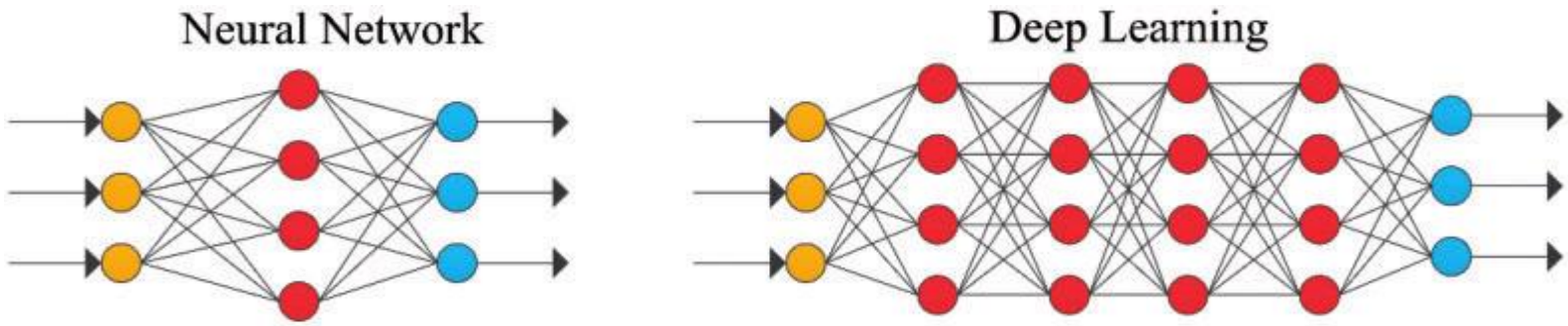
Μια εισαγωγή στην Βαθιά Μάθηση (Deep Learning)

Ιωάννης Χατζηλυγερούδης
(Διασκευή διαφανειών Νικολάου Γκοργκόλη)

Προσδιορισμός

- Πεδίο της μηχανικής μάθησης
- Βασίζεται στο τεχνητά νευρωνικά δίκτυα

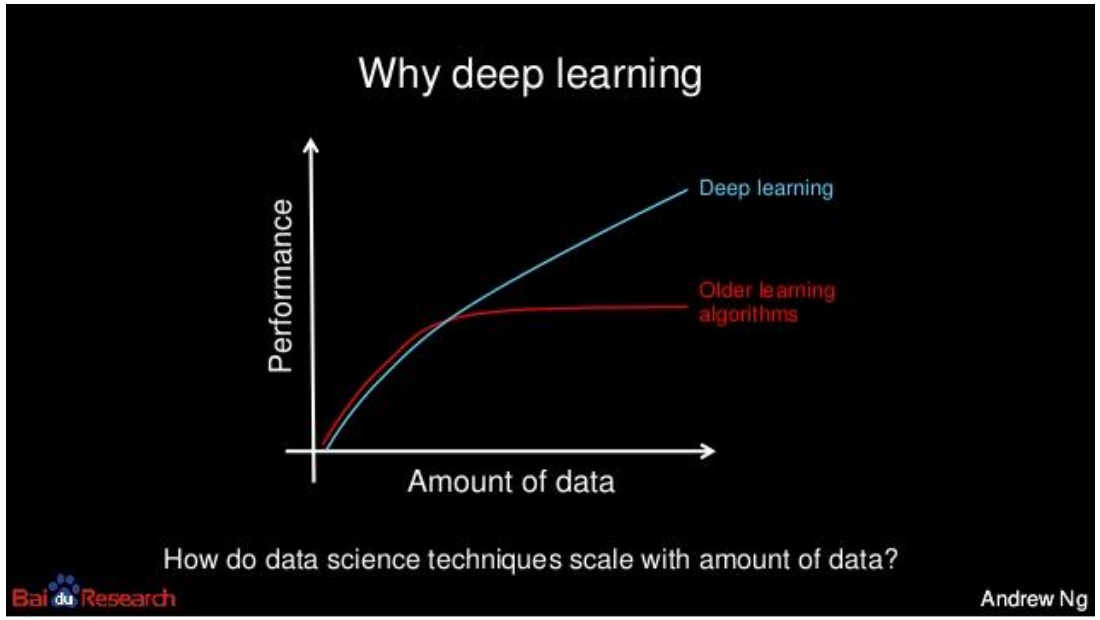
- Χρήση εκτεταμένων αρχιτεκτονικών και πολλαπλών αναπαραστάσεων. [1], [2], [3]



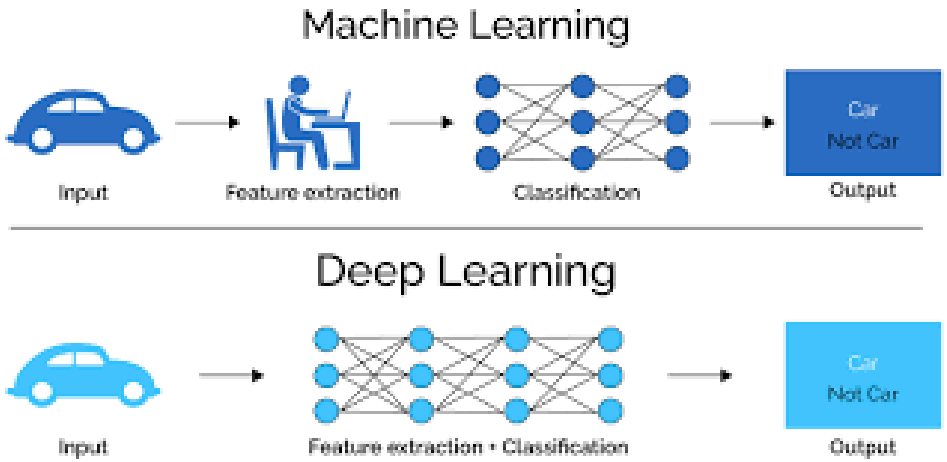
[5]

Κύρια Χαρακτηριστικά

- State of the art αποτελέσματα σε διάφορους τομείς.
- Αύξουσα μονοτονία της επίδοσης με το πλήθος των δεδομένων και των υπολογισμών
- Συνοψίζει διάφορα στάδια προεπεξεργασίας.
- Πλειοψηφικά (όχι αποκλειστικά) αναφέρεται σε επιβλεπόμενη μάθηση.
- Υπολογιστικά ακριβή διαδικασία
- Δυνητικά δύσκολη ερμηνεία των διαδικασιών.



[3]



[4]

Αλγόριθμοι Μάθησης

Gradient Descent Overview [8]

Μέθοδος ελαχιστοποίησης

- $J(\theta)$: αντικειμενική συνάρτηση
- $\theta \epsilon R^d$: παράμετροι
- η: σταθερά μάθησης
- n: μέγεθος παρτίδας

Σύμβαση: ο όρος SGD συνήθως αναφέρεται στη Mini – Batch GD.

- (Batch) Gradient Descent:
$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

- Stochastic Gradient Descent:
$$\theta = \theta - \eta \nabla_\theta J(\theta, x^{(i)}, y^{(i)})$$

- Mini – batch Gradient Descent:
$$\theta = \theta - \eta \nabla_\theta J(\theta, x^{(i:i+n)}, y^{(i:i+n)})$$

Αλγόριθμοι Μάθησης (1) [8]

- **Momentum**

$$u_t = \gamma \cdot u_{t-1} + \eta \cdot \nabla_\theta J(\theta) \ \ (update\ vector)$$

$$\theta = \theta - u_t \ \ (update\ rule)$$

  - Εκτενής περιγραφή: [8]

- **Nesterov Accelerated Gradient**

$$u_t = \gamma \cdot u_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma \cdot u_{t-1}) \ \ (update\ vector)$$

$$\theta = \theta - u_t \ \ (update\ rule)$$

  - Εκτενής περιγραφή: [8]

- **Adagrad** [9]

$$g_{t,i} = \nabla_\theta J(\theta_{t,i}) \ \ (parameter\ \theta_i, timestep\ t)$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i}^2 + \epsilon}} \cdot g_{t,i}$$

Αλγόριθμοι Μάθησης (2) [8]

RMSProp

- RMSprop and Adadelta have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates.

$$E[g_t^2] = 0.9E[g_{t-1}^2] + 0.1g_t^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[g_{t,i}^2] + \epsilon}} \cdot g_{t,i}$$

Αλγόριθμοι Μάθησης (3) [8]

Adam

- Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter.
- In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum:

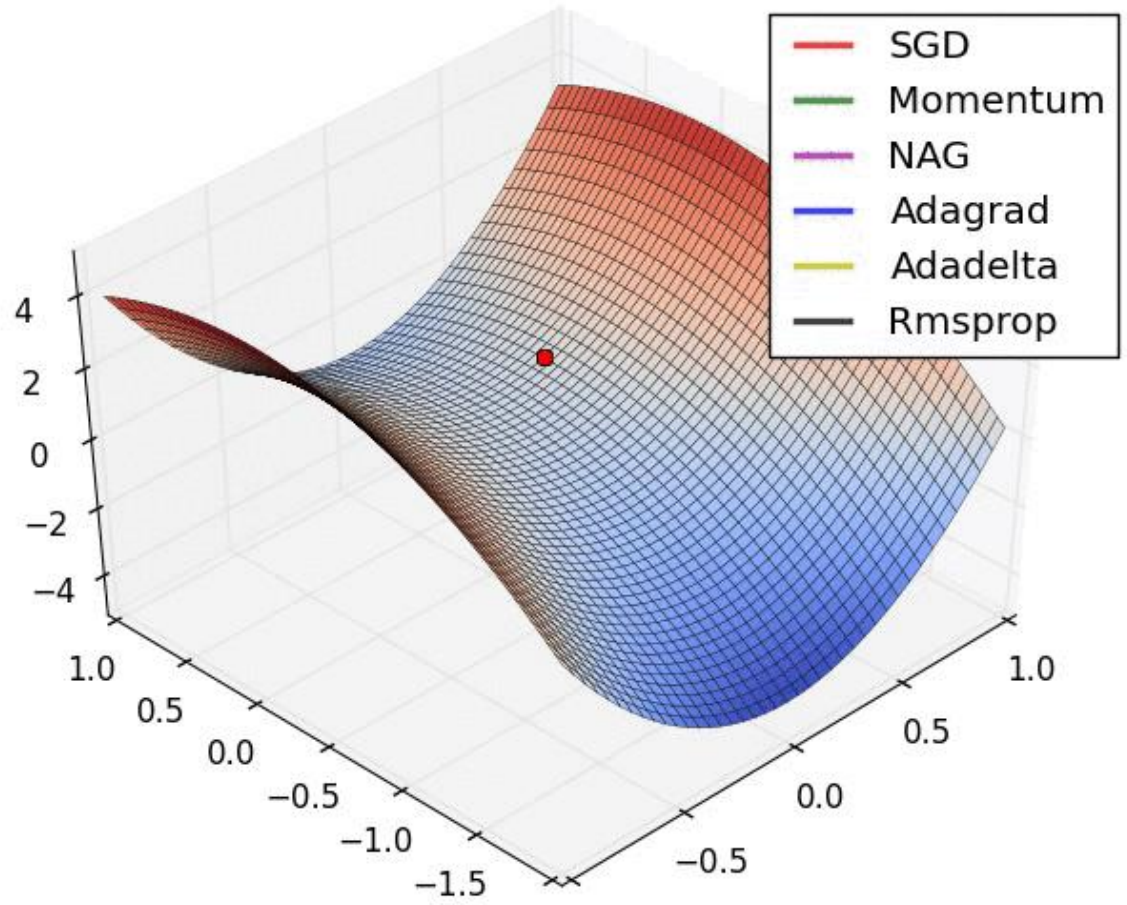
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{v_{t,i}} + \epsilon} . m_{t,i}$$

Αλγόριθμοι Μάθησης (4) [8]

Παράδειγμα σύγκλισης
σε saddle point.
($\nabla f = \vec{0}$, όχι βέλτιστο)

Πηγή: [8]

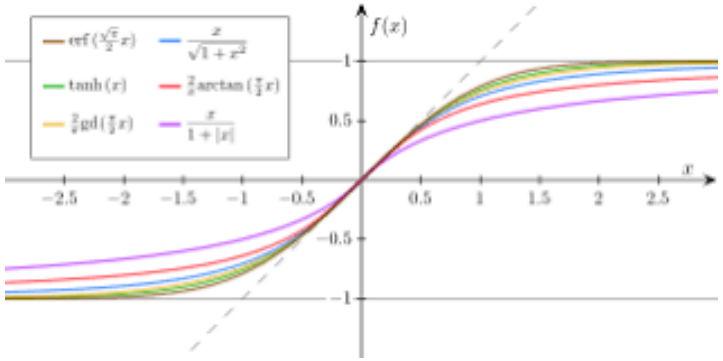


Αλγόριθμοι Μάθησης - Tips

- Δημοφιλέστεροι: **Adam**, **RMSprop**
- ΔΕΝ υπάρχει καθολικά καλύτερος αλγόριθμος μάθησης [10] (κάποιοι γενικεύουν καλύτερα, κάποιοι συγκλίνουν γρηγορότερα κτλ.)
- Εξάρτηση καταλληλότητας αλγορίθμου από το πρόβλημα που επιλύεται.
- Διαρκώς εξελισσόμενο πεδίο (βλ. Nadam [11] κ.α.).

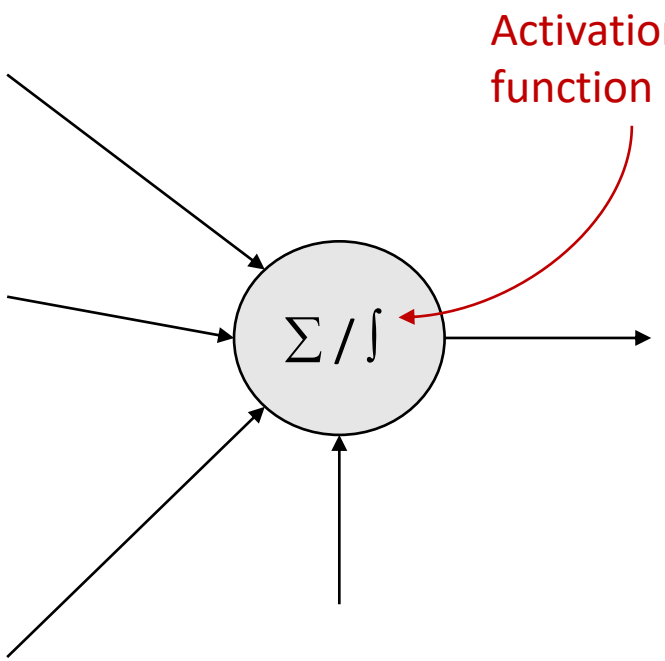
Συναρτήσεις Ενεργοποίησης

Συναρτήσεις Ενεργοποίησης



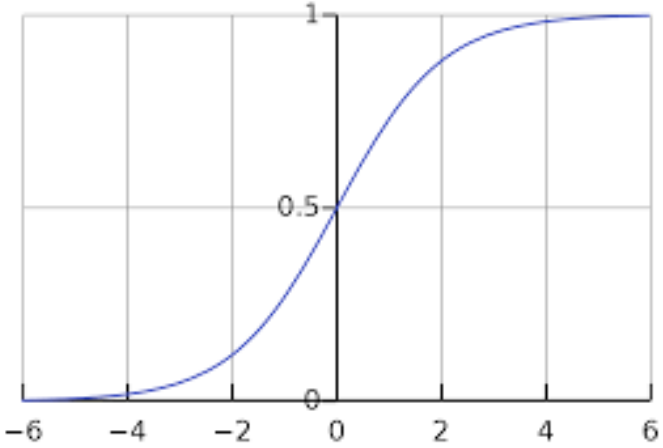
[12]

- Προσφέρουν μη γραμμικότητα στο μοντέλο.
  (διαφορετικά τα νευρωνικά θα μπορούσαν να προσεγγίσουν μόνο γραμμικές συναρτήσεις)
- Ρυθμίζουν και διαμορφώνουν την έξοδο των κρυφών επιπέδων.
- Επηρεάζουν αντίστοιχα την εκπαιδευτική διαδικασία.



