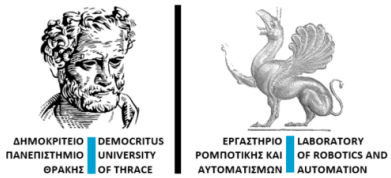
MatConvNet - An introduction



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

Σκοπός βιβλιοθήκης

Περιλαμβάνει ένα σύνολο κλάσεων, δομών και συναρτήσεων για την ανάπτυξη, εκπαίδευση και ελέγχου νευρωνικών δικτύων με την χρήση του matlab2016b, χωρίς την απαραίτητη χρήση κάρτας γραφικών (GPU).

Με τη βοήθεια της βιβλιοθήκης κάθε επίπεδο (layer) του δικτύου μπορεί να υλοποιηθεί απλά με τη χρήση της κατάλληλης συνάρτησης.

Παρέχεται δωρεάν στο παρακάτω link. Εμείς θα χρησιμοποιήσουμε την έκδοση **1.0-beta23**.

Χρήσιμα αρχεία

Ας περιηγηθούμε στον φάκελο **example**. Εκεί βρίσκονται τέσσερα **.m** αρχεία.

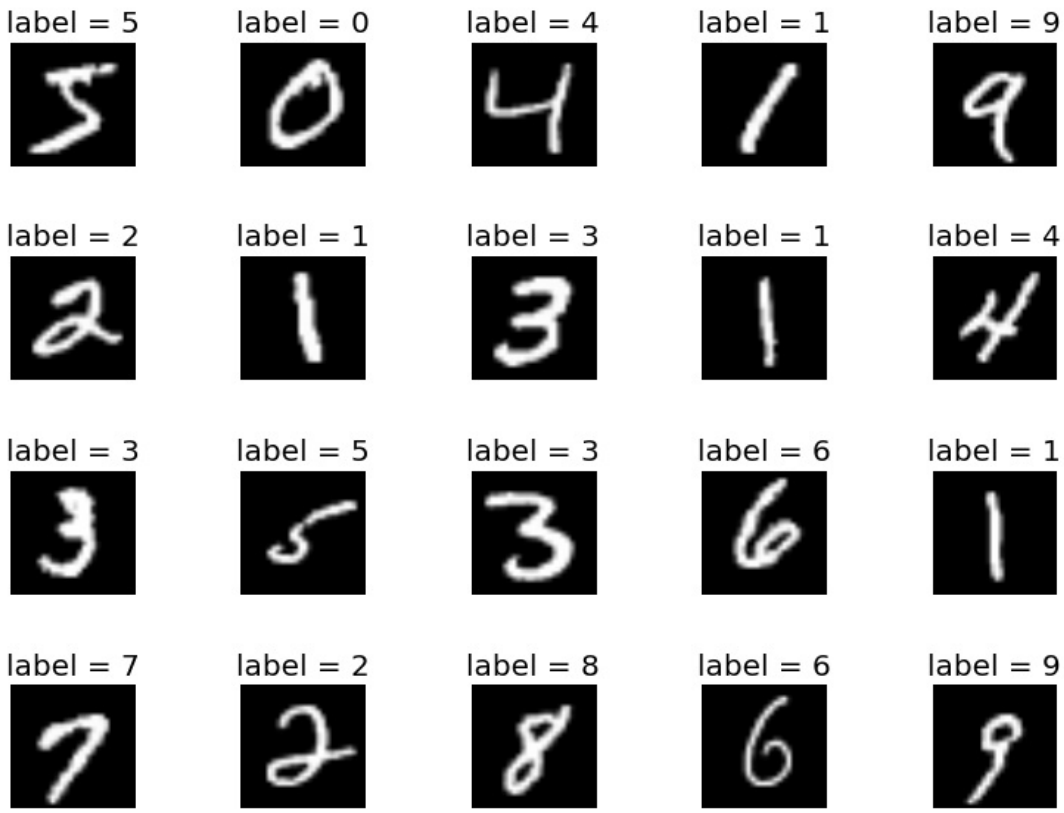
1. Το **run_experiment.m** είναι το αρχείο το οποίο εκτελούμε για να ξεκινήσουμε μία εκπαίδευση. Αυτό διαχειρίζεται τα άλλα τρία αρχεία που είναι συναρτήσεις.
2. Το **cnn.m** αναλαμβάνει την συλλογή των δεδομένων (training and testing data) και μεταπηδά από την μία διαδικασία στην άλλη.
3. Το **cnn_init.m** αρχικοποιεί ορισμένες παραμέτρους του δικτύου και της διαδικασίας εκπαίδευσής του.
4. Το **architecture.m** αποτελεί υποσυνάρτηση του **cnn_init.m**, καθώς ορίζει την αρχιτεκτονική του δικτύου και αρχικοποιεί τις παραμέτρους του κάθε επιπέδου (layer).

