i < 0$ and a large value if y_i is large. eg • $\exp(55) = 7.69e{+}23$ (A very large value) • $\exp(-55) = 1.30e{-}24$ (A very small value close to 0) **Note**: $\exp(*)$ is just $\mathbf{e}^*$ where $\mathbf{e} = 2.718$, the Euler's number.
$\sum_{j=1}^{n} \exp(y_j)$	A normalization term. It ensures that the values of output vector $S(y)_i$ sums to 1 for i-th class and each of them and each of them is in the range 0 and 1 which makes up a valid probability distribution.
n	Number of classes (possible outcomes)

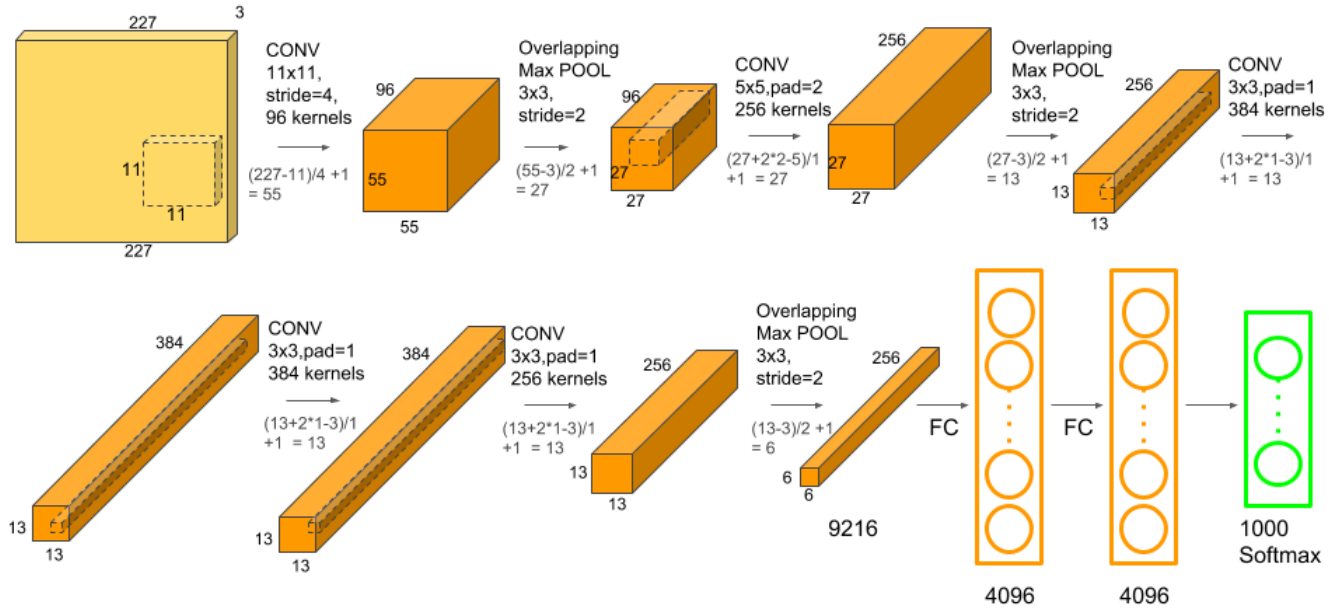
CNN μοντέλα: LeNet-5

- LeNet (LeCun, Bottou, Bengio, & Haffner, 1998)



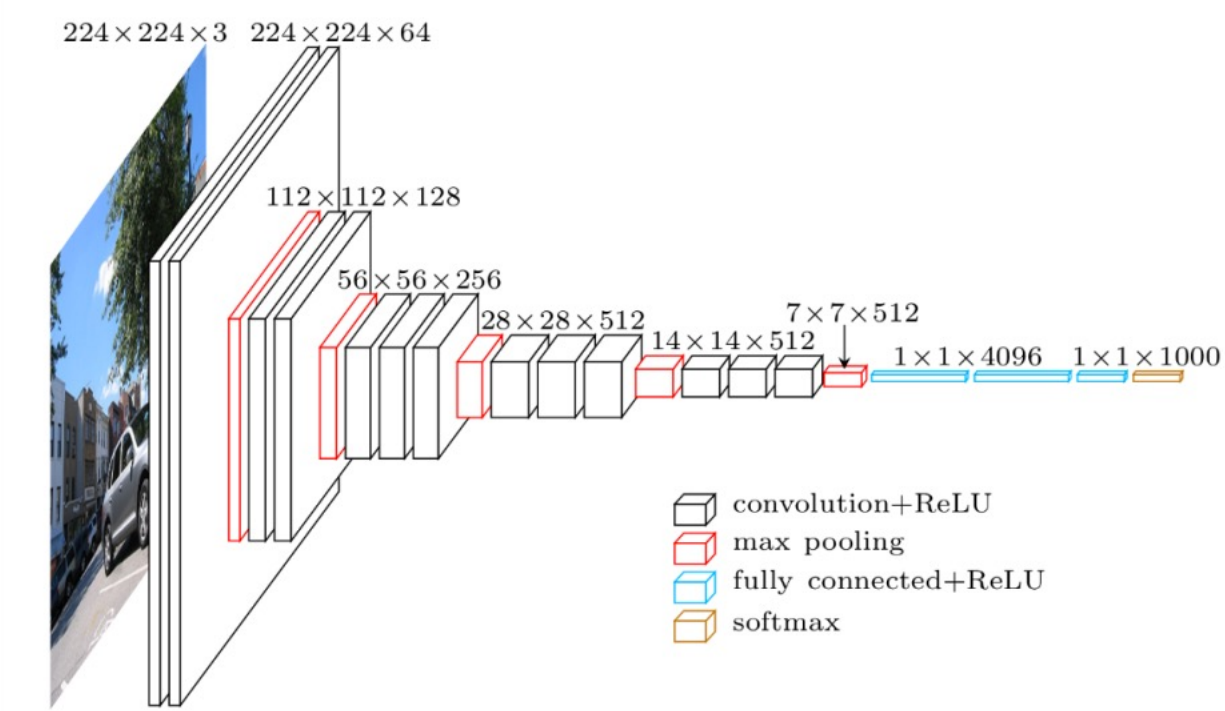
AlexNet

- AlexNet (Krizhevsky, Sutskever, & Hinton, 2012)