Συναρτήσεις Ενεργοποίησης – Σιγμοειδής (Λογιστική)

$$S(x) = \frac{1}{1 + e^{-x}}$$

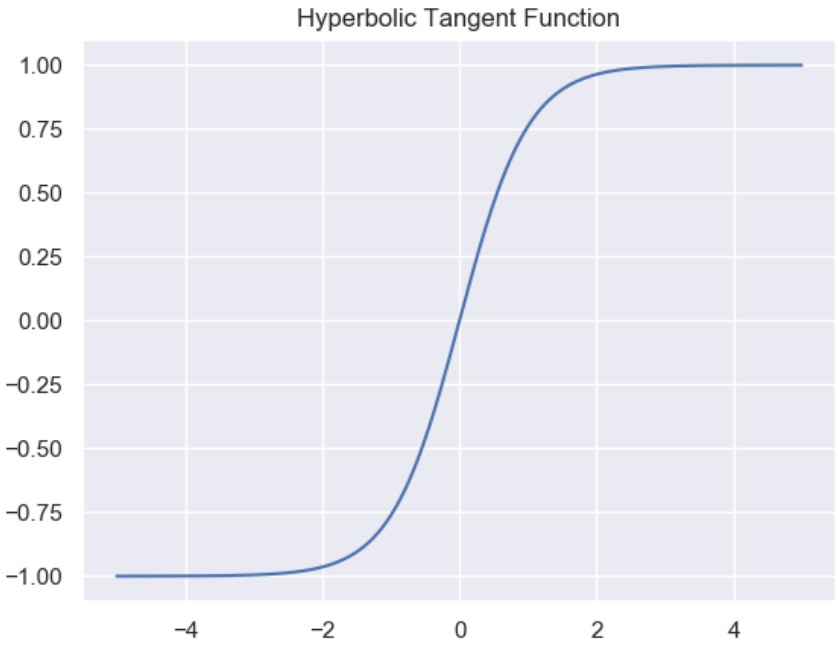


- Χαρακτηριστικά:
  - Μη γραμμική
  - Συνεχές πεδίο τιμών ενεργοποίησης
  - Σχετικά ήπια κλίση.
  - Εύρος πεδίου τιμών: (0, 1)
  - Εμφανείς αποκλίσεις στο διάστημα [-2, +2] (άρα κατάλληλο για classification)
- Μειονεκτήματα:
  - Μη συμμετρία ως προς το 0 → υπολογιστικές δυσκολίες
  - Αρκετά μικρή κλίση για ακραίες τιμές
  - Κατ’ επέκταση, εξασθένιση των παραγώγων μέσω των αλλεπάλληλων πολ/μων κατά το backpropagation.
  - Δυνητικά δυσκολεύει / καθυστερεί τη μάθηση

Συναρτήσεις Ενεργοποίησης – Υπερβολική Εφαπτομένη

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

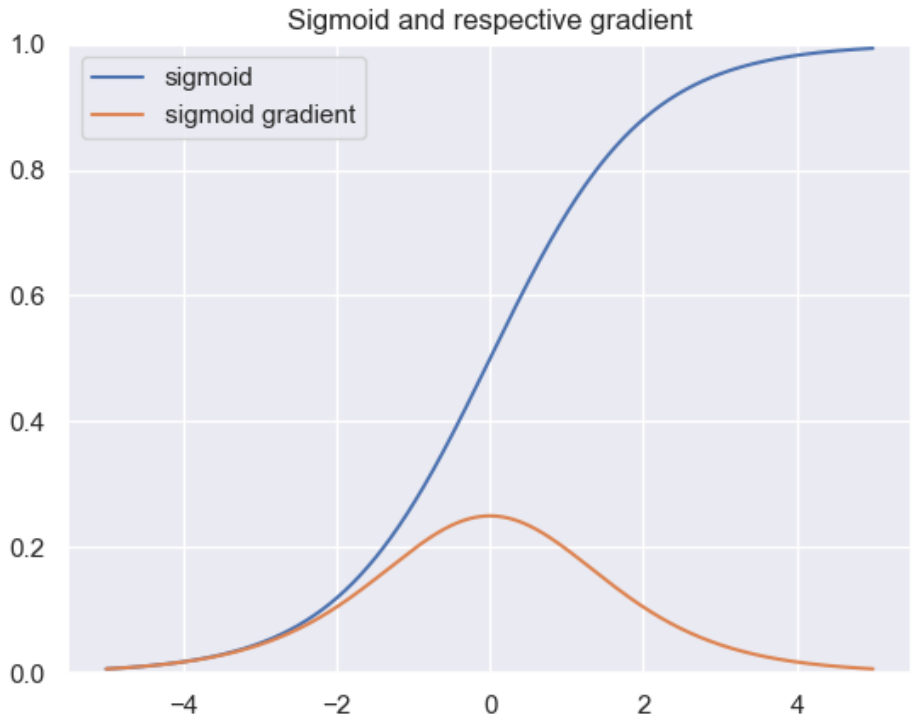
- Χαρακτηριστικά
  - $\tanh(x) = 2 * sigmoid(x) - 1$
  - Παρόμοια με λογιστική, ανήκει στην οικογένεια σιγμοειδών.
  - Πεδίο ενεργοποίησης: (-1, +1)
  - Εμφανίζει μεγαλύτερη κλίση σε σχέση με τη λογιστική.
  - Έχει ως κέντρο συμμετρίας το 0.



- Μειονεκτήματα:
  - Vanishing Gradient
  - Καθυστερεί / δυσκολεύει τη σύγκλιση (αντίστοιχα με λογιστική)

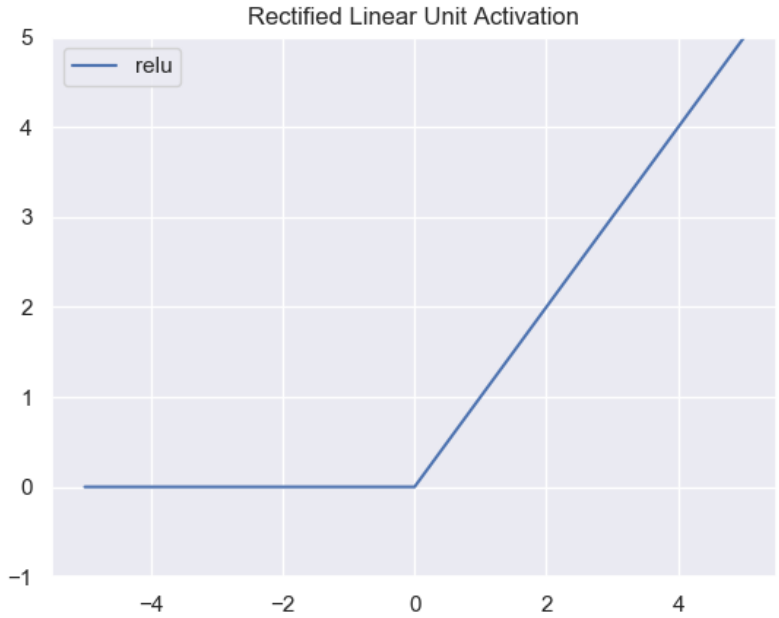
Vanishing Gradient

- Αναφέρεται στην επαγωγική μείωση των παραγόμενων τοπικών κλίσεων κατά την εκπαίδευση του δικτύου.
- Οι αλλεπάλληλοι πολλαπλασιασμοί κλίσεων του εύρους τιμών [0, 0.5) μέσω Gradient Descent, εκφυλίζουν την εκπαίδευση των βαρών.
- Είναι χαρακτηριστικό τόσο της λογιστικής, όσο και της υπερβολικής εφαπτομένης.



Συναρτήσεις Ενεργοποίησης Rectified Linear Unit (Relu)

$$f(x) = \max(0, x)$$

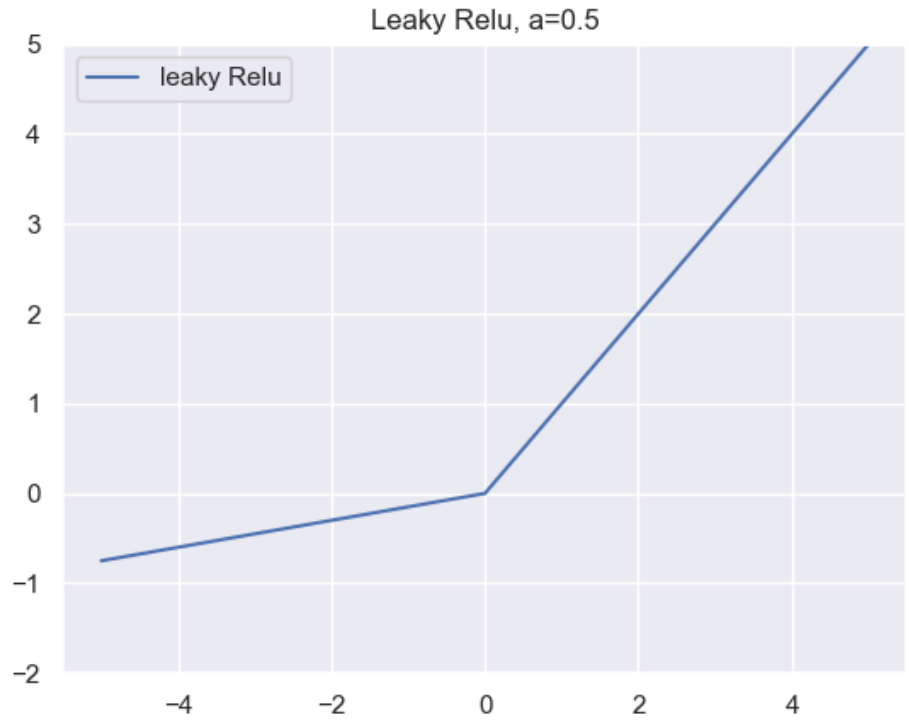


- Χαρακτηριστικά:
  - Μη γραμμική ($f(-1) + f(1) = 1$, $f(-1 + 1) = 0$)
  - Αραιότερες ενεργοποιήσεις – υπολογιστικά ευνοϊκή
  - Ιδανική για βαθιά δίκτυα
- Μειονεκτήματα:
  - Δεν είναι άνω φραγμένη
  - Για αρνητικές παραμέτρους, η τοπική παράγωγος είναι 0, επομένως δεν ανταποκρίνονται κατά την εκπαίδευση (dying Relu problem)

Συναρτήσεις Ενεργοποίησης – Leaky Relu

- $f(x) = \begin{cases} a * x, & \varepsilon\alpha\nu\ x < 0 \\ x, & \varepsilon\alpha\nu\ x > 0 \end{cases}$

- Χαρακτηριστικά:
  - Αποτελεί προσπάθεια διόρθωσης του dying Relu problem.
  - Αρκετά παρόμοια με την Relu
  - Μέσω του τελεστή a αποφεύγονται οι μηδενικές τοπικές παράγωγοι.



Συναρτήσεις Ενεργοποίησης - Softmax

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

- $softmax(\begin{bmatrix} 1.0 \\ 2.2 \\ -0.5 \end{bmatrix}) = \begin{bmatrix} 0.220 \\ 0.730 \\ 0.049 \end{bmatrix}$

- Χαρακτηριστικά:
  - Απεικονίζει ένα διάνυσμα πραγματικών αριθμών στον πιθανοτικό χώρο $[0, 1]$.
  - Λειτουργεί ως γενίκευση της λογιστικής για πολλές διαστάσεις.
  - Εφαρμόζεται πριν την παραγωγή εξόδου κατά το multiclass classification.
  - Κατασκευάζει μια κατανομή των προβλέψεων βάση των logits.

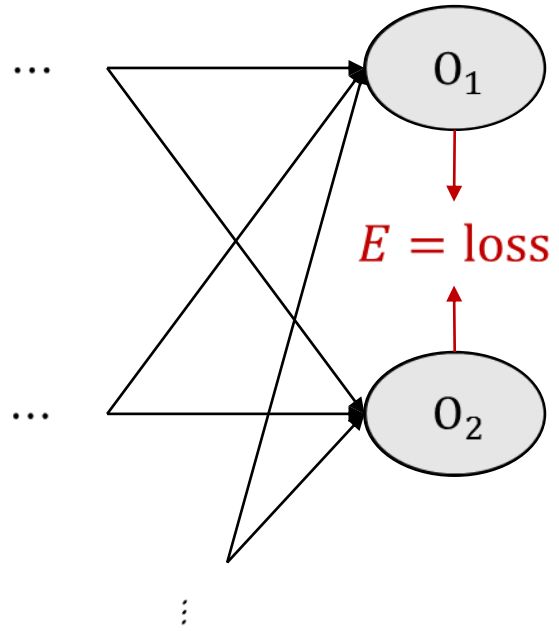
Συναρτήσεις Ενεργοποίησης – Παρατηρήσεις

- Η καταλληλότερη επιλογή της συνάρτησης κόστους διαφέρει ανάλογα με το πρόβλημα.
- Υπάρχει μια γενική προτίμηση ως προς τη Relu ως αρχική προσέγγιση.
- Ιδιαίτερη προσοχή ως προς το πρόβλημα Vanishing / Exploding Gradient (χρήση τεχνικών gradient monitoring, gradient clipping).
- Σε multiclass classification, χρήση της συνάρτησης Softmax πριν την παραγωγή της εξόδου.

Συνάρτηση Κόστους

Συνάρτηση Κόστους – Γενική Περιγραφή

- Σκοπός: Χρήση της συνάρτησης κόστους που αντικατοπτρίζει κατάλληλα το πρόβλημα.
- Κύριος διαχωρισμός: Πρόβλεψη διακριτής ή συνεχούς τιμής. (classification / regression)



Συνάρτηση Κόστους - Regression Loss

- Mean Squared Error :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \\ = \frac{1}{n}\sum_{i=1}^{n}(E_i)^2$$

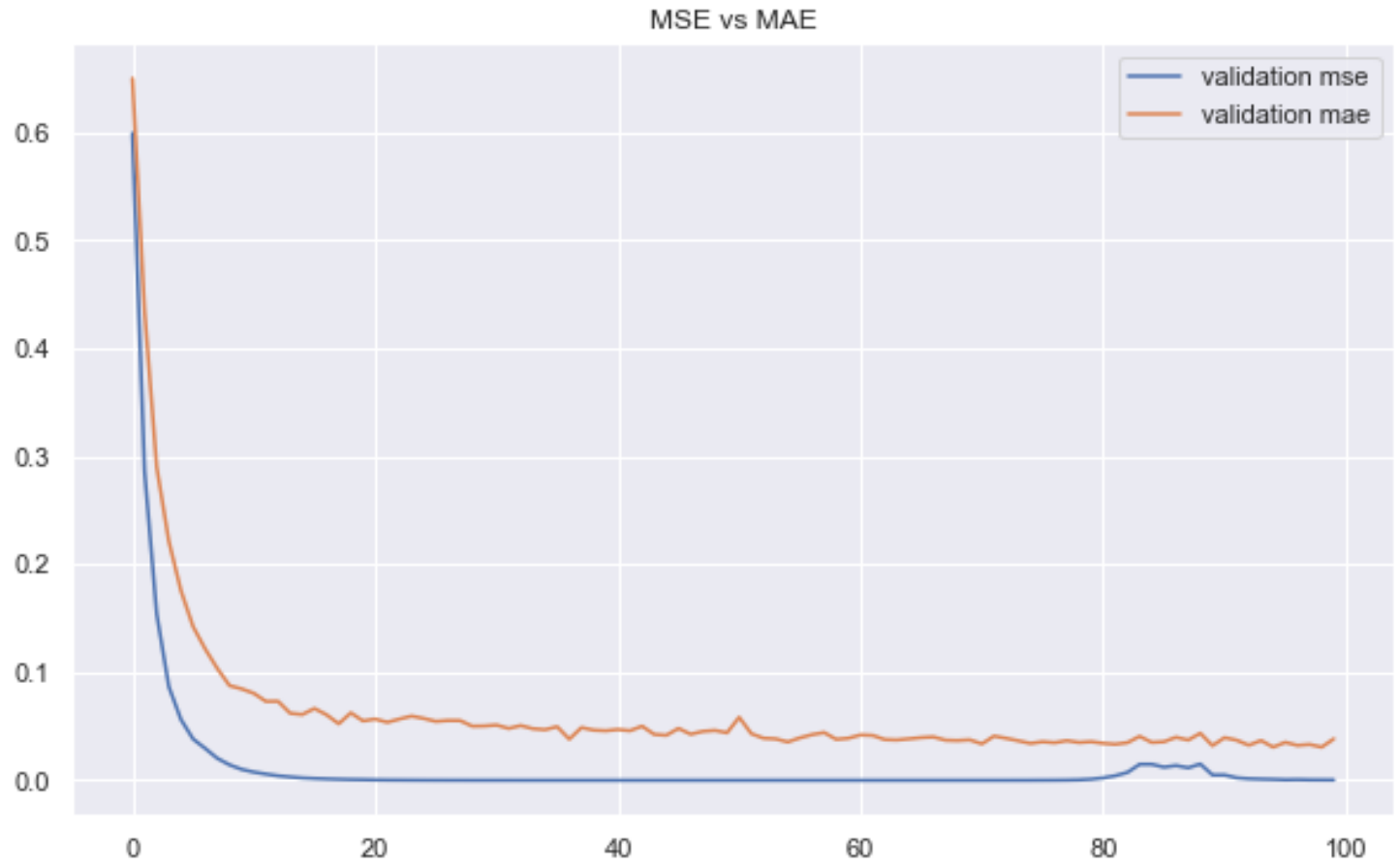
  - Ίσως η πιο συνηθισμένη προσέγγιση.
  - Ιδανικά όταν τα δεδομένα ακολουθούν κανονική κατανομή.