Σκοπός του παρόντος μαθήματος είναι ο πειραματισμός με την αρχιτεκτονική ενός δικτύου και επομένως θα επέμβουμε μόνο στο τελευταίο αρχείο.

Στο φάκελο **data/mnist-baseline** βρίσκεται η βάση δεδομένων (dataset) που θα χρησιμοποιήσουμε για την άσκηση.

MNIST Dataset

Περιλαμβάνει ένα σύνολο grayscale εικόνων από χειρόγραφα ψηφεία.



- image size: [28 x 28 x 1]
- training data: 60,000
- testing data: 10,000
- classes: 10

Σχεδιασμός νέας αρχιτεκτονικής

Ανοίγουμε το αρχείο **architecture.m**.

```
function layers = architecture()

    f=1/100 ;
    layers = {} ;

end
```

Παρατηρούμε μία μεταβλητή **f**. Λόγω της ReLU είναι καλό να αρχικοποιούμε τις παραμέτρους του δικτύου με τυχαίες μικρές τιμές. Για το λόγο αυτό ορίζουμε την τιμή f=1/100 που πολλαπλασιάζεται με την randn και επιστρέφει τυχαίες μικρές ποσότητες. Επίσης έχει αρχικοποιηθεί μία κενή δομή (struct) με όνομα layers.

Για να προσθέσουμε ένα νέο επίπεδο στο τέλος της δομής εργαζόμαστε ως εξής:

```
layers{end+1} = struct(...);
```

όπου μέσα στη δομή διευκρινίζονται το είδος του επιπέδου και οι παράμετροί του.

Layers and Activation Functions in MatConvNet

Convolutional layer

struct('type', 'conv', 'weights', {{f*randn(K_h,K_w,Channels,Depth, 'single'), zeros(1, Depth, 'single')}}, 'stride', S, 'pad', P);

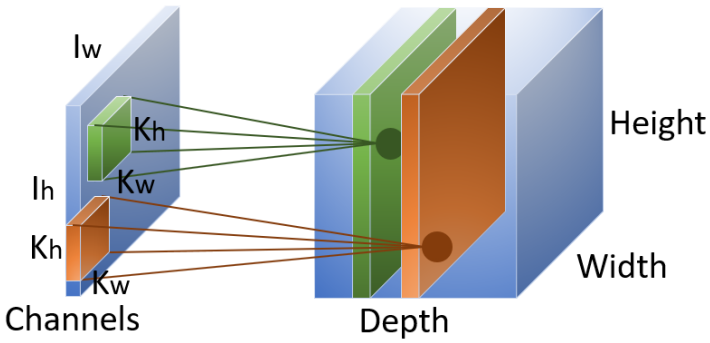
Θυμίζουμε ότι:

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

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

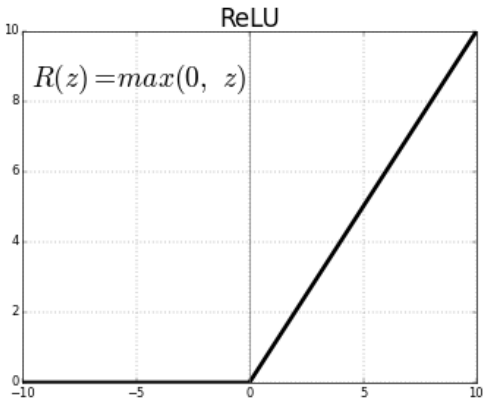
$Channels = Depth_of_previous_layer$

$Depth = Number_of_filters$



ReLU activation function

struct('type', 'relu');



MaxPool layer

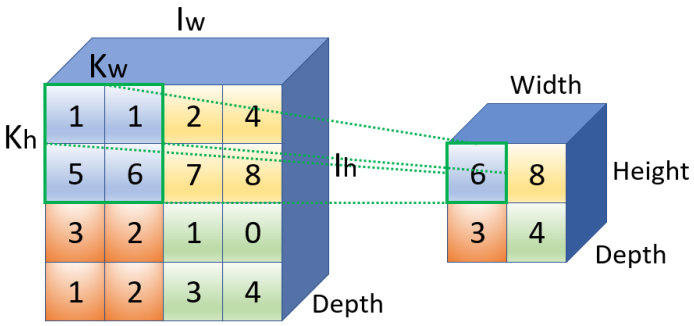
struct('type', 'pool', 'method', 'max', 'pool', [K_h, K_w], 'stride', S, 'pad', P);

Δεδομένου ότι $S=K_h=K_w$ και $P=0$,
θυμίζουμε ότι:

$Width = \frac{I_w}{K_w}$

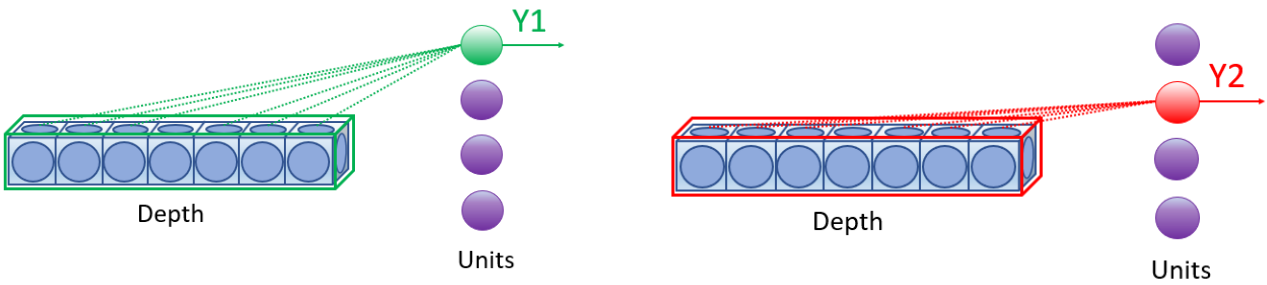
$Height = \frac{I_h}{K_h}$

$Depth = Depth_of_previous_layer$



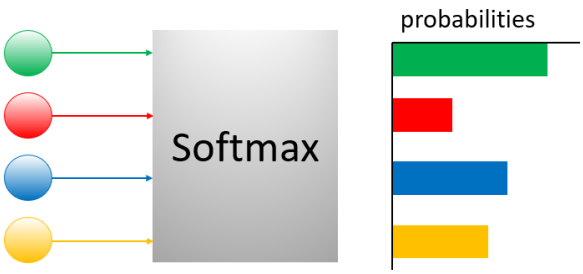
Dense layer

struct('type', 'conv', 'weights', {{f*randn(1,1,Channels,Units, 'single'), zeros(1,Units, 'single')}}, 'stride', 1, 'pad', 0);