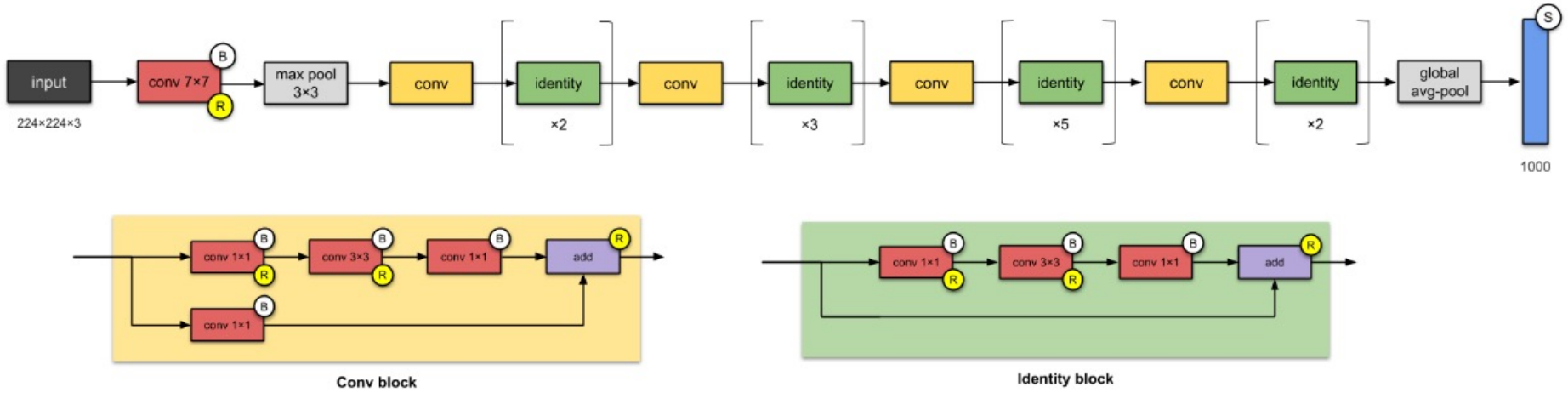
VGG

- VGG (Simonyan & Zisserman, 2015)



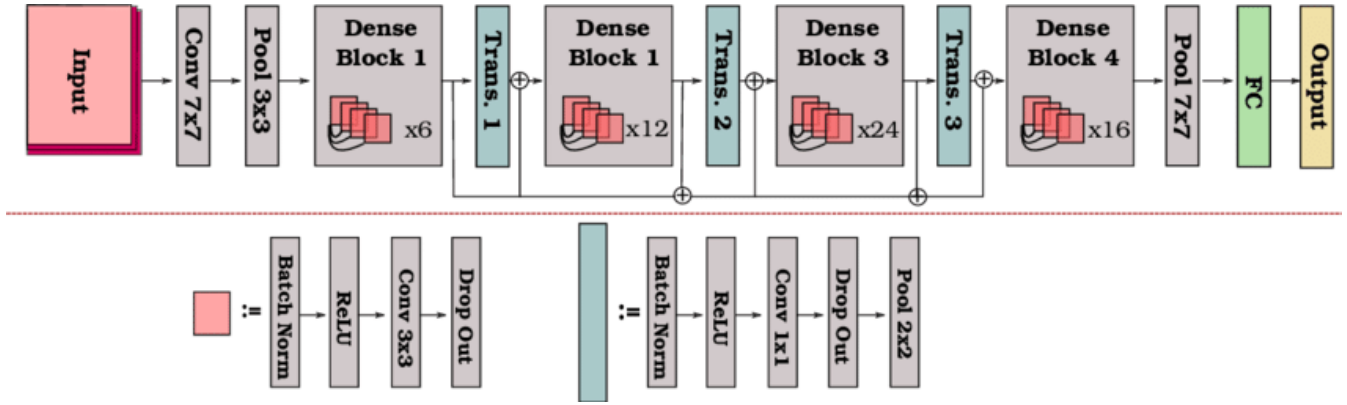
ResNet – Residual Neural Network

- K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*. 2016



DenseNet: Densely Connected Convolutional Networks

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			



CNN Models

Architectures	Total Parameters
AlexNet	62,378,344 [1]
VGG-11	138,357,544 [2]
ResNet-50	25,636,712 [3]
DenseNet-121	7.033.282[4]
LeNet-5	60,000 [5]

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems* (2012).

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." (2015).

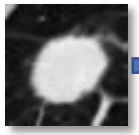
[3] K. He, X. Zhang, S. Ren, and J. Sun, "*Deep Residual Learning for Image Recognition*." (2016).

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243

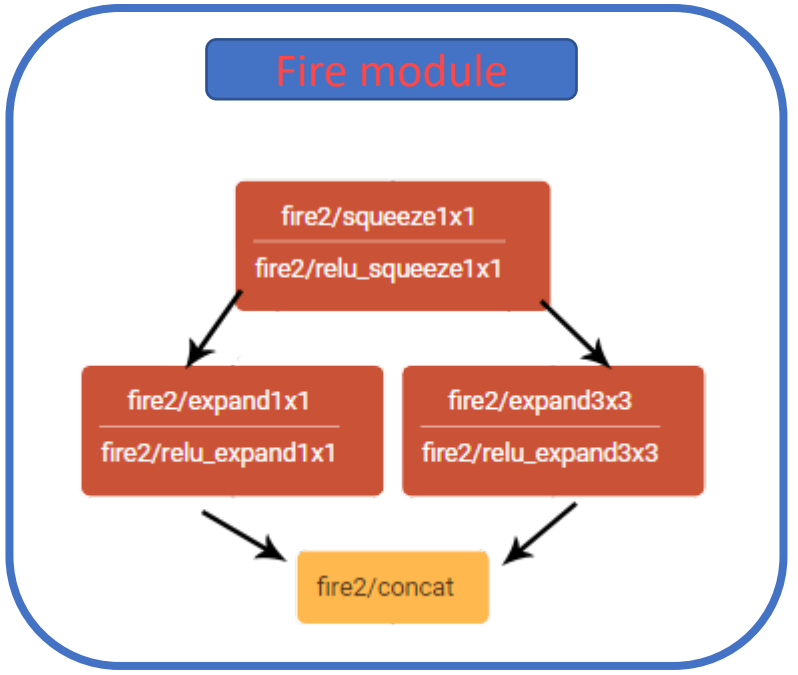
[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278– 2324 (1998).