- Mean Absolute Error:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| = \\ = \frac{1}{n}\sum_{i=1}^{n}|E_i|$$

  - Ανταποκρίνεται αποδοτικότερα σε περίπτωση πολλών outliers.

Συνάρτηση Κόστους - Regression Loss – Παράδειγμα



Συνάρτηση Κόστους - Classification Loss

- **Categorical Cross – Entropy**

$$CE = -\frac{1}{N}\sum_{c\epsilon C}\sum_{s\epsilon S} t_{s\epsilon c}\log(s\epsilon c)$$

  - Στην ουσία, υπολογίζει το σφάλμα της παραγόμενης κατανομής σε σχέση με την κατανομή των πραγματικών τιμών (labels).
  - Αποτελεί την κύρια συνάρτηση κόστους για multiclass classification.
  - Απαιτεί one – hot encoding των labels.

- **Sparse Categorical Cross – Entropy**
  - Κατάλληλη όταν οι κλάσεις ως υποσύνολα είναι ξένα μεταξύ τους (mutually exclusive)
  - Δεν απαιτεί one – hot encoding των labels, λειτουργεί με μη διανυσματική ακέραια αναπαράσταση των κλάσεων.
  - Ωφελεί αρκετά την εκπαιδευτική διαδικασία όταν υπάρχει μεγάλο πλήθος κλάσεων.

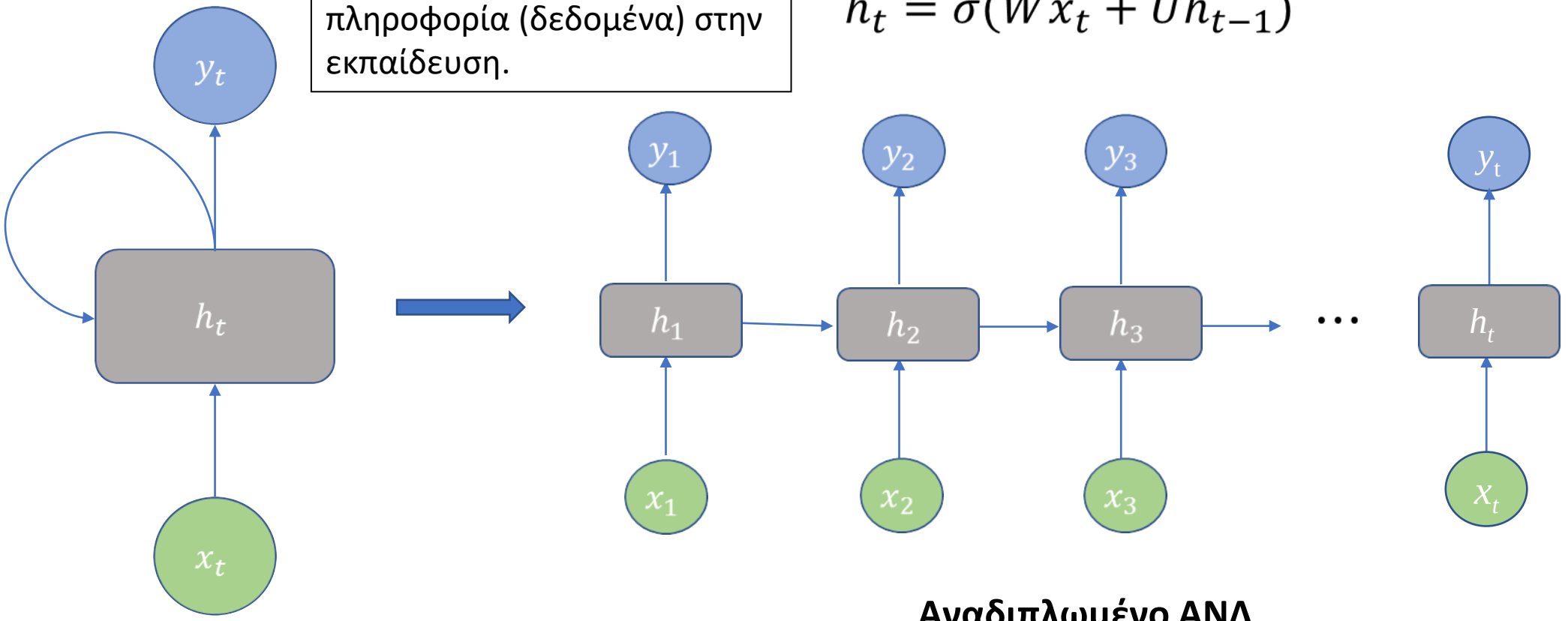
- **Εκτενής περιγραφή και παραλλαγές**: [13] ή https://keras.io/losses/

Αναδρομικά Νευρωνικά Δίκτυα

Αναδρομικά Νευρωνικά Δίκτυα-ΑΝΔ (Recurrent Neural Networks-RNN) – Βασική Δομή

Χρησιμοποιεί προηγούμενη πληροφορία (δεδομένα) στην εκπαίδευση.

$$h_t = \sigma(Wx_t + Uh_{t-1})$$



Αναδιπλωμένο ΑΝΔ

Αναδρομικά Νευρωνικά Δίκτυα – Περιγραφή

$$h_t = \sigma(W x_t + U h_{t-1})$$

- x_t: είσοδος του δικτύου κατά το χρονοβήμα t.
- h_t: έξοδος του κρυφού επιπέδου (hidden state) κατά το χρονοβήμα t.
- y_t: έξοδος του δικτύου (hidden state) κατά το χρονοβήμα t.
- W, U: Μητρώα βαρών.

- **Βασικό χαρακτηριστικό**: η έξοδος του κρυφού επιπέδου (h_t) τροφοδοτείται συμπληρωματικά με τη νέα είσοδο (x_{t+1}) στο κρυφό επίπεδο.
- Το δίκτυο αποκτά “μνήμη”, καθίσταται έτσι κατάλληλο για την επεξεργασία ακολουθιακών δεδομένων.

Αναδρομικά Νευρωνικά Δίκτυα – Κίνητρο

- Θέλουμε να προβλέψουμε την τελευταία λέξη στην πρόταση:

  «The ships move on the _____»

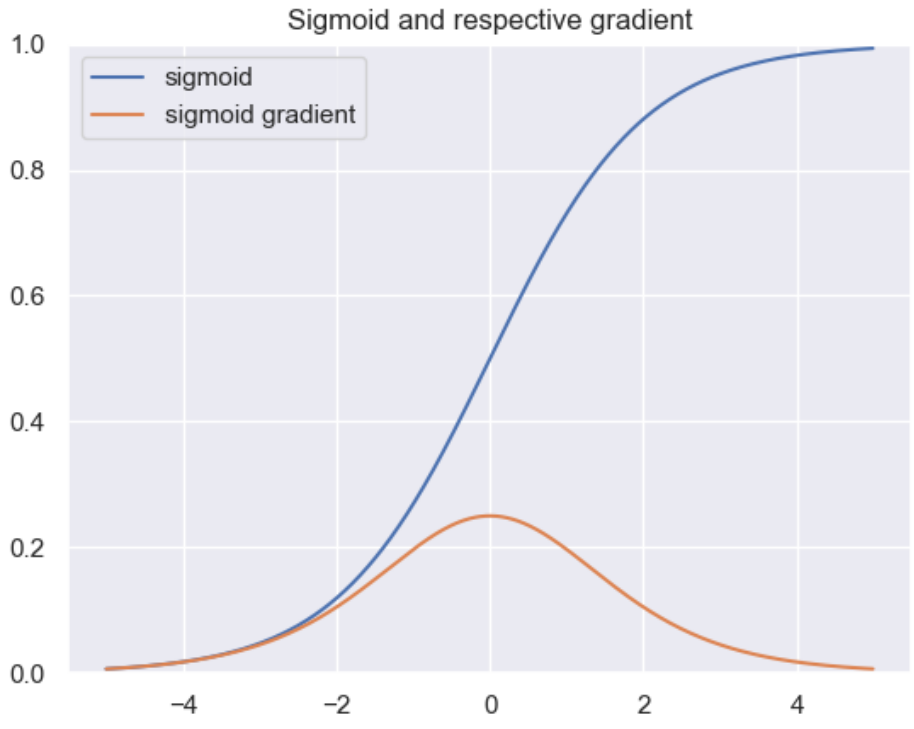
- Για να μπορεί να γίνει αυτό πρέπει να πρέπει να στηριχτούμε στις προηγούμενες λέξεις, οπότε να καταλήξουμε στη λέξη «sea». Αυτό λέγεται short-term dependence.
- Αν τώρα θέλουμε να προβλέψουμε τη λέξη στη πρόταση:

  «George speaks fluent _____»

- Τα πράγματα είναι δυσκολότερα. Οι προηγούμενες λέξεις δεν μας βοηθούν και πρέπει να ανατρέξουμε σε προηγούμενες προτάσεις: «George lives in Greece», οπότε καταλήγουμε στη λέξη «Greek».
- Αυτά είναι κάτι που τα συνήθη ΝΔ δεν μπορούν να κάνουν. Το κάνουν τα ΑΝΔ.

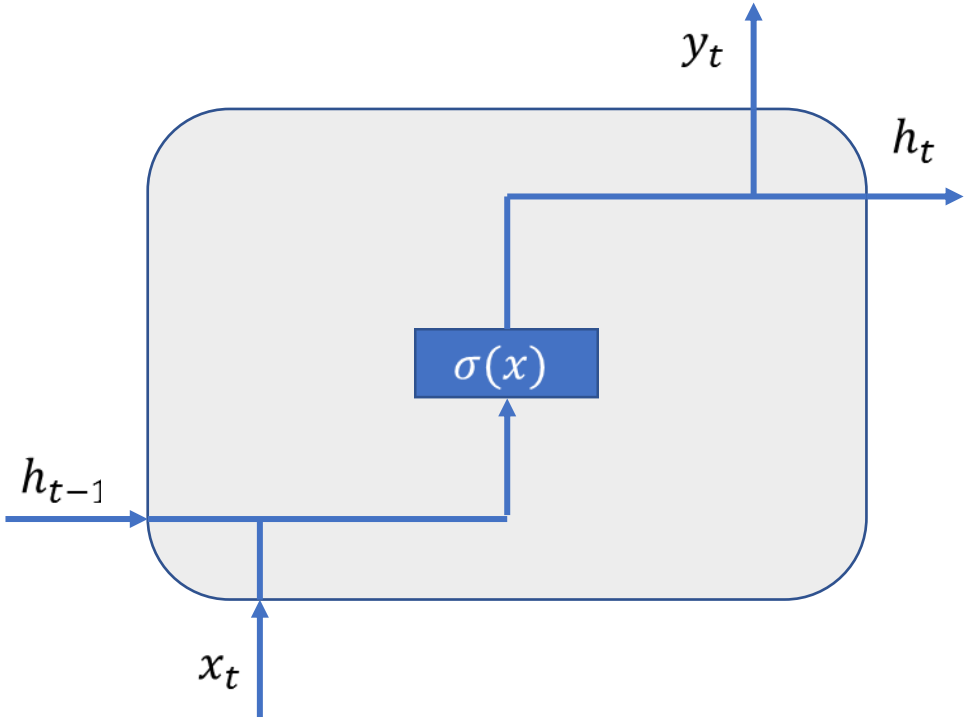
Αδυναμία Μακροχρόνιων Εξαρτήσεων

- Όπως και τα απλά ΝΔ, τα ΑΝΔ είναι universal function estimators.
- Θεωρητικά, έχουν την ικανότητα απεριόριστης “μνήμης”.
- Αλλεπάλληλοι πολ/μοι τοπικών κλίσεων (βλ. vanishing gradient problem) → αδυνατούν να συγκρατήσουν μακροχρόνιες (long - term) εξαρτήσεις.
- Ως απάντηση χρησιμοποιούνται ανεπτυγμένες αρχιτεκτονικές και τεχνικές κανονικοποίησης.