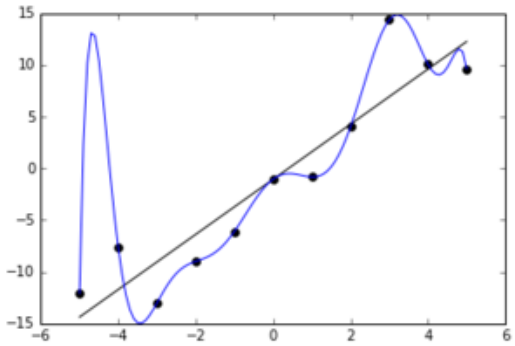
Softmax

struct('type', 'softmaxloss');



Training vs Validation Data

Αν η εκπαίδευση ενός δικτύου διαρκέσει για πολλές επαναλήψεις (**epochs**), παρατηρείται **overfitting** στα δεδομένα εκπαίδευσης. Το γεγονός αυτό είναι ανεπιθύμητο καθώς οταν δοθούν στο δίκτυο νέα δεδομένα δεν μπορεί να τα ξεχωρίσει αποδοτικά. Για το λόγο αυτό κατά τη διάρκεια της εκπαίδευσης, συνήθως μετά από κάθε epoch, εισάγουμε στο σύστημα ορισμένα δεδομένα που δεν έχουν συμπεριληφθεί στην εκπαίδευση για να δούμε πως τα πηγαίνει σε αυτά. Τα δεδομένα αυτά ονομάζονται **validation data**.



ΠΡΟΣΟΧΗ: Τα validation data τα εισάγουμε κάθε φορά απλά για να δούμε το σφάλμα του συστήματος **αλλά δεν διορθώνουμε τις παραμέτρους με βάσει αυτά**, παρα μόνο με τα δεδομένα εκπαίδευσης (training data).

Εγκατάσταση

- Αποσυμπιέζουμε το συμπιεσμένο αρχείο **matconvnet-1.0-beta23.tar**.
- Αντιγράφουμε το φάκελο **matconvnet-1.0-beta23** στο σημείο που βρίσκονται τα εγκατεστημένα αρχεία του matlab.
- Ανοίγουμε το matlab και πηγαίνουμε στο path μέσα στο φάκελο **matconvnet-1.0-beta23**.
- Εκτελούμε τις παρακάτω εντολές

Αν το Matlab ειναι εγκατεστημένο στα **Αρχεία Εφαρμογών**

```
>> mex -setup:'C:\Program Files\MATLAB\R2016b\bin\win64\mexopts\msvc2015.xml' C
>> mex -setup:'C:\Program Files\MATLAB\R2016b\bin\win64\mexopts\msvcpp2015.xml' C++
>> run matlab/vl_compilenn;
>> run matlab/vl_setupnn.m
```

Αν το Matlab ειναι εγκατεστημένο στην **Επιφάνεια Εργασίας**

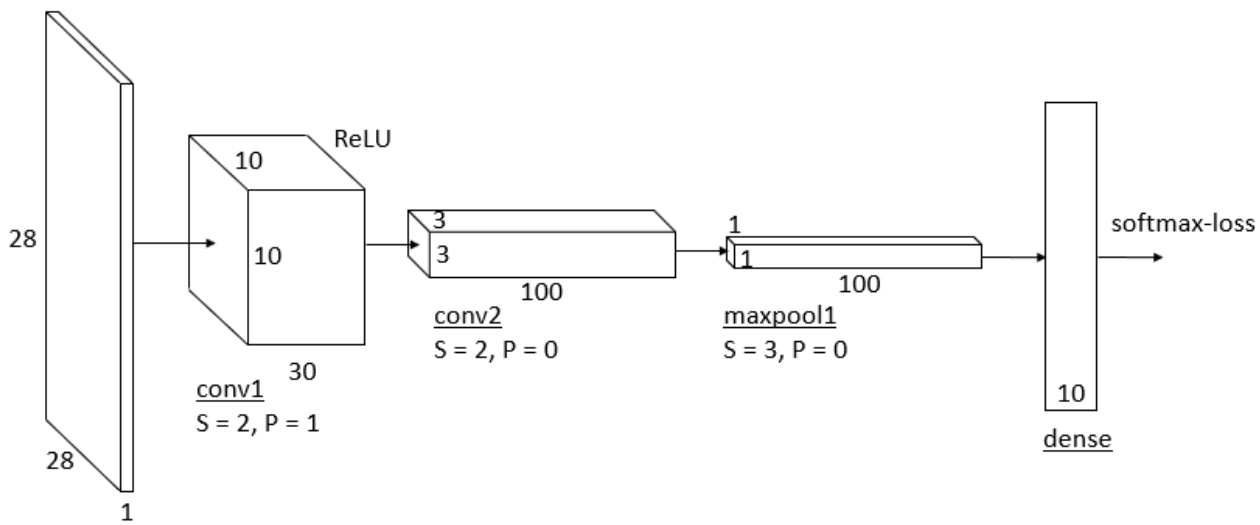
```
>> mex -
setup:'C:\Users\**Username**\Desktop\MATLAB\R2016b\bin\win64\mexopts\msvc2015.xml' C
>> mex -
setup:'C:\Users\**Username**\Desktop\MATLAB\R2016b\bin\win64\mexopts\msvcpp2015.xml'
C++
>> run matlab/vl_compilenn;
>> run matlab/vl_setupnn.m
```