SqueezeNet

227x227x3



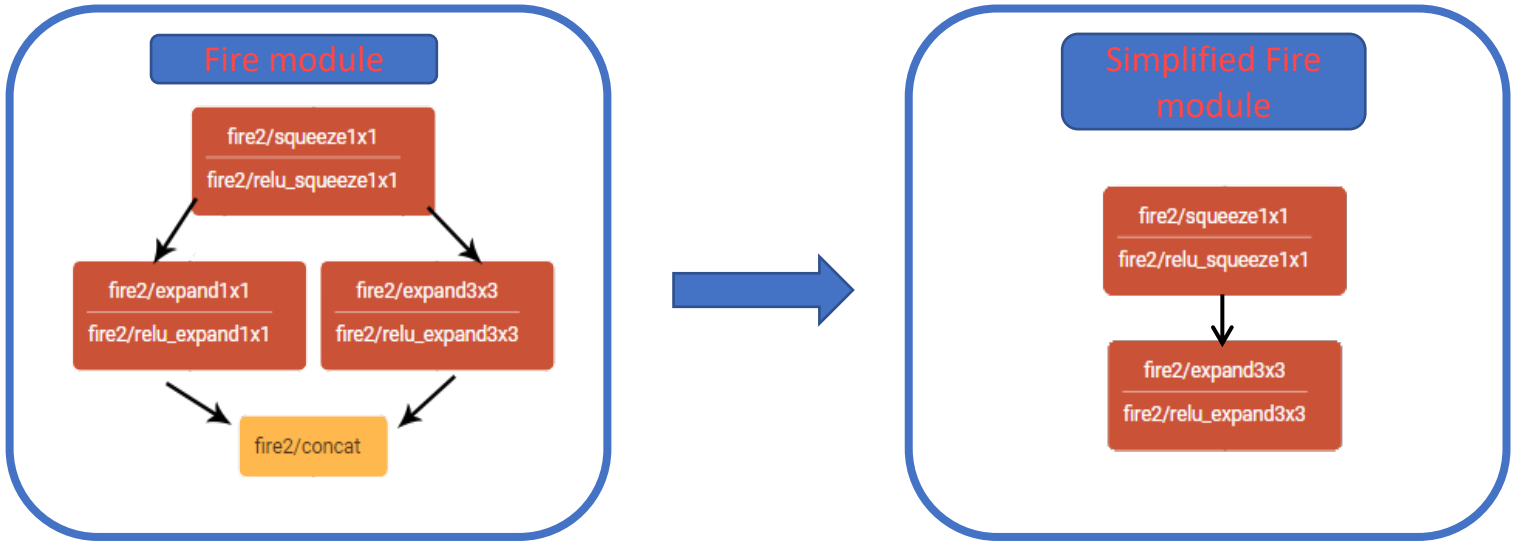
50x50x1



	SqueezeNet	SqueezeNet
Total Params	1,248,424	722,376

SqueezeNet Modification

Squeeze Ratio: 0.125



Squeeze Ratio: 0.25 (V1)

Squeeze Ratio: 0.50 (V2)

SqueezeNodule-Net V1: Decreasing # parameters while attempting to preserve accuracy

- Replace 3×3 filters with 1×1 filters
- Decrease the number of input channels to 3×3 filters (number of input channels) × (number of filters) × (3×3)

SqueezeNodule-Net V2: Maximizing accuracy on a limited budget of parameters

- Downsample late in the network so that convolution layers have large activation maps.

M. Tsivgoulis, T. Papastergiou, V. Megalooikonomou, Machine Learning with Applications, 2022

FN Iandola, S Han, MW Moskewicz, K Ashraf, WJ Dally, K Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size (2017)

RNN μοντέλα

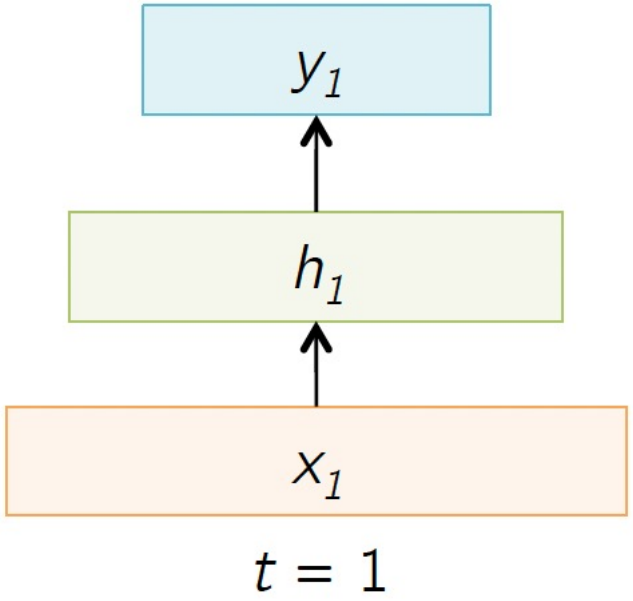
- Για προβλήματα που δεν μπορούν να μετασχηματιστούν σε fixed length inputs and outputs
- Π.χ. speech recognition ή time series prediction
- Όπου απαιτείται η αποθήκευση και χρήση context information
- Όπου είναι δύσκολο/αδύνατο να χρησιμοποιηθεί fixed content window

RNN μοντέλα

- Τα RNNs παίρνουν την προηγούμενη έξοδο ή κρυφές καταστάσεις ως εισόδους
- Η σύνθετη είσοδος τη χρονική στιγμή t έχει κάποια ιστορική πληροφορία σχετικά με τα γεγονότα που συνέβησαν τη χρονική στιγμή T < t!
- Στα RNNs οι ενδιάμεσες τιμές τους (κατάσταση) αποθηκεύουν πληροφορία σχετικά με τις παρελθοντικές εισόδους για μια χρονική στιγμή που δεν είναι καθορισμένη εκ των προτέρων