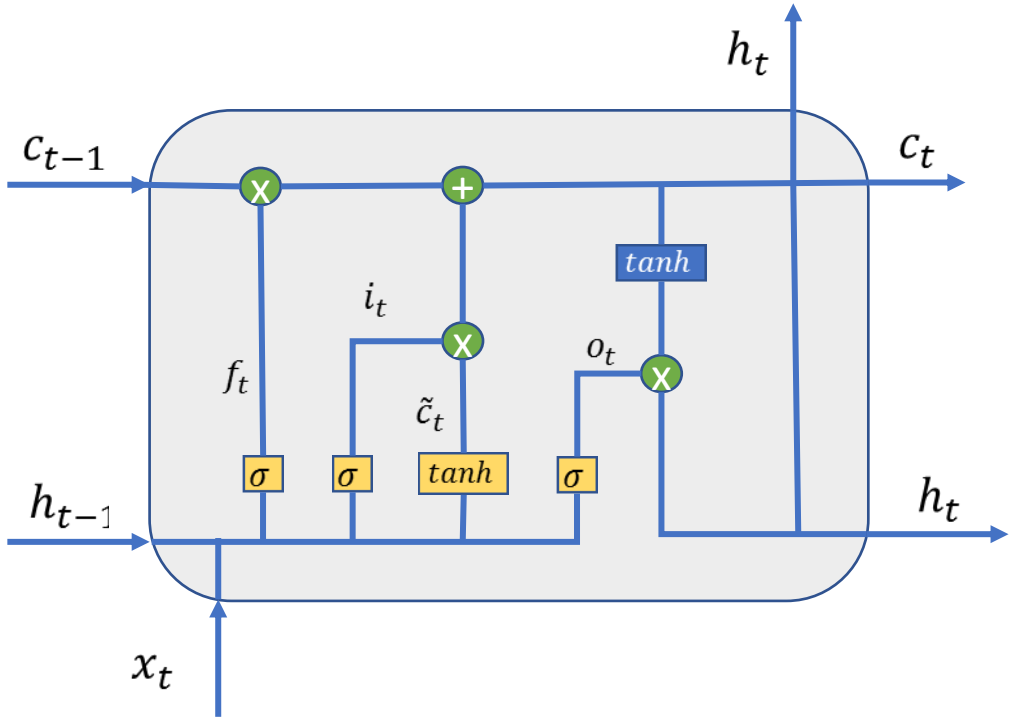


Long Short – Term Memory Units (LSTM)

Βασικό ΑΝΔ



LSTM [14, 16]



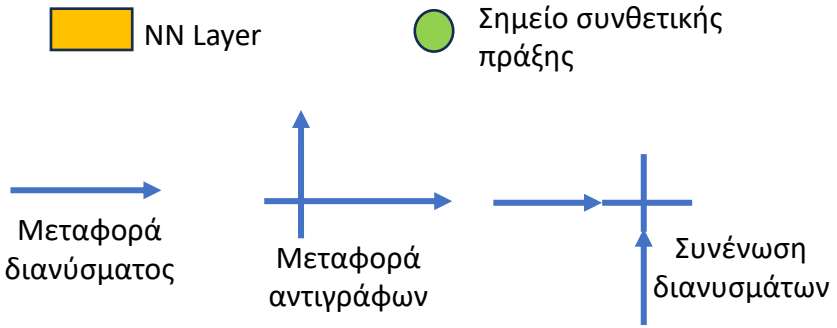
Κύρια πηγή: [14]

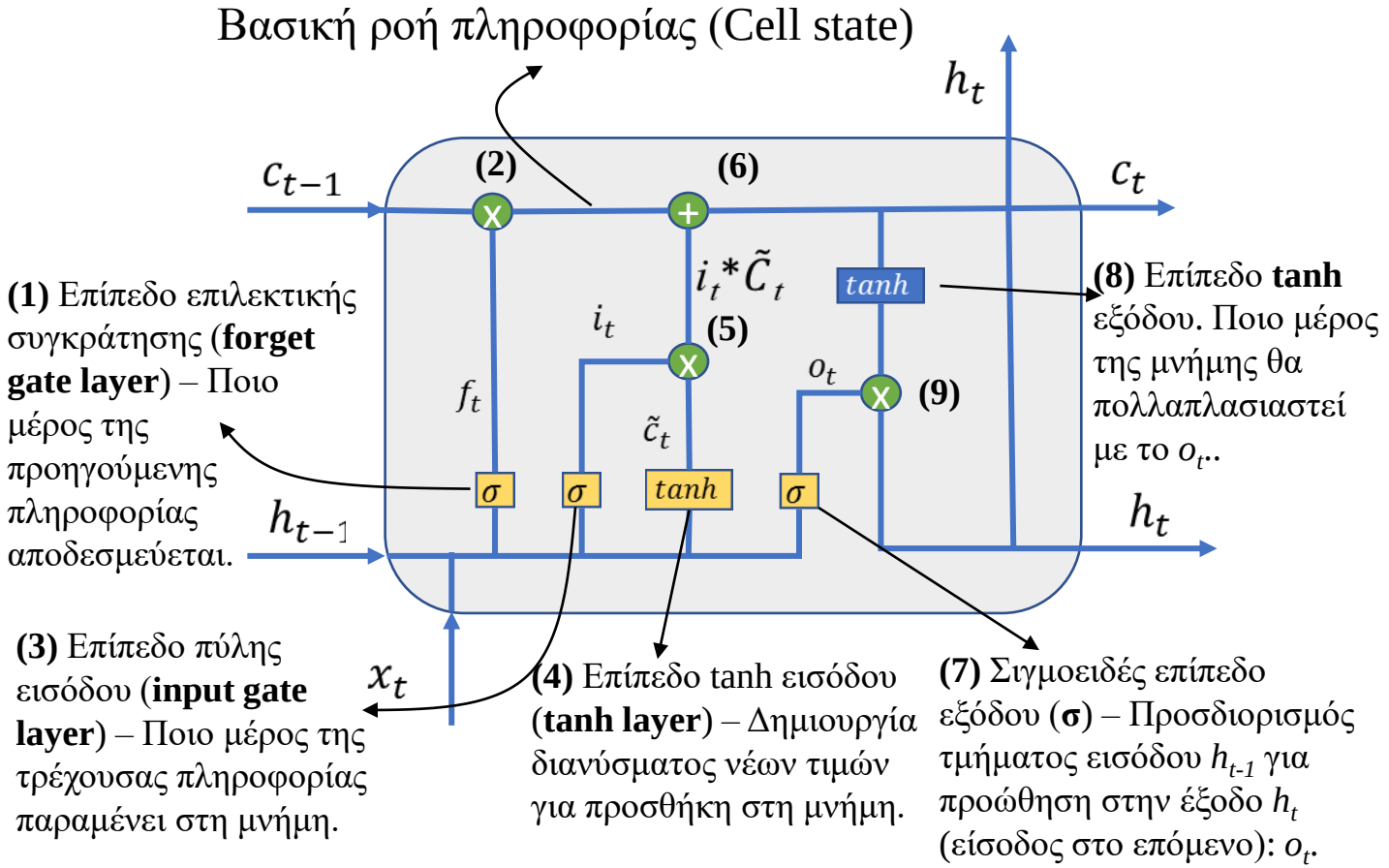
LSTM – Περιγραφή (1)

- $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- $\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
- $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

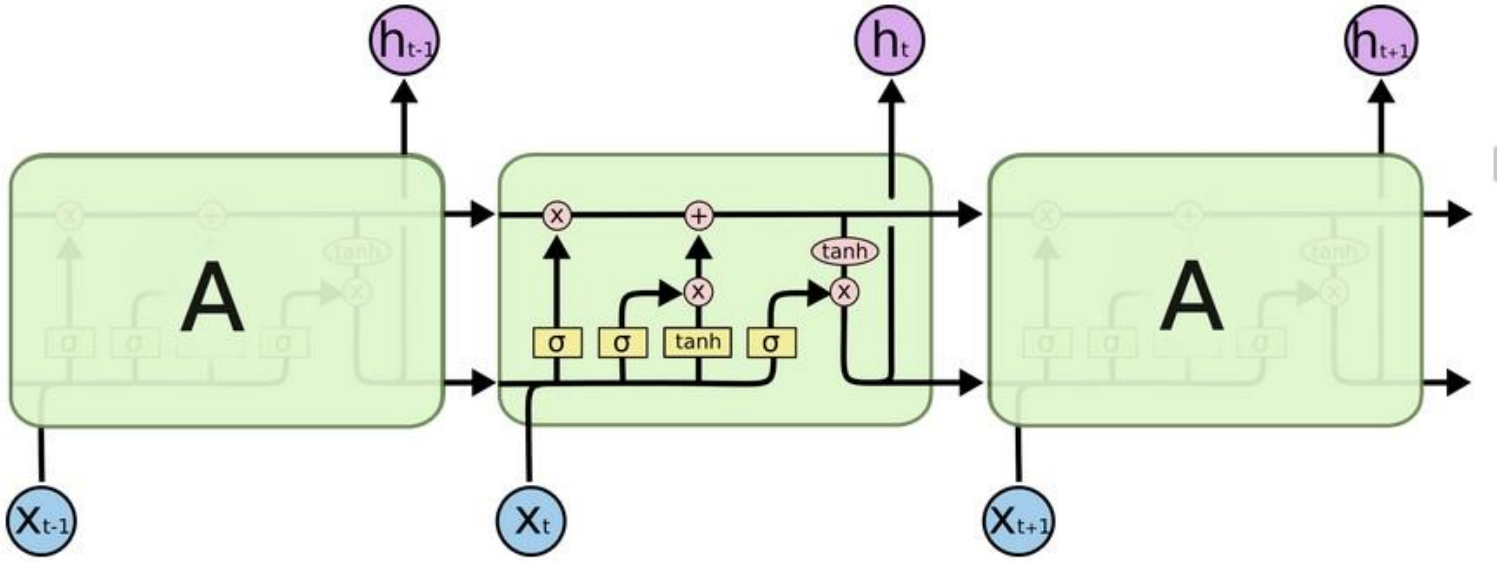
- $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- $h_t = o_t * \tanh(C_t)$

- Κύρια πηγή: [14]



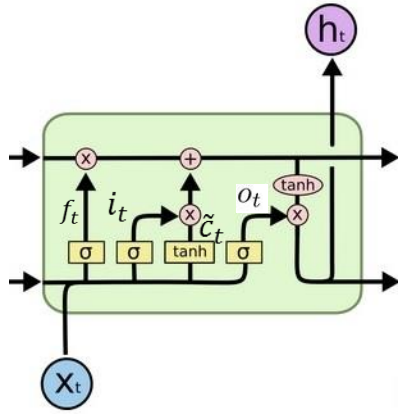


LSTM – Περιγραφή (2)



- The key to LSTMs is the **cell state**, the horizontal line running through the top of the diagram.
- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called **gates**. Gates are composed out of a **sigmoid** neural net layer and a pointwise multiplication operation.

LSTM – Περιγραφή (3)

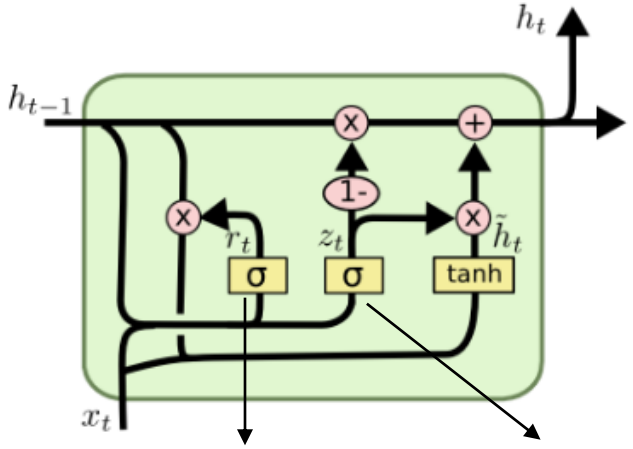


LSTM Operation

1. The **forget gate** layer decides what information we're going to throw away from the cell state (f_t):
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
2. The **input gate** layer decides what new information we're going to store in the cell state (i_t).
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
3. A **tanh layer** creates a vector of new candidate values ($\tilde{c}_t$) that could be added to the state.
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
4. We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add i_t * $\tilde{c}_t$. This is the new candidate values, scaled by how much we decided to update each state value. So, the new cell state is produced:
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
5. The **output** is a filtered version of the cell state.
   - First, we run a sigmoid layer which decides what parts of the cell state we're going to output.
   $$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
   - Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to
   $$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Units (GRU)

- Η αρχιτεκτονική GRU [17] αποτελεί μια απλοποιημένη μορφή της LSTM.
- Δεν περιέχει cell state.
- Απαρτίζεται από μια update και μια reset πύλη.
- Και οι δυο αρχιτεκτονικές θεωρούνται ισάξιες και εξίσου δημοφιλείς.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

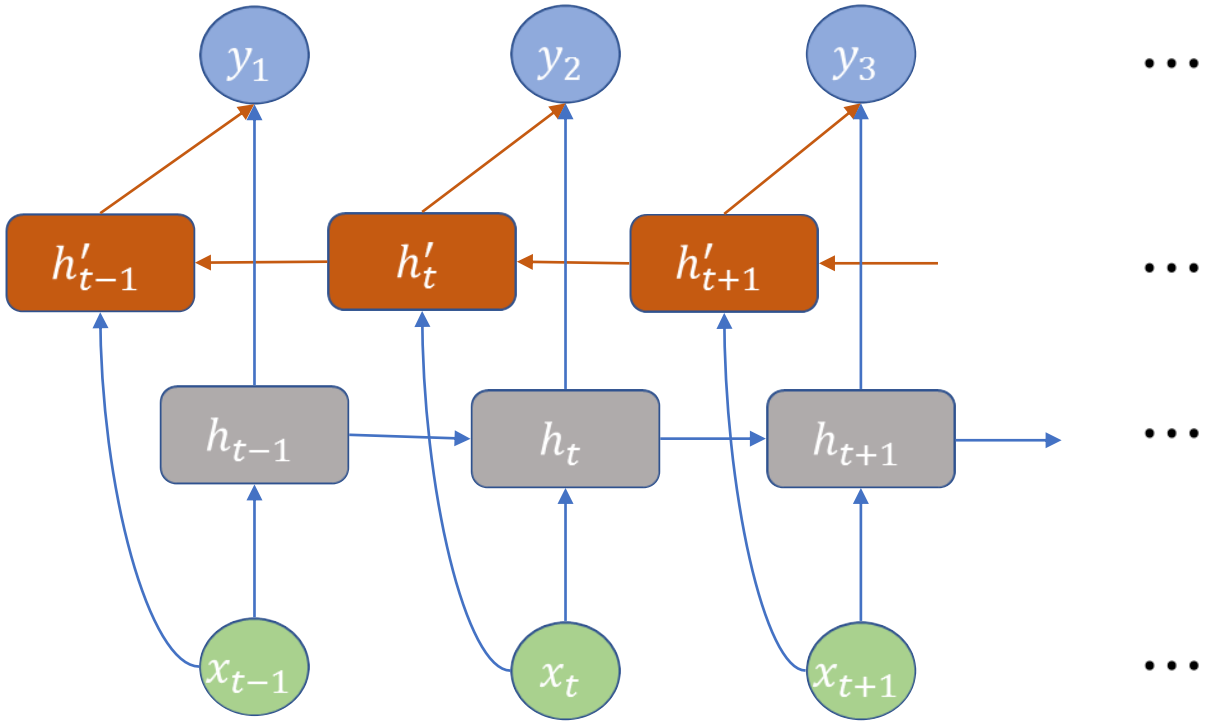
Πύλη ενημέρωσης (**update gate**) – Συγχώνευση input και forget gates.

Πύλη επανεκκίνησης (**reset gate**).

Εικόνα από: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Προχωρημένες Αρχιτεκτονικές – Bidirectional RNN

- Παράλληλα με την κανονική ανάγνωση των δεδομένων, ένα έξτρα κρυφό επίπεδο επεξεργασίας δέχεται την είσοδο με αντίστροφη σειρά. Οι έξοδοι των κρυφών επιπέδων συνενώνονται σε μια κοινή έξοδο.
- Κατ’ αυτό το τρόπο το δίκτυο δημιουργεί συσχετίσεις όχι μόνο με τα παρελθοντικά δεδομένα, αλλά και με αυτά που έπονται.



Αναδρομικά Νευρωνικά Δίκτυα - Χρήσιμοι Σύνδεσμοι

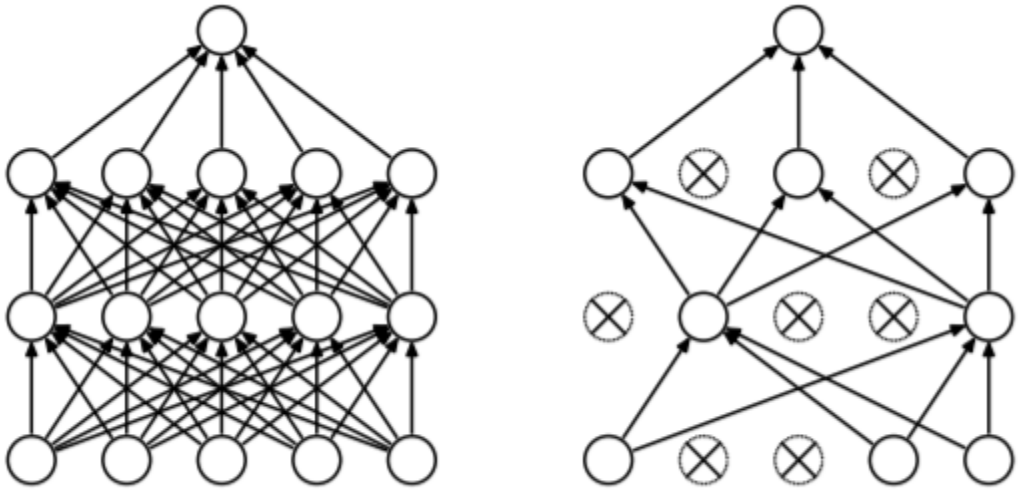
Χρήσιμοι σύνδεσμοι:

- https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://playground.tensorflow.org/

Τεχνικές Κανονικοποίησης

Τεχνικές Κανονικοποίησης – Dropout (1)

- Το dropout [18] αποτελεί τη βασική τεχνική κανονικοποίησης νευρωνικών δικτύων.
- Τυχαία επιλεγμένοι κόμβοι απενεργοποιούνται κατά την εκπαιδευτική διαδικασία. Αποφεύγεται έτσι ενδεχόμενο overfitting.

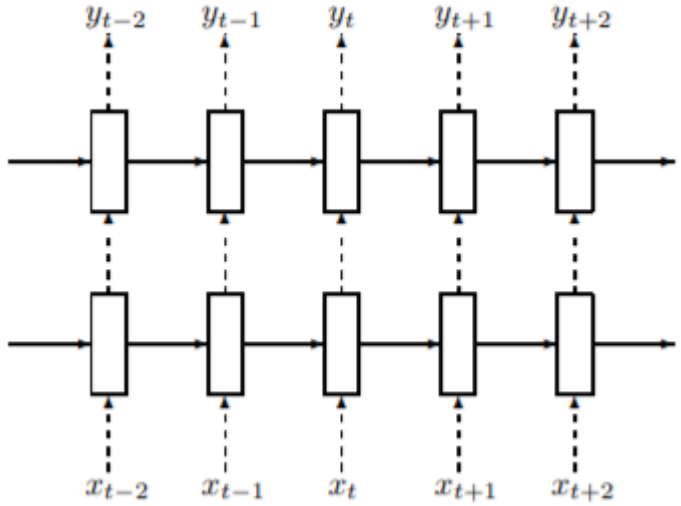


(a) Standard Neural Net (b) After applying dropout.

Εικόνα από τη δημοσίευση [18].

Τεχνικές Κανονικοποίησης – Dropout (2)

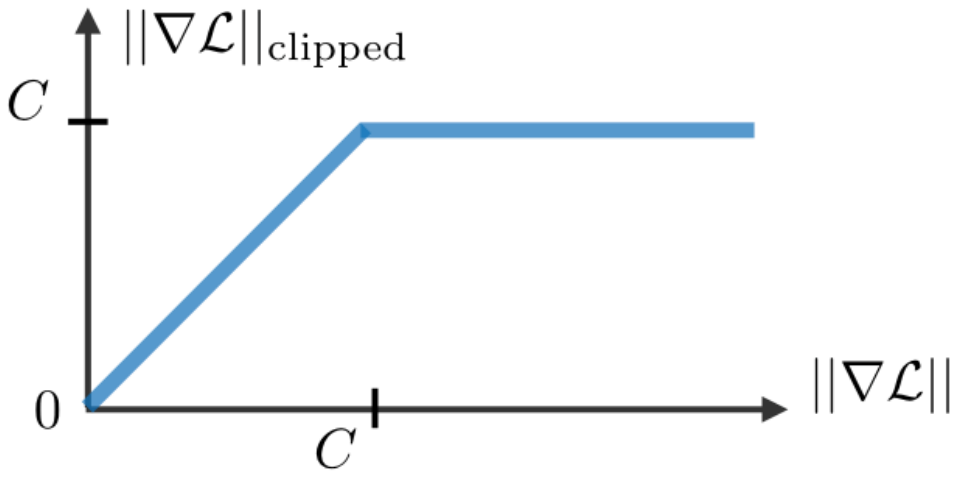
- Στην περίπτωση των αναδρομικών νευρωνικών , το dropout προτείνεται να υλοποιείται μόνο στις μη αναδρομικές συνάψεις, για τη κατάλληλη διατήρηση της πληροφορίας (βλ. [19]).



Εικόνα από τη δημοσίευση [19].

Τεχνικές Κανονικοποίησης – Gradient Clipping

- Εφαρμόζεται ως άνω φράγμα στη τιμή των υπολογιζόμενων τοπικών κλίσεων κατά την εκπαίδευση, για την αποφυγή του gradient explosion και γρηγορότερη σύγκλιση.
- Συνήθως υλοποιείται μέσω κανονικοποίησης της τιμής της κλίσεως, όταν η L_2 νόρμα της ξεπεράσει το ορισμένο άνω φράγμα.



Εικόνα από: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Convolutional Neural Networks
(Συνελικτικά Νευρωνικά Δίκτυα)

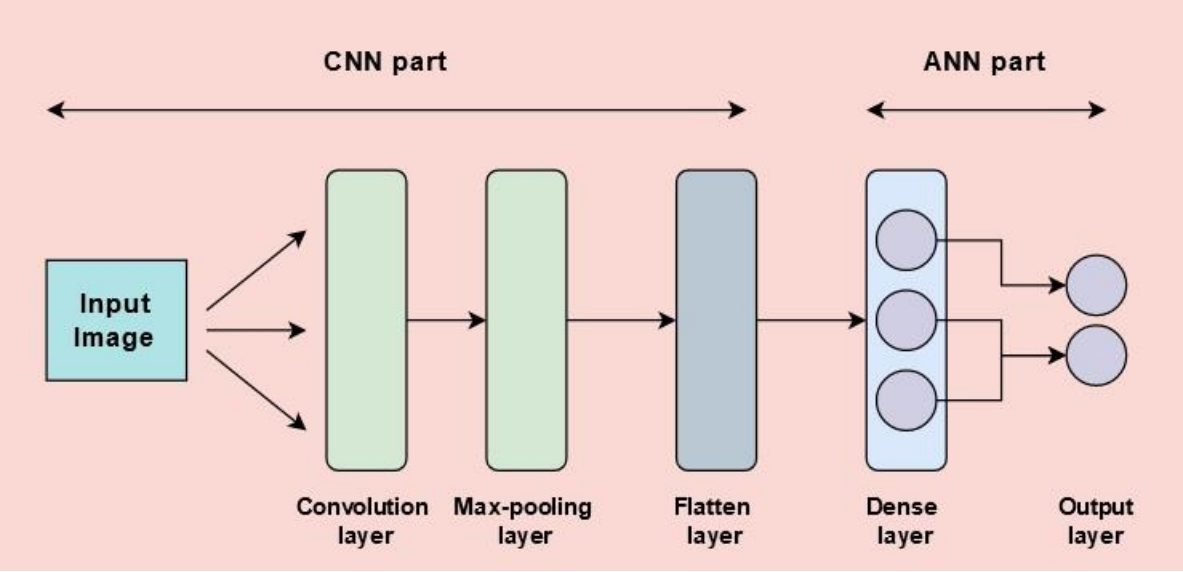
Προβλήματα Δικτύων Εμπρόσθιας Τροφοδότησης (FNN) για αναγνώριση εικόνας

- Η πλήρης συνδεσιμότητα μεταξύ νευρώνων των διαδοχικών επιπέδων έχει ως αποτέλεσμα τον προσδιορισμών πάρα πολλών τιμών βαρών.
- Έστω μια ασπρόμαυρη εικόνα εισόδου 64X64 pixels, με μέγεθος καναλιού (channel size) 1, δηλ. 1 τιμή/pixel (= απόχρωση γκρί από 0-255), έχουμε 64X64X1 = 4.096 τιμές εισόδου και επομένως εισόδους.
- Ας υποθέσουμε ότι το πρώτο κρυφό επίπεδο έχει 500 νευρώνες. Τότε μιλάμε για 4,096 X 500 = 2.048.000 τιμές βαρών για προσδιορισμό. Δεδομένων και των υπόλοιπων κρυφών επιπέδων, ο αριθμός αυτός γίνεται ακόμη μεγαλύτερος.
- Αυτό κάνει την εκπαίδευση του δικτύου χρονοβόρα και αυξάνει την πιθανότητα για υπερεκπαίδευση (overfitting). Η κατάσταση γίνεται χειρότερη στην περίπτωση τρισδιάστατων εικόνων όπου channel size = 3 (3 τιμές χρωμάτων για κάθε pixel).
- Ένα άλλο πρόβλημα είναι ότι 2-διάστατες εικόνες αναπαριστάνονται ως μονοδιάστατα διανύσματα.

Convolution Neural Networks (CNNs)

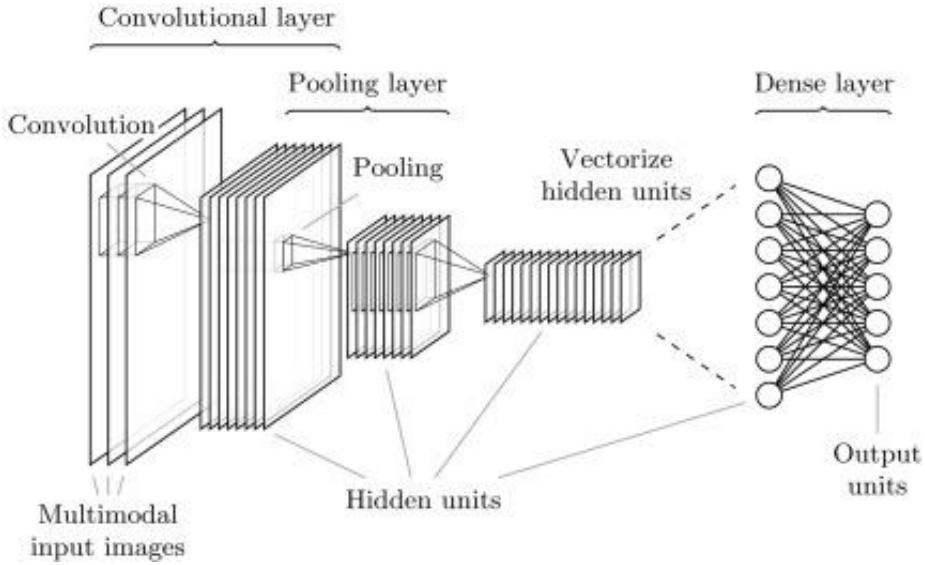
Ένα πλήρες δίκτυο CNN μπορεί να θεωρηθεί ότι αποτελείται από δύο μέρη:

- Ένα δίκτυο CNN που περιλαμβάνει τα συνελικτικά επίπεδα συνδυασμένα και με άλλους τύπους επιπέδων
- Ένα ANN (ΤΝΔ), συνήθως εμπρόσθιας ανατροφοδότησης, που αποτελείται από ένα ή περισσότερα λεγόμενα πυκνά επίπεδα (dense layers)



Convolution Neural Networks - Περιγραφή

- Αποτελούν feed – forward δίκτυα, όπως τα MLP’s.
- Έχουν την ικανότητα να ανακαλύπτουν χωρικές και χρονικές εξαρτήσεις.
- Μειώνουν κατά πολύ τις εκπαιδευόμενες παραμέτρους του μοντέλου και τις πιθανότητες overfitting.
- Η είσοδος θεωρείται ως ένα αρχείο εικόνας.
- Κατηγορίες επιπέδων:
  - Input
  - Convolutional
  - Pooling
  - Flatten
  - Fully – Connected (Dense)

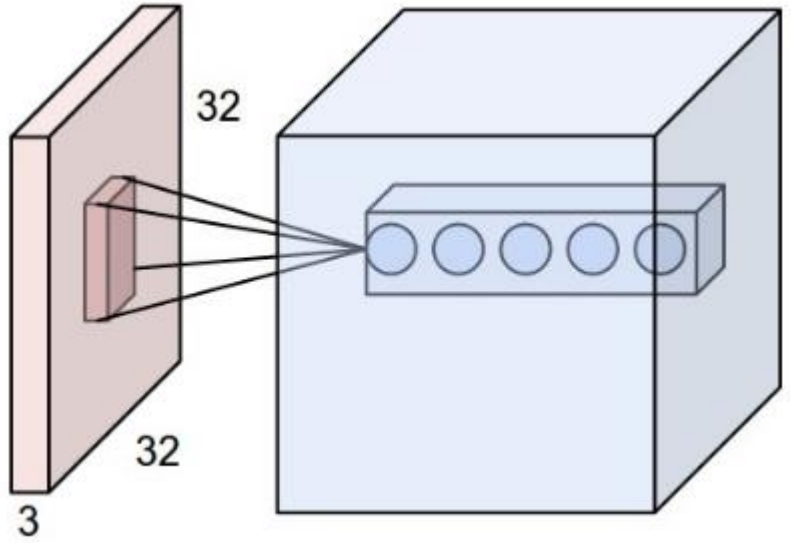


Βασική περιγραφή των Convolutional NN, από *Machine Learning and Medical Images, 2016* [26]

Το **input layer** δέχεται είσοδο ως ένα 2-διάστατο διάνυσμα, δηλ. ένα δισδιάστατο πίνακα. Αυτό έχει ως αποτέλεσμα καλύτερη εκμετάλλευση των χωρικών σχέσεων μεταξύ των pixels.

Convolutional Layers (1)

- Χρησιμοποιεί φίλτρα (kernels), τα οποία τομογραφούν την είσοδο μέσω συνεχών συνελίξεων (convolutions) για την ανακάλυψη τοπικών χαρακτηριστικών.
- Απαρτίζεται από τρεις κύριες παραμέτρους:
  - Number of filters
  - Kernel size
  - Padding



Παράδειγμα Input – Convolutional Layers, για μια εικόνα διαστάσεων : $(32x32x3)$. Πηγή: http://cs231n.github.io/convolutional-networks/

Convolutional Layers (2)

- Η εικόνα εισόδου αποτελείται από τρεις διαστάσεις: (μήκος, πλάτος, βάθος). Το βάθος αναφέρεται στα channels (1 για grayscale, 3 για rgb κ.α.)
- Τα στοιχεία του πίνακα είναι ακέραιοι συγκεκριμένου εύρους (π.χ. [0, 255]), που αντιπροσωπεύουν την ένταση του χρώματος.
- Ο kernel αποτελείται από ένα πίνακα συντελεστών, ο οποίος φιλτράρει τοπικά την εικόνα (εσωτερικό γινόμενο kernel και μέρος του πίνακα), μετατοπίζοντας κάθε φορά, οριζόντια ή κάθετα, κατά *stride* σε πλήθος pixels.
- Για k φίλτρα, προκύπτουν k αναπαραστάσεις (feature maps).

k_1	k_2
k_3	k_4

Kernel Shape: (2x2x1)

p_1	p_2	p_3	p_4	p_5
p_6	p_7	p_8	p_9	p_{10}
p_{11}	p_{12}	p_{13}	p_{14}	p_{15}
p_{16}	p_{17}	p_{18}	p_{19}	p_{20}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}

Input Image: (5x5x1)