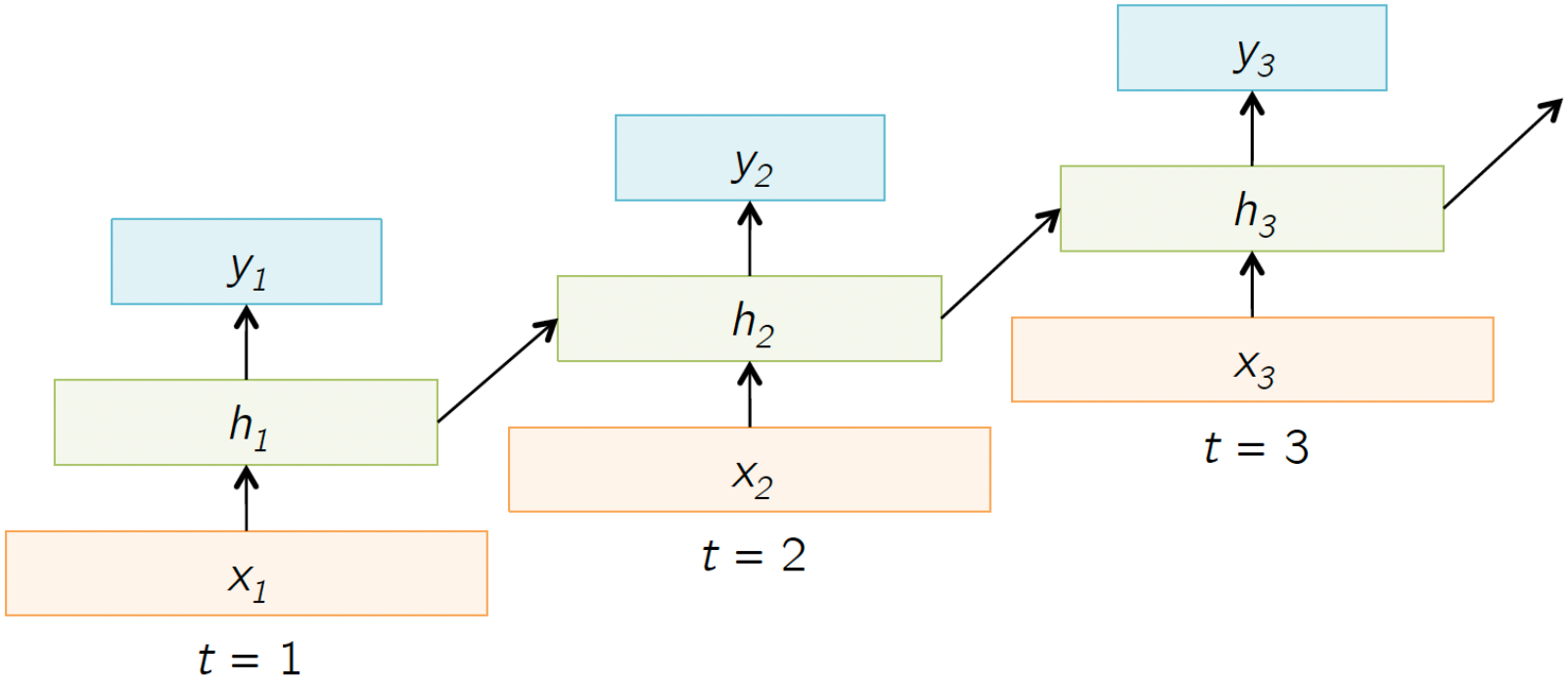
RNN μοντέλα

Παράδειγμα feed forward network (FFN):



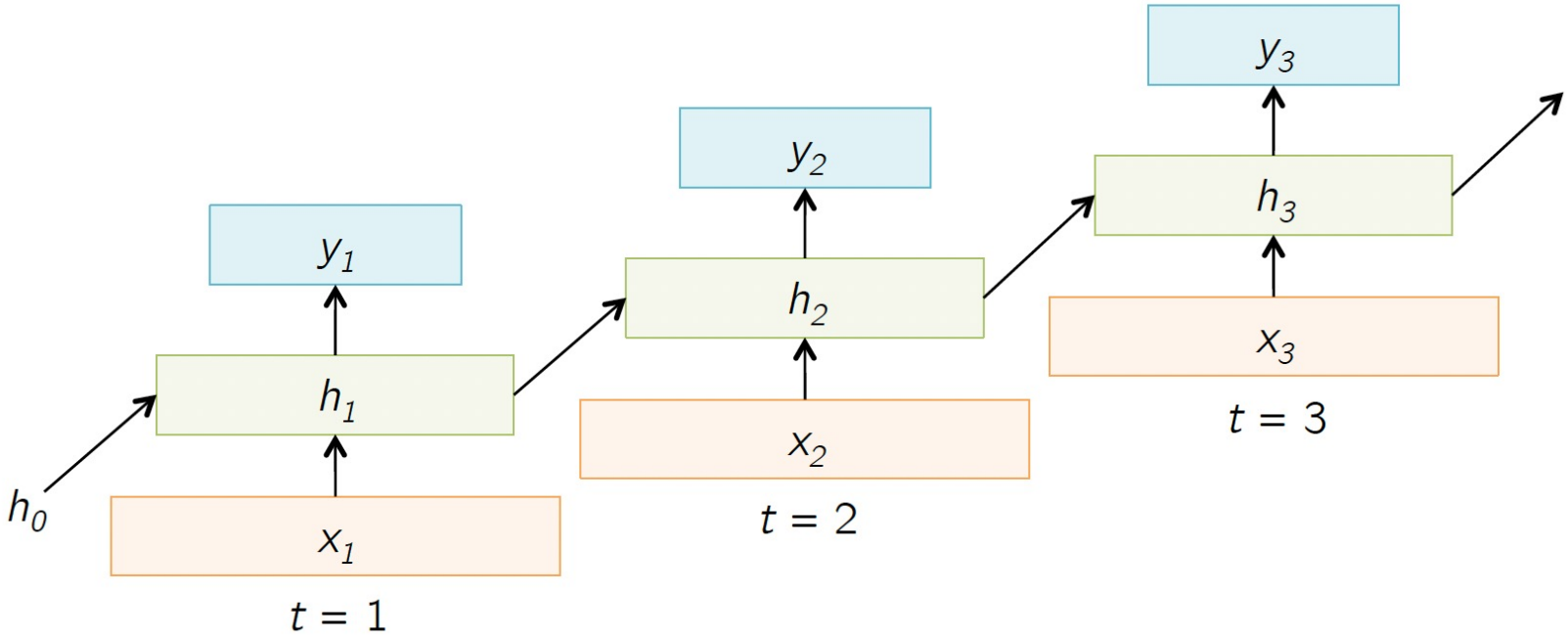
RNN μοντέλα

Παράδειγμα RNN:



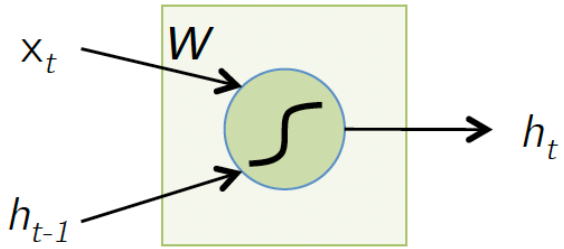
RNN μοντέλα

Παράδειγμα RNN:



RNN μοντέλα

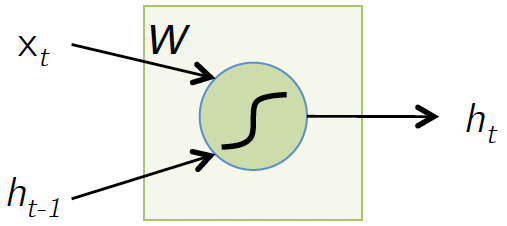
Απλό RNN cell:



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

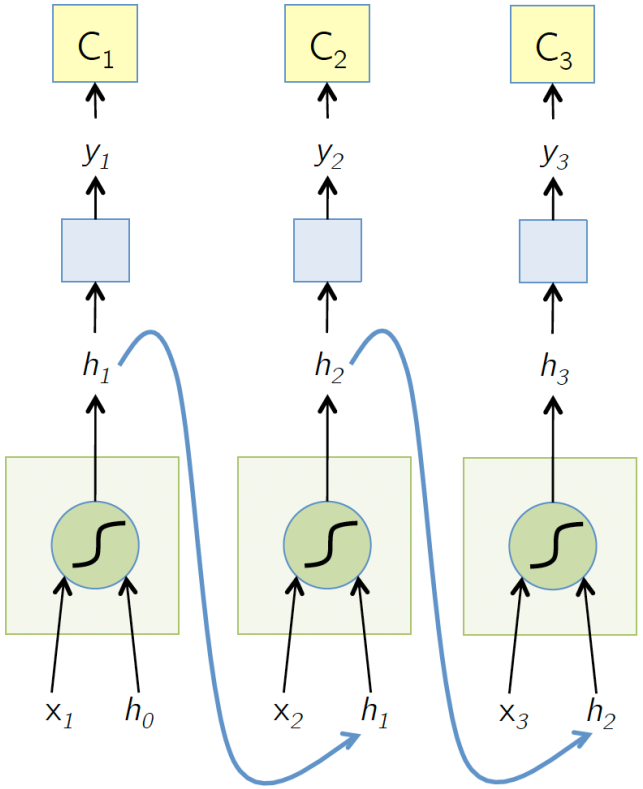
RNN μοντέλα

Απλό RNN cell:



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

Απλό RNN forward:



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

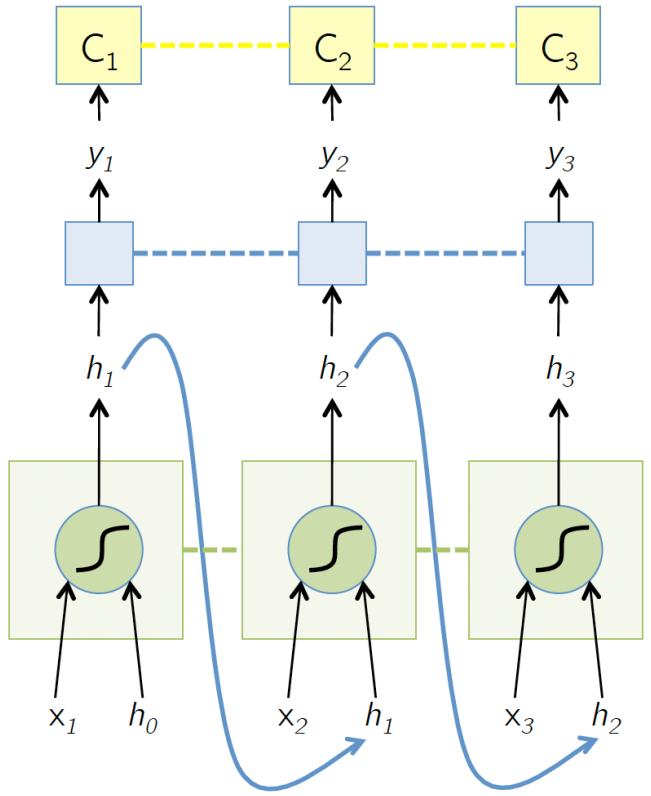
$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

RNN μοντέλα

Απλό RNN forward:

- τα βάρη μοιράζονται με την πάροδο του χρόνου!
- Ουσιαστικά, δημιουργούνται αντίγραφα του κυττάρου RNN με την πάροδο του χρόνου (ξετύλιγμα/ξεδίπλωμα), με διαφορετικές εισόδους σε διαφορετικές χρονικές στιγμές (βήματα)



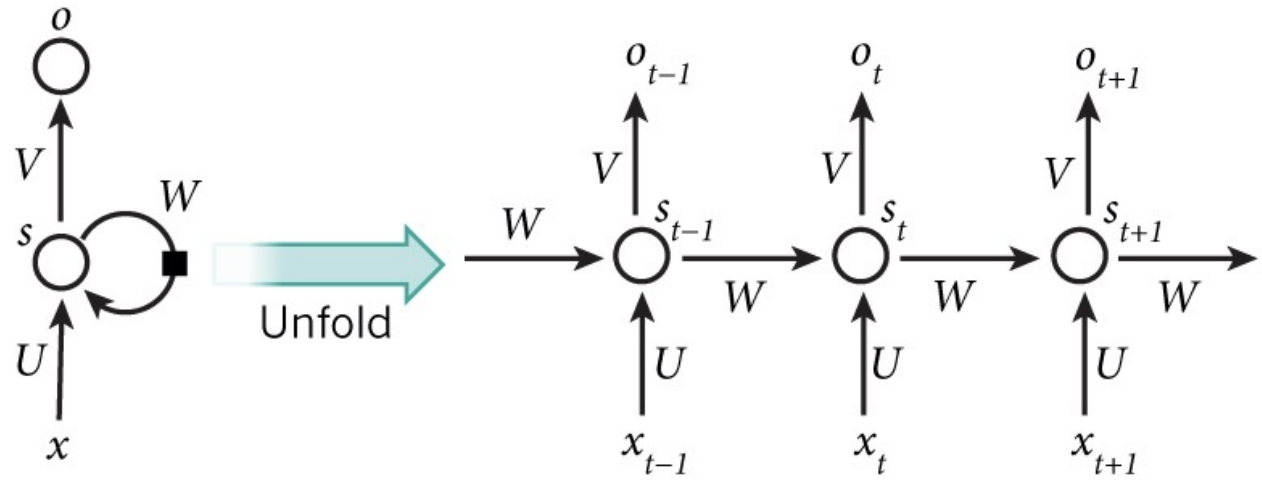
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