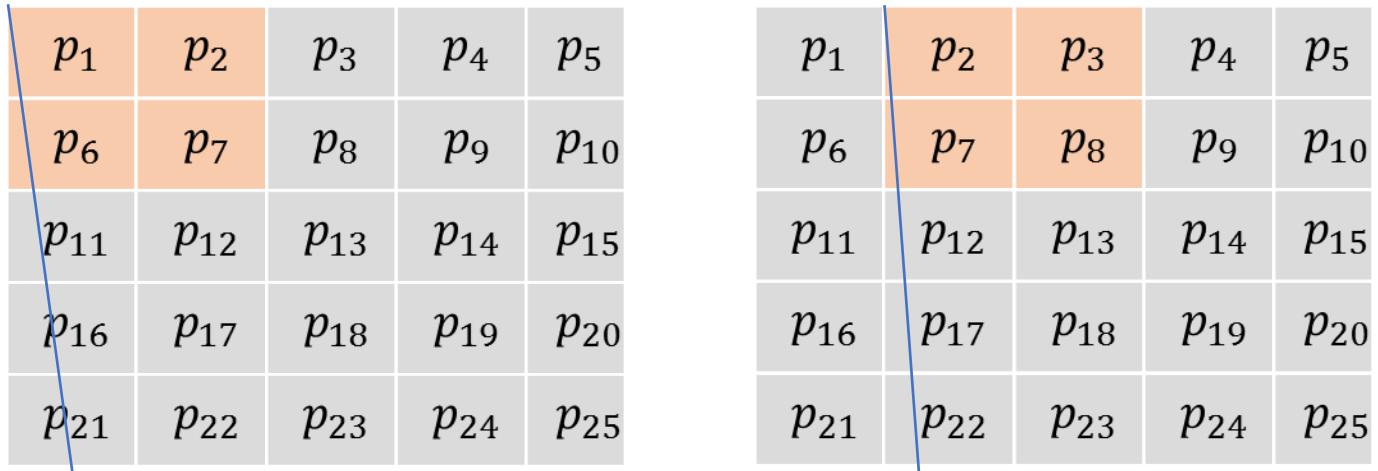
Convolutional Layers (3)

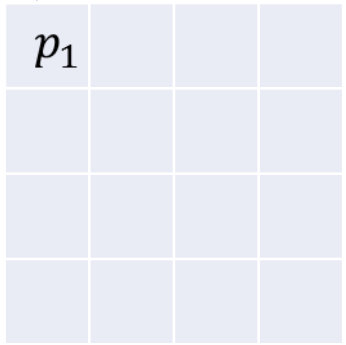
1	0
0	0

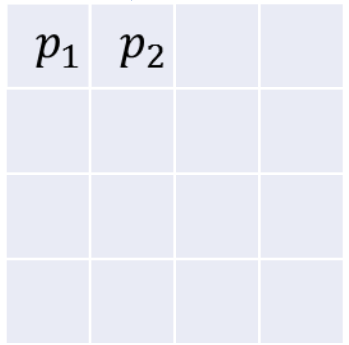
Kernel

p_1	p_2	p_3	p_4	p_5
p_6	p_7	p_8	p_9	p_{10}
p_{11}	p_{12}	p_{13}	p_{14}	p_{15}
p_{16}	p_{17}	p_{18}	p_{19}	p_{20}
p_{21}	p_{22}	p_{23}	p_{24}	p_{25}

...







Convolutions, παράδειγμα με stride = (1,1)

...

p_1	p_2	p_3	p_4
p_6	p_7	p_8	p_9
p_{11}	p_{12}	p_{13}	p_{14}
p_{16}	p_{17}	p_{18}	p_{19}

Convolutional Layers (3)

p1	p2	p3	p4	P5
p6	p7	p8	p9	p10
p11	p12	p13	p14	p15
p16	p17	p18	p19	p20
p21	p22	p23	p24	p25

1	0
0	0

Kernel

stride = (1,1) or
stride = 1

p1	p2	p3	p4
p6	p7	p8	P9
p11	p12	p13	p14
p16	p17	p18	p19

feature map

Convolutional Layers (3)

p1	p2	p3	p4	P5
p6	p7	p8	p9	p10
p11	p12	p13	p14	p15
p16	p17	p18	p19	p20
p21	p22	p23	p24	p25

1	0
0	0

Kernel

stride = (2,3)

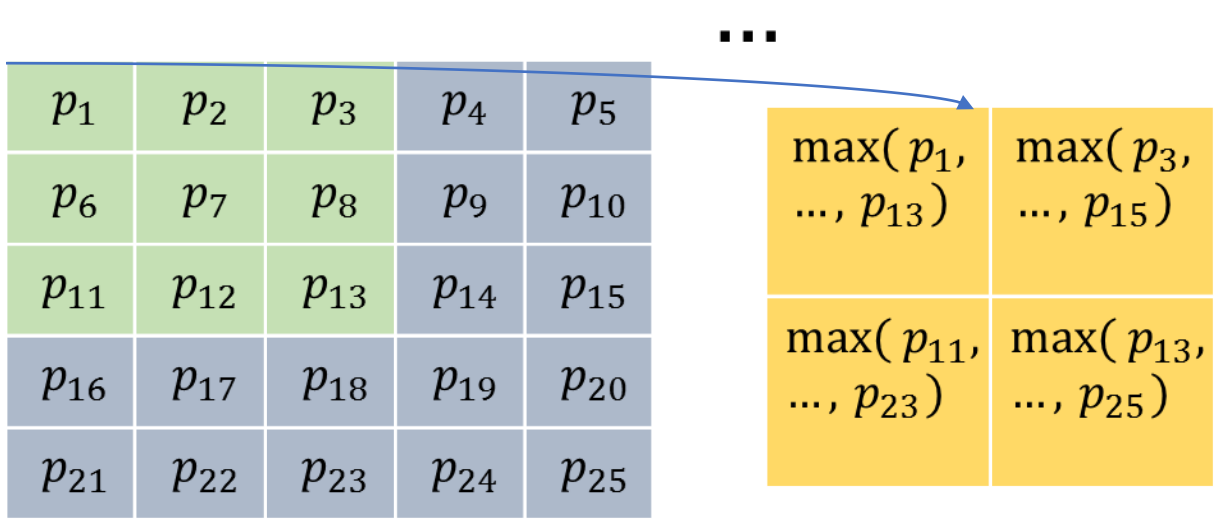
p1	p3
p16	p18

feature map

Pooling Layer

- **Χρήση:** Συνηθίζεται να τοποθετείται ανάμεσα από διαδοχικά convolutional layers.
- **Σκοπός:** Συντελεί στη μείωση των διαστάσεων των feature maps και στην καταπολέμηση του overfitting.
- **Περιγραφή:** Πραγματοποιείται επίσης μέσω της τοπικής εφαρμογής ενός φίλτρου, που μετατοπίζεται μέσω *stride*. Μπορεί να χαρακτηρισθεί ως ένα επιπλέον convolution, με μη γραμμικό kernel.

- **Είδη:** Τα κυριότερα είναι τα Average και Max. Συνηθίζεται max pooling, ενώ τα kernel size και stride ίσα με (2, 2) και 2.



Παράδειγμα για max pooling, με pool_size=(3, 3) και stride=2.

Padding

- Βοηθά στη ρύθμιση των διαστάσεων των output feature maps. Πραγματοποιείται πάντα μέσω μηδενικών.
- Χρησιμοποιείται τόσο στο Convolutional όσο και στο Pooling επίπεδο.
- Εφαρμόζεται ανάλογα με τις διαστάσεις του feature map που επιθυμούμε, ανάλογα με το stride. (no padding, half padding, same padding, full padding etc.) [28]

0	0	0	0	0	0	0
0	p_1	p_2	p_3	p_4	p_5	0
0	p_6	p_7	p_8	p_9	p_{10}	0
0	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	0
0	p_{16}	p_{17}	p_{18}	p_{19}	p_{20}	0
0	p_{21}	p_{22}	p_{23}	p_{24}	p_{25}	0
0	0	0	0	0	0	0

Simulation

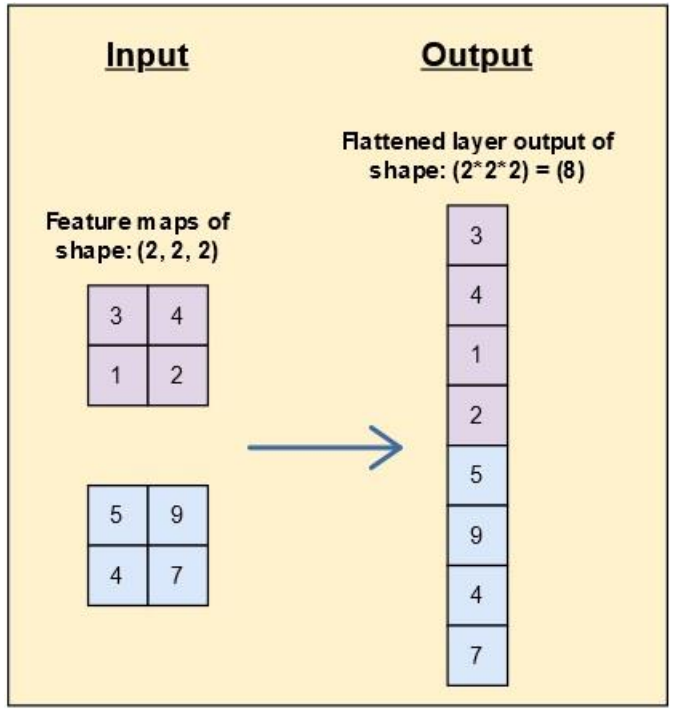
https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/tutorial.html

Flatten

- Το επίπεδο flatten βρίσκεται μεταξύ του CNN και του ANN και η δουλειά του είναι να μετατρέπει την έξοδο του CNN σε μια είσοδο που μπορεί να επεξεργαστεί το ANN.
- Το επίπεδο flatten χρησιμοποιείται για τη μετατροπή του χάρτη χαρακτηριστικών (feature map) που έλαβε από το επίπεδο max-pooling σε μια μορφή που μπορούν να κατανοήσουν τα πυκνά στρώματα (dense layers).
- Ένας χάρτης χαρακτηριστικών είναι ουσιαστικά ένας πολυδιάστατος πίνακας που περιέχει τιμές pixel. Τα πυκνά στρώματα απαιτούν μια μονοδιάστατη διάταξη ως είσοδο για επεξεργασία. Έτσι, το επίπεδο επίπεδο χρησιμοποιείται για την μετατροπή των χαρτών χαρακτηριστικών σε έναν μονοδιάστατο πίνακα για τα πυκνά στρώματα.

Flatten

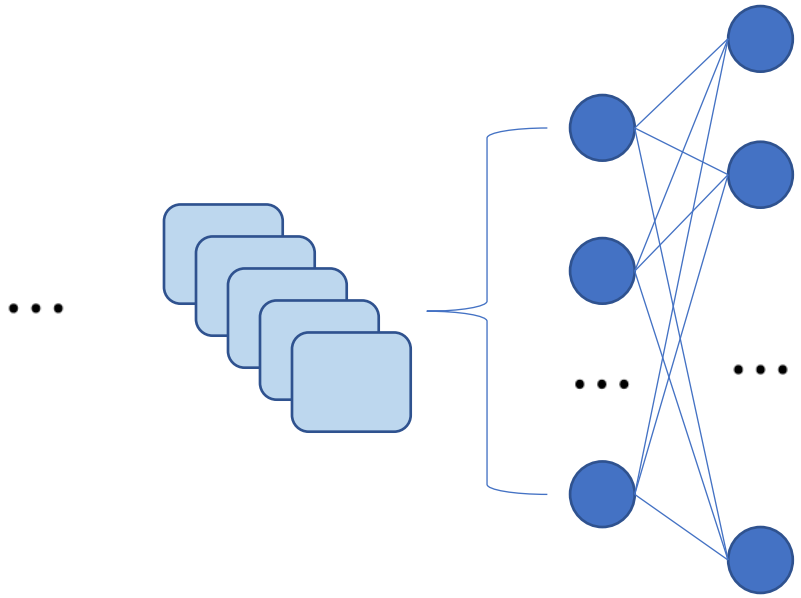
- Το διάγραμμα δείχνει ότι το επίπεδο flatten λαμβάνει χάρτες χαρακτηριστικών ως εισόδους από το επίπεδο max-pooling. Οι χάρτες χαρακτηριστικών μπορούν να έχουν τον τύπο (Ύψος, Πλάτος, Βάθος) όπου το Ύψος x Πλάτος αντιπροσωπεύει την πυκνότητα εικονοστοιχείων ενός χάρτη χαρακτηριστικών και το Βάθος δείχνει τον αριθμό των χαρτών χαρακτηριστικών.
- Στη συνέχεια, δρα επαναληπτικά πάνω από τους χάρτες χαρακτηριστικών και εισάγει τις τιμές των pixel μία προς μία σε έναν πίνακα που έχει μέγεθος ίσο με Ύψος x Πλάτος x Βάθος.



Πηγή: https://www.educative.io/answers/what-is-a-neural-network-flatten-layer

Fully Connected Layer & Output

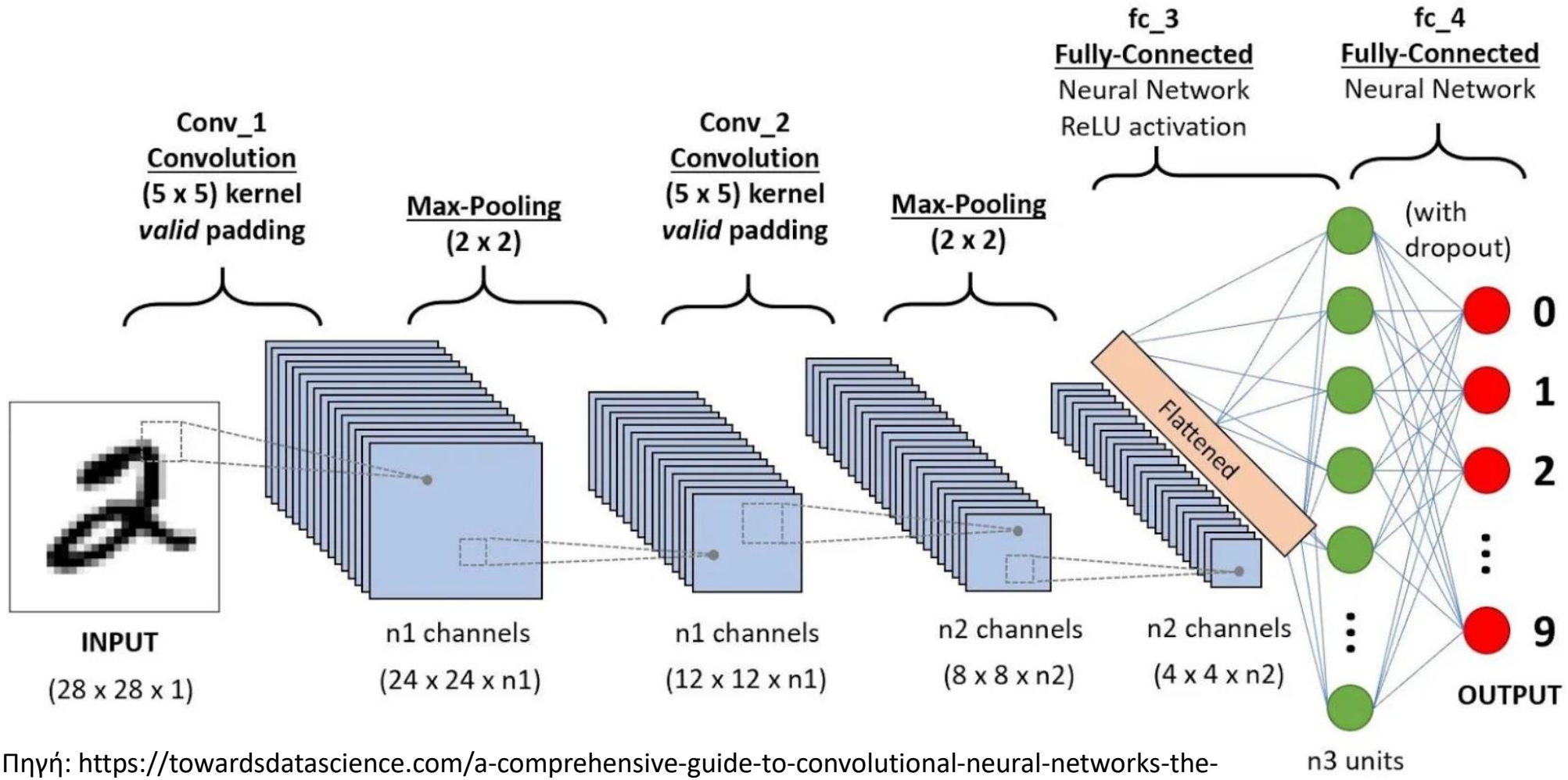
- Για την παραγωγή εξόδου:
  - το προϊόν των διαδοχικών Convolutional και Pooling επιπέδων δέχεται κατάλληλη διαμόρφωση διαστάσεων.
  - Συνηθίζεται να τροφοδοτείται σε ένα πλήρως συνδεδεμένο επίπεδο με κατάλληλη συνάρτηση ενεργοποίησης.
  - Εάν πρόκειται για classification task, η έξοδος παράγεται μέσω ενός τελικού Softmax επιπέδου.