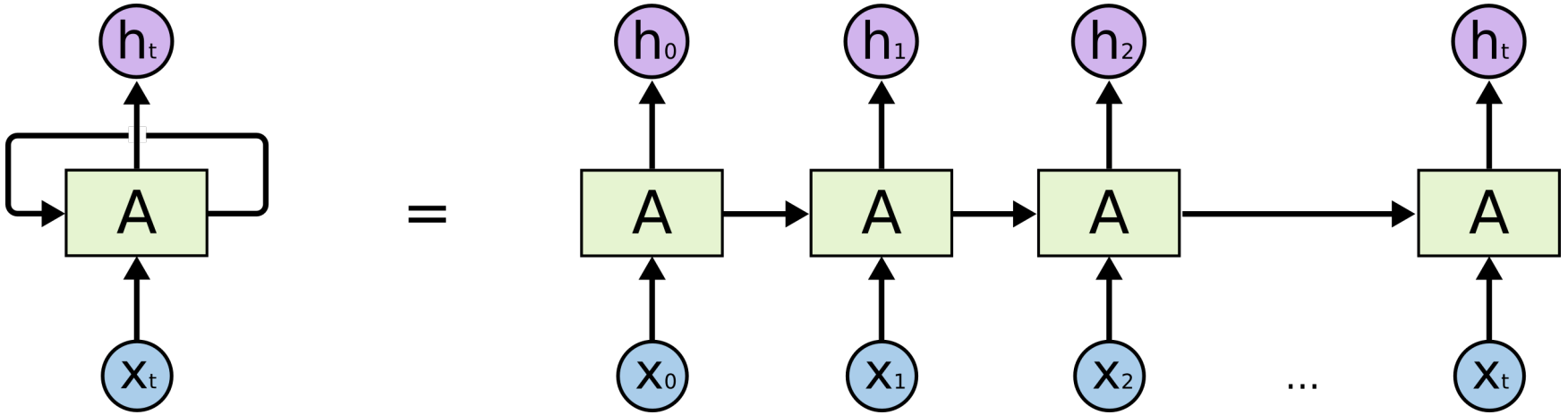
$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

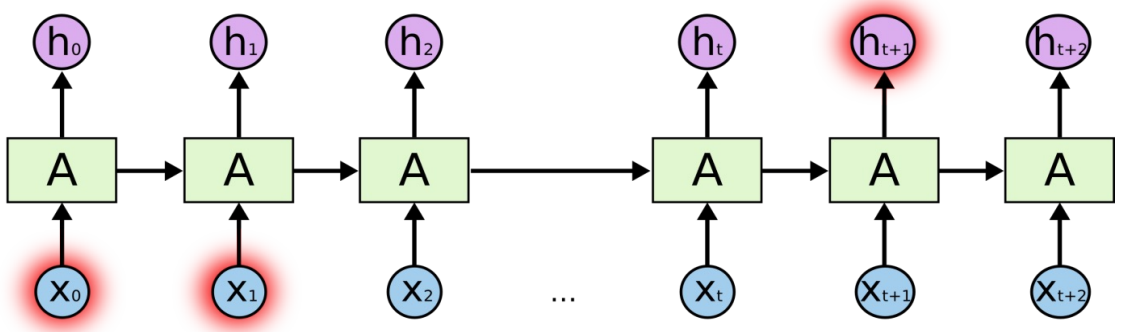
RNN μοντέλα


o
V
W
s
U
x
Unfold
o_{t-1}
o_t
o_{t+1}
s_{t-1}
s_t
s_{t+1}
x_{t-1}
x_t
x_{t+1}


h_t
A
x_t
=
h_0
h_1
h_2
h_t
A
A
A
A
x_0
x_1
x_2
...
x_t

RNN μοντέλα: προβλήματα

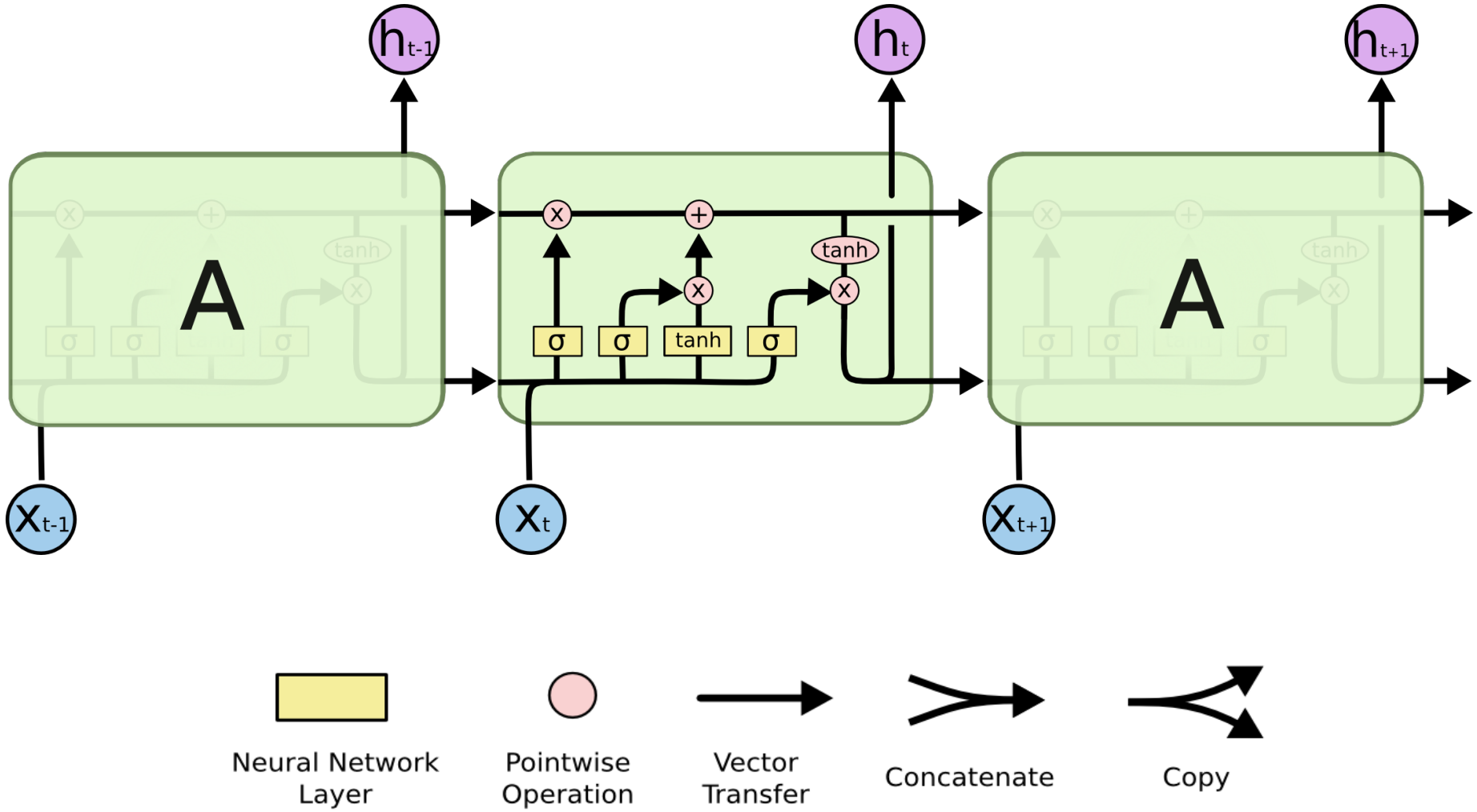
- Πρόβλημα του vanishing/exploding gradient:
  - Τα backpropagated σφάλματα σε κάθε επίπεδο πολλαπλασιάζονται, με αποτέλεσμα εκθετική μείωση (αν η παράγωγος είναι μικρή) ή αύξηση (αν η παράγωγος είναι μεγάλη).
  - Καθιστά πολύ δύσκολη την εκπαίδευση βαθιών δικτύων ή απλών επαναλαμβανόμενων δικτύων για πολλά χρονικά βήματα.
- Πρόβλημα των long distance dependencies:
  - Πολύ δύσκολο να εκπαιδεύσουμε τα RNN να διατηρούν πληροφορίες για πολλά χρονικά βήματα.
  - Πολύ δύσκολη η εκμάθηση RNN που χειρίζονται εξαρτήσεις σε μεγάλες αποστάσεις, όπως η συμφωνία υποκειμένου-ρήματος.



LSTM (Long Short Term Memory) μοντέλα

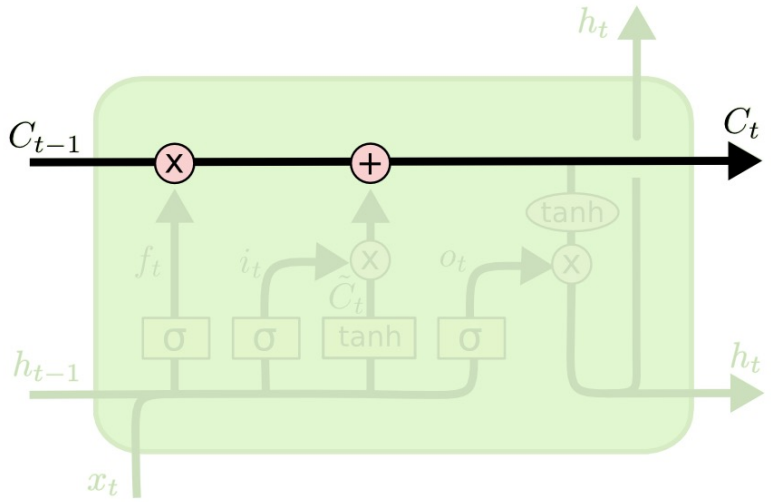
- LSTM μοντέλα, προσθέτουν επιπλέον gating units σε κάθε memory cell:
    - Forget gate
    - Input gate
    - Output gate
- Αποτρέπουν το πρόβλημα του vanishing/exploding gradient και επιτρέπουν στο δίκτυο να διατηρεί πληροφορίες κατάστασης για μεγαλύτερες χρονικές περιόδους

LSTM (Long Short Term Memory) μοντέλα: Αρχιτεκτονική



LSTM (Long Short Term Memory) μοντέλα: Cell State

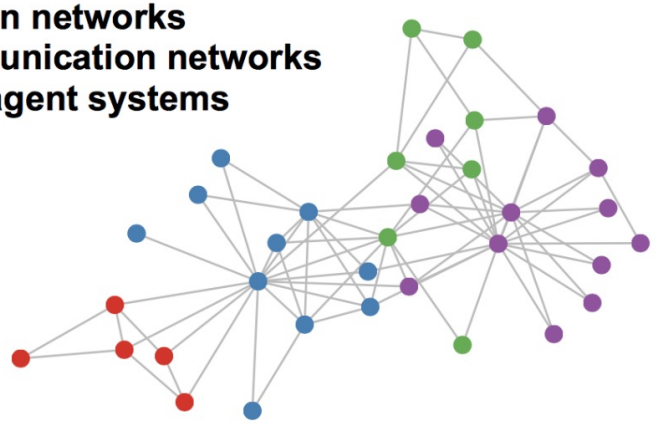
- Διατηρεί ένα διάνυσμα C_t που έχει την ίδια διάσταση με την κρυφή κατάσταση, h_t
- Πληροφορίες μπορούν να προστεθούν ή να διαγραφούν από αυτό το διάνυσμα κατάστασης μέσω των πυλών forget και input.