Convolutional Neural Networks – Σημειώσεις

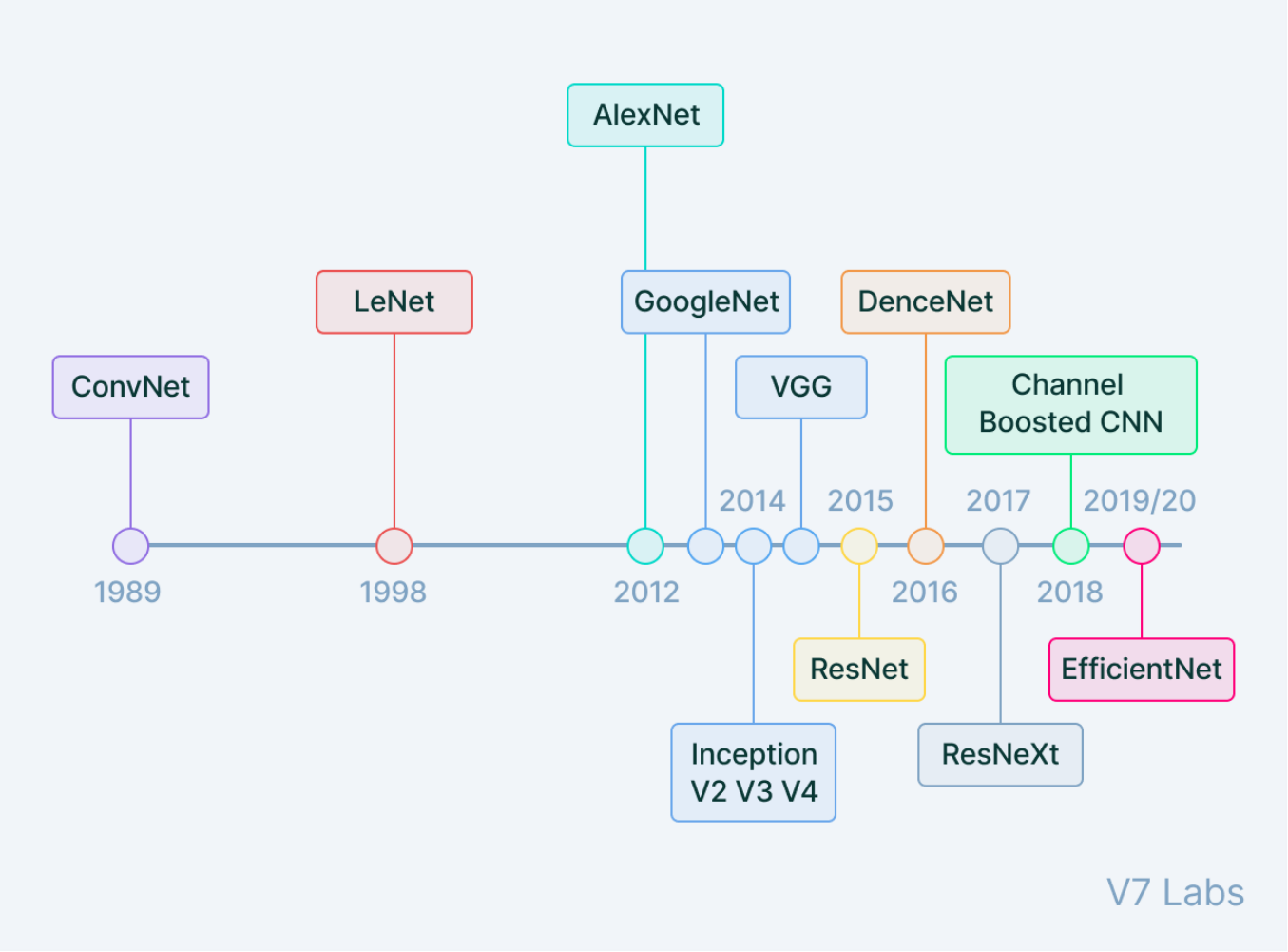
- Περισσότερα convolutional επίπεδα συντελούν στην ανακάλυψη και αφαίρεση προχωρημένων δομών και γνωρισμάτων των εικόνων. Αυξάνουν επίσης κατά πολύ την πολυπλοκότητα του δικτύου.
- Συνηθίζεται τα επίπεδα συνελίξεων να ακολουθούνται από Pooling επίπεδα, για τη μείωση τής διαστατικότητας, την αφαίρεση θορύβου και την αποφυγή overfitting. Δεν είναι αναγκαίο, αλλά στην κρίση του σχεδιαστή του δικτύου.
- Αν και συνηθίζεται, ούτε η χρήση του πλήρως συνδεδεμένου επιπέδου είναι αναγκαία. Συγκεκριμένα, στο state of the art μοντέλο ResNet, δε χρησιμοποιείται. Αντικαθίσταται με Global Average Pooling (βλ. [27], case studies).
- Γενικά προτιμάται η μέθοδος κανονικοποίησης Batch Normalization [31] αντί της Dropout στα Convolutional Neural Networks (βλ. [30]).

Παράδειγμα CNN



Πηγή: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Αρχιτεκτονικές CNN (1)



Πηγή:
https://www.v7labs.com
/blog/convolutional-
neural-networks-guide

Αρχιτεκτονικές CNN (2)

LeNet

This was the first introduced convolutional neural network. LeNet was trained on 2D images, grayscale images with a size of 32*32*1. The goal was to identify hand-written digits in bank cheques. It had **two convolutional-pooling layer blocks followed by two fully connected layers** for classification.

AlexNet

AlexNet was trained on the Imagenet dataset with 15 million high-resolution images with 256*256*3. ReLU activation function was used between convolution layers and pooling layers for the first time as well as the overlapping pooling with stride < window size. It had **five convolutional-pooling layer blocks followed by three fully connected dense layers** for classification.

Αρχιτεκτονικές CNN (3)

VGGNet

VGGNet came with a solution to improve performance rather than keep adding more dense layers in the model.

The key innovation came down to **grouping layers into blocks that were repetitively used** in the architecture because more layers of narrow convolutions were deemed more powerful than a smaller number of wider convolutions.

A **VGG-block** had a bunch of **3x3 convolutions padded by 1** to keep the size of output the same as that of input, **followed by max pooling to half the resolution**. The architecture had **n number of VGG blocks followed by three fully connected dense layers**

GoogLeNet

This architecture has **Inception blocks** that comprise **1x1, 3x3, 5x5 convolution layers followed by 3x3 max pooling with padding** (to make the output of the same shape as the input) on the previous layer and concatenates their output.

It has **22 layers**, **none** of which are **fully connected** layers. It requires a total of **4 million parameters** which is still 12 times fewer parameters than previous architectures like AlexNet.

Αρχιτεκτονικές CNN (3)

ResNet

It was observed that with the network depth increasing, the accuracy gets saturated and eventually degrades. Therefore, data scientists proposed a solution of **skip connections**.

These connections provide an alternate pathway for data and gradients to flow, make training fast, and enable skipping one or more layers. The idea of **residual blocks** was proposed which was based on the fact that deeper models should not produce higher training error than their shallow counterparts.

As a matter of fact, a deeper network was made from shallow networks by setting other layers in the deeper network to be identity mapping.

DenseNet

A limitation that was seen in ResNet was that of **vanishing gradients**. The key solution was to **create short paths** from early layers to later layers to train deep networks. All layers were connected directly to each other.

ZFNet

It is a modification of AlexNet. The major difference is that the architecture of ZFNet uses **7x7 filters**, whereas AlexNet uses 11x11 filters based on the thought that bigger filters might lead to loss of information. These changes proved effective and improved efficiency.

Convolutional Neural Networks – Χρήσιμοι Σύνδεσμοι

- Official Tensorflow Introduction to CNN: https://www.tensorflow.org/tutorials/images/cnn

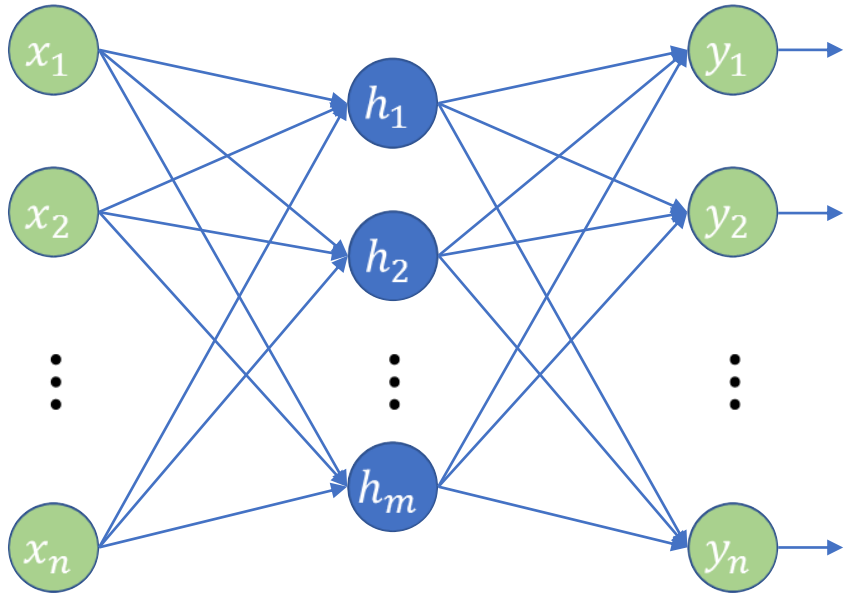
- Stanford CS231n: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks [27]

- "A Guide to Convolution Arithmetic for Deep Learning", [Dumoulin & Visin, 2018]: https://arxiv.org/pdf/1603.07285.pdf [28]

Autoencoders
(Αυτοκωδικοποιητές)

Autoencoders – “Unsupervised” Neural Networks (1)

- Εφαρμόζεται σε unlabeled δεδομένα.
- Οι διαστάσεις του επιπέδου εισόδου ισούνται με τις διαστάσεις του επιπέδου εξόδου.
- Εφαρμογές:
  - Μείωση της διαστατικότητας για data visualization (παρόμοια με PCA)
  - Denoising εικόνων ή ακολουθιών
  - Generative learning models
- Απλούστερη μορφή:
  - Feedforward
  - Fully Connected



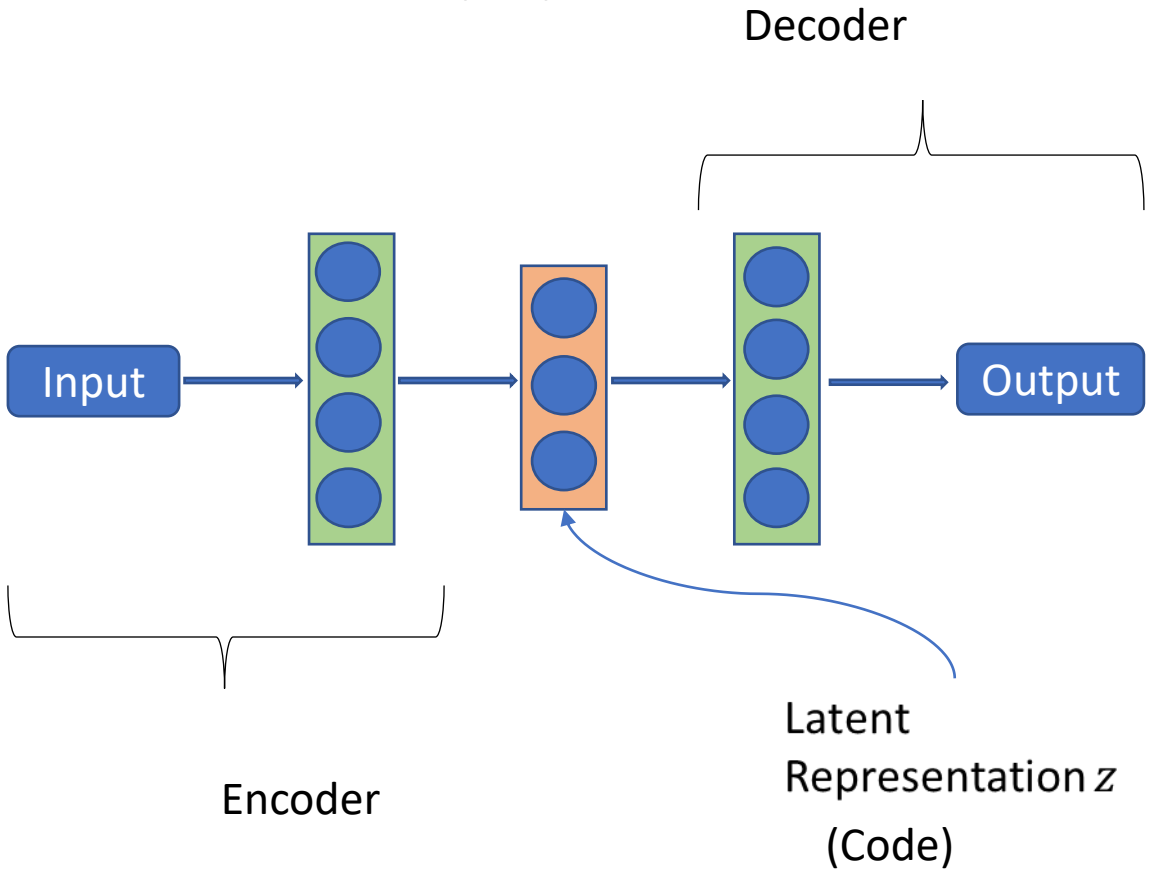
Autoencoders – “Unsupervised” Neural Networks (2)

- Σκοπός είναι η έξοδος $\vec{y}$ να ισούται με την είσοδο $\vec{x}$.

$$y = h_{W,b}(\vec{x}) \approx \vec{x}$$

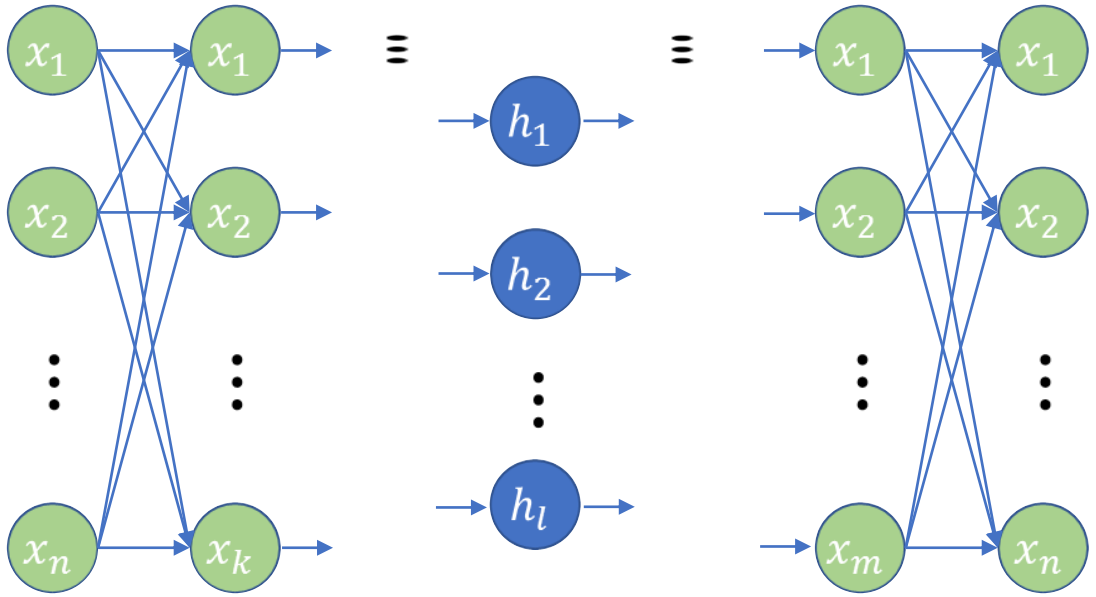
(Προσεγγίζει τη ταυτοτική συνάρτηση)

- Απαρτίζεται από δύο κύρια μέρη:
  - Encoder $\rightarrow$ Μαθαίνει τη νέα αναπαράσταση z.
  - Decoder $\rightarrow$ Ανακατασκευάζει τα δεδομένα από την αναπαράσταση z.