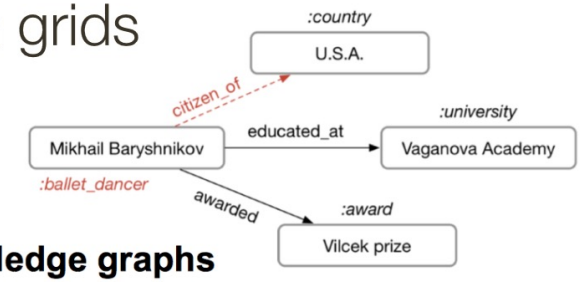


Graph structured data

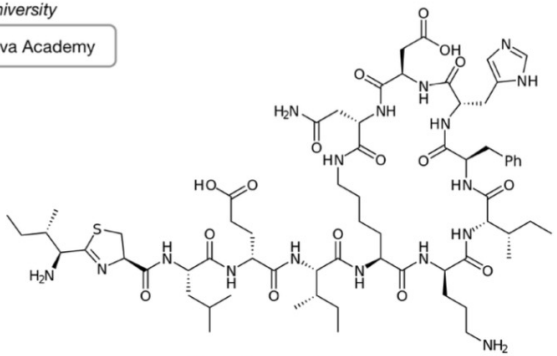
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

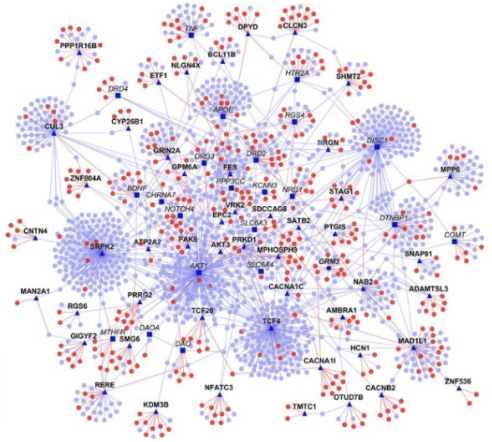




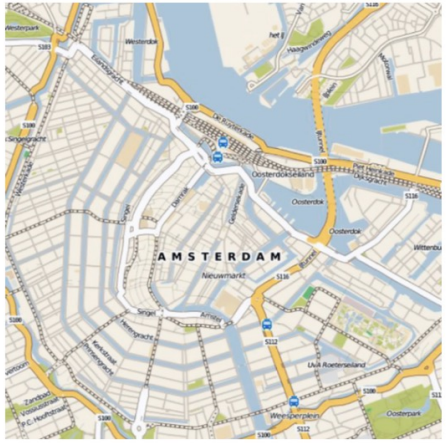
Knowledge graphs



Molecules



Protein interaction networks

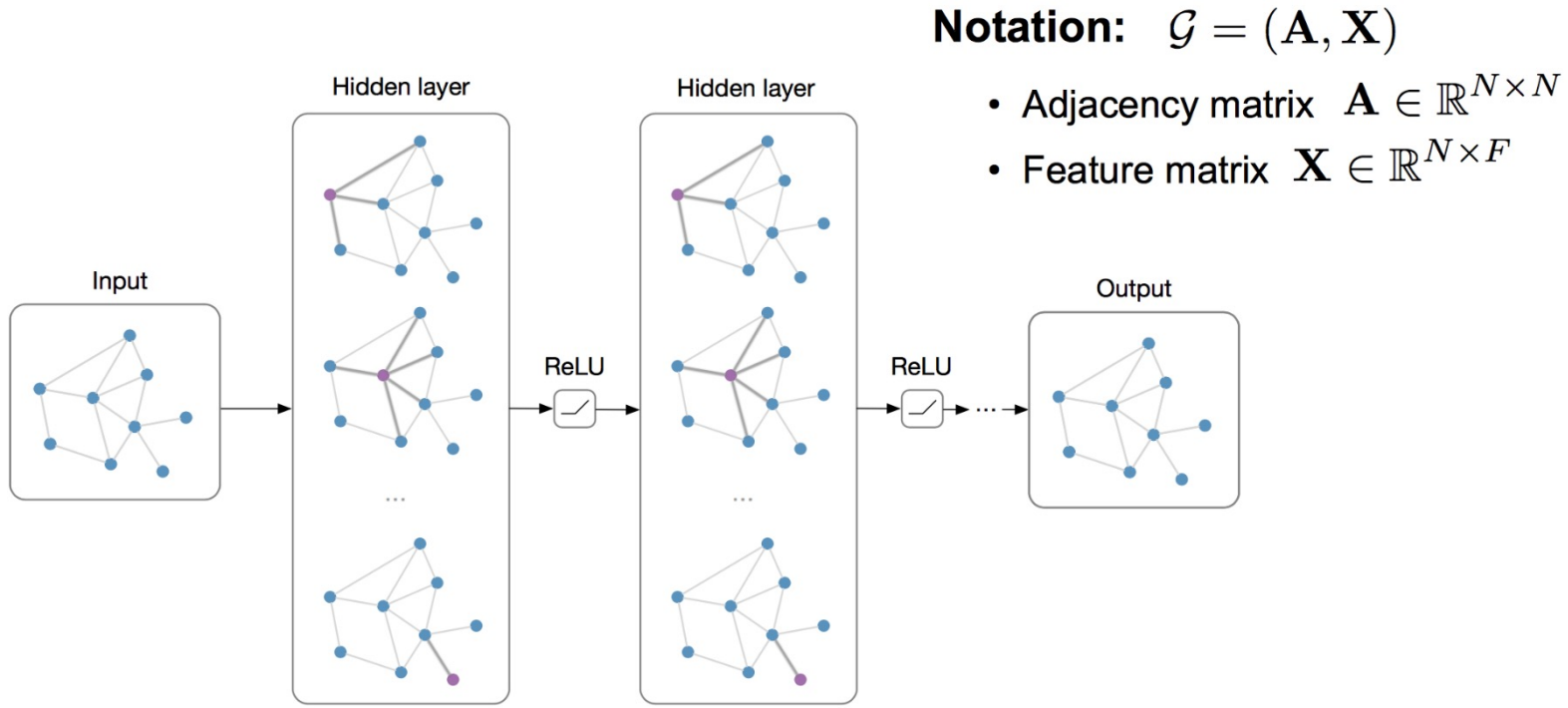


Road maps

Standard **CNN** and **RNN** architectures don’t work on this data

* slide from Thomas Kipf, **University of Amsterdam**

Graph Neural Networks



Main Idea: Pass massages between pairs of nodes and agglomerate

Alternative Interpretation: Pass massages between nodes to refine node (and possibly edge) representations

Graph Neural Networks: Types

Graph Convolutional Networks (GCNs) are similar to traditional CNNs. It learns features by inspecting neighboring nodes. GNNs aggregate node vectors, pass the result to the dense layer, and apply non-linearity using the activation function.

Graph Auto-Encoder Networks learn graph representation using an encoder and attempt to reconstruct input graphs using a decoder. The encoder and decoders are joined by a bottleneck layer.

Recurrent Graph Neural Networks(RGNNs) learn the best diffusion pattern, and they can handle multi-relational graphs where a single node has multiple relations.

Gated Graph Neural Networks (GGNNs) are better than the RGNNs in performing tasks with long-term dependencies.

Graph Neural Networks: Types of tasks