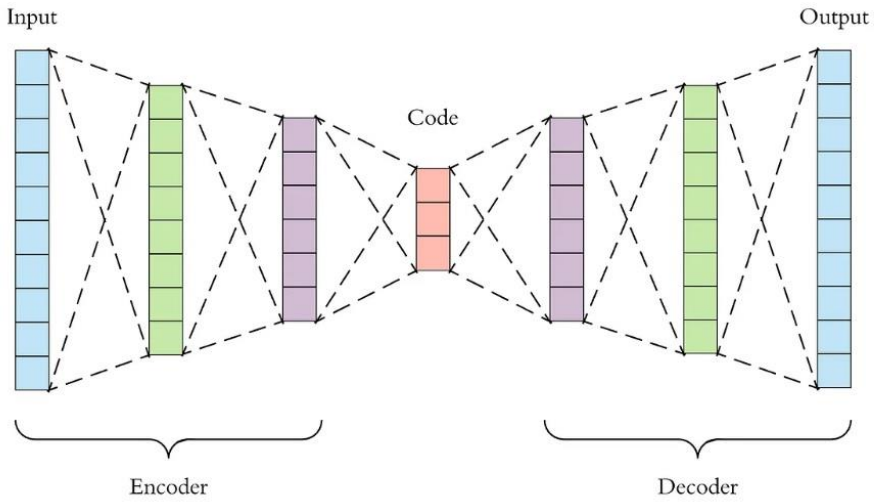
Autoencoders – “Unsupervised” Neural Networks (3)

- Μέσω περιορισμών (π.χ. διαφορετική διαστατικότητα), παράγονται διαφορετικές αναπαραστάσεις, $h_{W,b}(\vec{x})$, της εισόδου.
- Ανάλογα με το πρόβλημα που επιλύεται, οι encoder – decoder μπορούν να απαρτίζονται από DNN, LSTM, CNN και άλλα.
- Πραγματοποιείται μέσω backpropagation και Gradient Descent.



Autoencoders-Hyperparameters

- **Code size** (ο αριθμός νευρώνων του κώδικα)
- **Number of layers** (ο αριθμός των επιπέδων στον encoder και στον decoder)
- **Number of nodes per layer** (ο αριθμός των νευρώνων ανά επίπεδο)
- **Loss function** (Χρησιμοποιούμε είτε μέσο τετραγωνικό σφάλμα-mse είτε δυαδική διασταυρούμενη εντροπία- binary crossentropy.



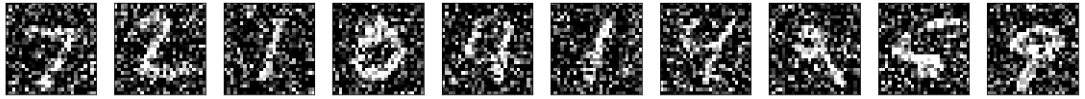
Οι αυτοκωδικοποιητές εκπαιδεύονται με τον ίδιο τρόπο όπως τα ANN μέσω backpropagation.

Τύποι Autoencoders

- Undercomplete Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Denoising Autoencoder
- Convolutional Autoencoder
- Variational Autoencoder

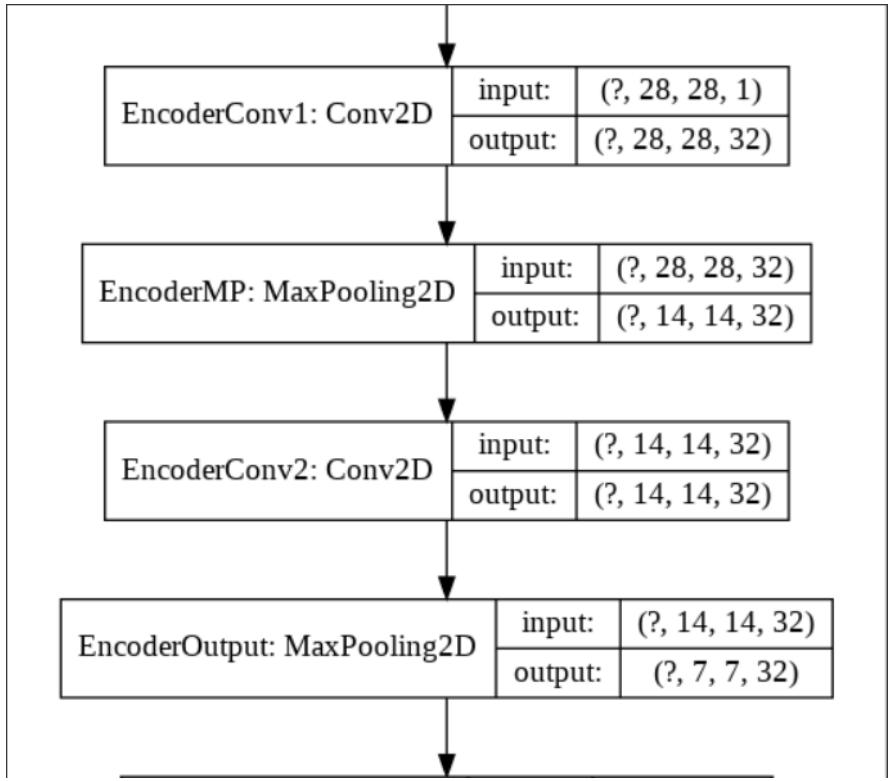
Παράδειγμα: Convolutional Autoencoder – Mnist Image Denoising [32] (1)

- **MNIST dataset** [32]**:** σύνολο χειρόγραφων grayscale εικόνων ακεραίων του εύρους [0, 9].
- Κάθε εικόνα αναπαρίσταται από έναν τανυστή διαστάσεων διαστάσεις: (28, 28, 1)
- Εισαγωγή θορύβου.
- Κατασκευή κατάλληλου Convolutional Autoencoder.

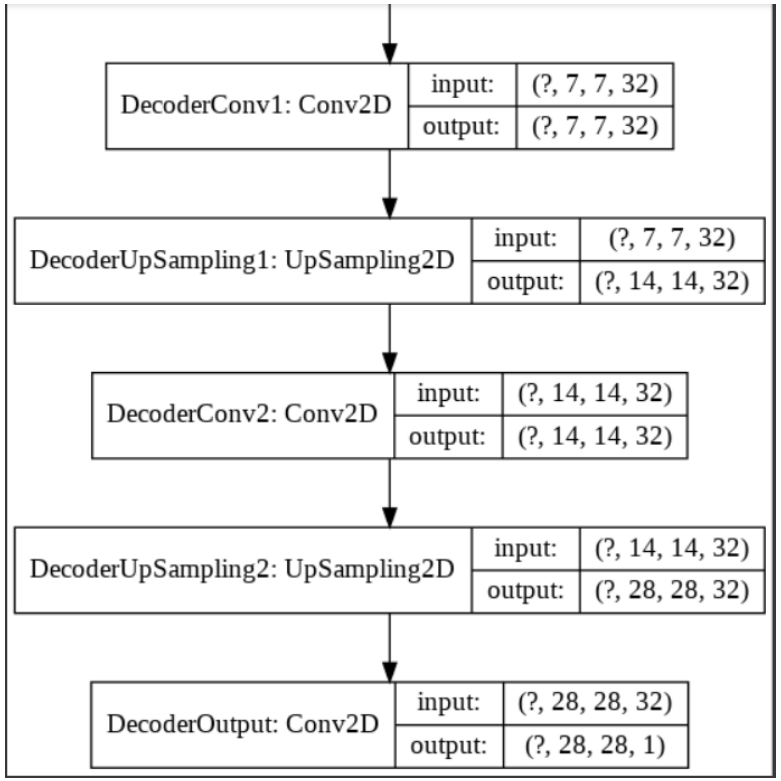


Παράδειγμα: Convolutional Autoencoder – Mnist Image Denoising [33] (2)

Encoder

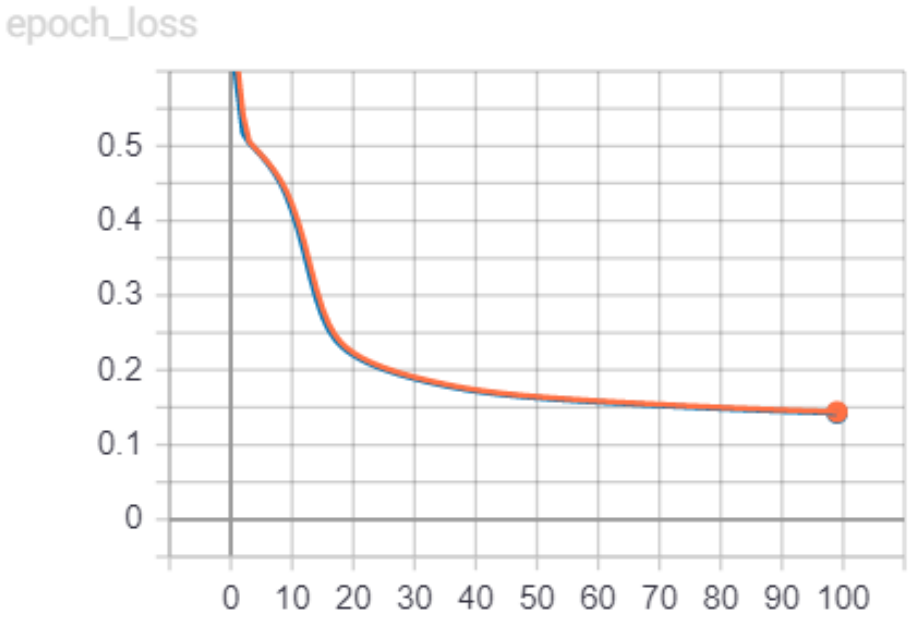


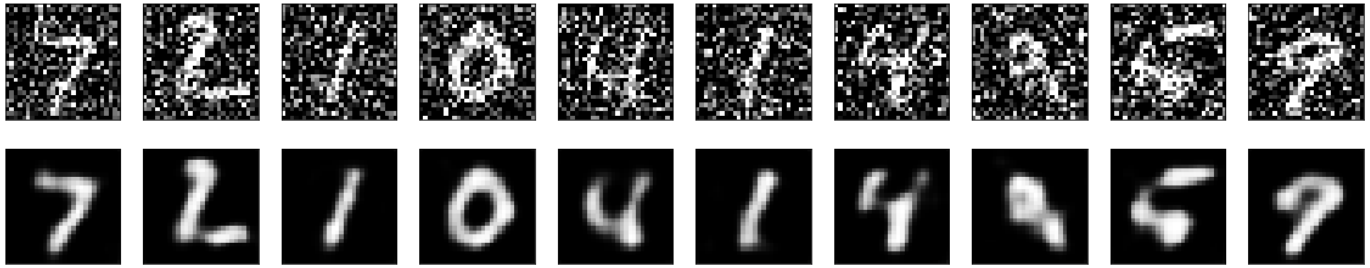
Decoder



Παράδειγμα: Convolutional Autoencoder – Mnist Image Denoising [33] (3)

- Εκπαίδευση του Convolutional Autoencoder:
  - Training set = (noisy training images, training images)
  - Validation set = (noisy validation images, validation images)
- Αποτελέσματα:





Autoencoders – Χρήσιμοι Σύνδεσμοι

- Deep Learning book, 2016, Goodfellow et al, Chapter 14, https://www.deeplearningbook.org/contents/autoencoders.html [34]
- Keras official tutorials on Autoencoders, 2016, Francois Chollet, https://blog.keras.io/building-autoencoders-in-keras.html[33]
- Sparse Autoencoders lecture notes, Andrew Ng, Stanford University https://web.stanford.edu/class/archive/cs/cs294a/cs294a.1104/sparseAutoencoder.pdf [35]
- "Auto – Encoding Variational Bayes", Kingma et al, 2014 https://arxiv.org/pdf/1312.6114.pdf [36]

References (1)

- [1]: “Deep Learning”, [Bengio, LeCun & Hinton, 2015]
- [2]: “Deep Learning in Neural Networks: an Overview”, [Schmidhuber, 2014]
- [3]: [Andrew Ng, Baidu Research Slides, ExtractConf 2015]
- [4]: “Geological Facies Prediction using Computed Tomography in a Machine Learning and Deep Learning Environment”, [Odi, Nguygen, 2018]
- [5]: “Dropout prediction in MOOC: using Deep Learning for Personalized Intervention”, [Xing, Du, 2018]
- [6]: “Adam: A method for stochastic optimization”, [Kingma & Ba, 2015]
- [7]: RMSprop, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
- [8]: “An overview of stochastic gradient descent optimization algorithms”, [Ruder, 2017]
- [9]: “Adaptive Subgradient Methods for Online Learning and Stochastic Oprimization”, [Duchi et al, 2011]
- [10]: https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/

References (2)

- [11]: “Incorporating Nesterov Momentum into Adam”, [Dozat, 2018]
- [12]: “On Loss Functions for Deep Neural Networks in Classification”, [Janocha & Czarnecki, 2017]
- [13]: https://github.com/keras-team/keras/blob/master/keras/losses.py
- [14]: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- [15]: “Learning Long – term Dependencies with Gradient Descent is Difficult”, [Bengio, 1993]
- [16]: “Long Short – Term Memory Units”, [Hochreiter & Schmidhuber, 1997]
- [17]: “Learning Phrase Representations using RNN Encoder –Decoder for Statistical Machine Translation”, [Cho et al, 2014]
- [18]: “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, [Srivastava et al, 2014]
- [19]: “Recurrent Neural Network Regularization”, [Zaremba et al, 2014]
- [20]: “Analysis of Gradient Clipping and Adaptive Scaling with a Relaxed Smoothness Condition”, [Zhang et al, 2019]

References (3)

- [21]: “Bidirectional Recurrent Neural Network”, [Schuster & Paliwal, 1997]
- [22]: https://keras.io/
- [23]: https://www.tensorflow.org/
- [24]: https://www.python.org/
- [25]: https://colab.research.google.com/
- [26]: “Machine Learning and Medical Images”, Wu et al, 2016
- [27]: http://cs231n.github.io/convolutional-networks/
- [28]: “A Guide to Convolution Arithmetic for Deep Learning”, [Dumoulin & Visin, 2018]
- [29]: Convolution Arithmetic Visualization (per [28]), https://github.com/vdumoulin/conv_arithmetic?source=post_page-----3bd2b1164a53----------
- [30]: https://www.kdnuggets.com/2018/09/dropout-convolutional-networks.html

References (4)

- [31]: “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, [Ioffe & Szegedy, 2015]
- [32]: http://yann.lecun.com/exdb/mnist/
- [33]: https://blog.keras.io/building-autoencoders-in-keras.html
- [34]: “Deep Learning Book”, [Goodfellow et al, 2016]
- [35]:https://web.stanford.edu/class/archive/cs/cs294a/cs294a.1104/sparseAutoencoder.pdf
- [36]: “Auto – Encoding Variational Bayes”, [Kingma et al, 2014]