Σε περίπτωση που είναι εγκατεστημένο σε άλλο μέρος πρέπει να βρείτε το path για να το συμπεριλάβετε στις δύο πρώτες εντολές.

```
>> mex -setup:'**PATH**\MATLAB\R2016b\bin\win64\mexopts\msvc2015.xml' C
>> mex -setup:'**PATH**\MATLAB\R2016b\bin\win64\mexopts\msvcpp2015.xml' C++
>> run matlab/vl_compilenn;
>> run matlab/vl_setupnn.m
```

Εφαρμογή - example

Να υλοποιηθεί το δίκτυο με την παρακάτω αρχιτεκτονική και να εκπαιδευτεί για 6 epochs.



Πρώτα φορτώνουμε το **imdb.mat** και ας εμφανίσουμε ορισμένα στοιχεία του (π.χ. το 1ο).

```
# Load imdb.mat and imshow
```

```
load(C:\...\example\data\mnist-baseline\imdb.mat)

imshow(images.data(:,:,1))
```

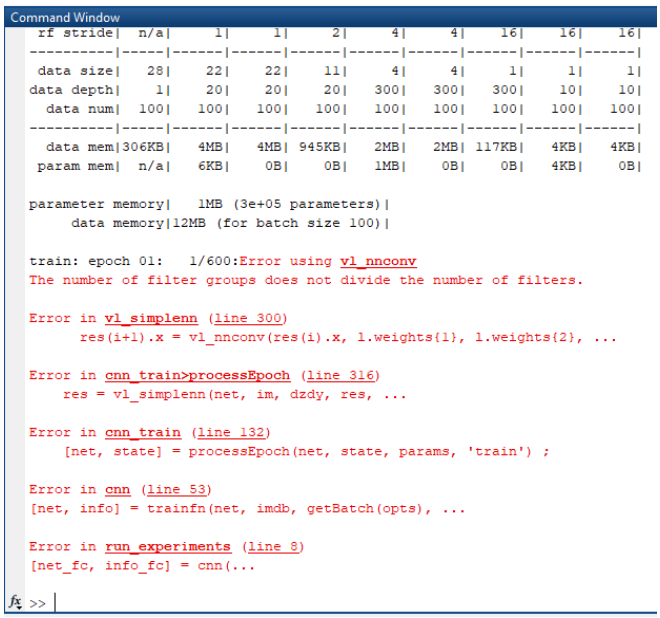
Έπειτα σχεδιάζουμε την αρχιτεκτονική προσθέτοντας **structs** στο **architecture.m** .

ΠΡΟΣΟΧΗ: Τα μεγέθη των φίλτρων πρέπει να υπολογιστούν βάσει των εξισώσεων που αναλύθηκαν παραπάνω.

Αποθηκεύουμε το αρχείο και εκτελούμε την εντολή:

```
>> run_experiments
```

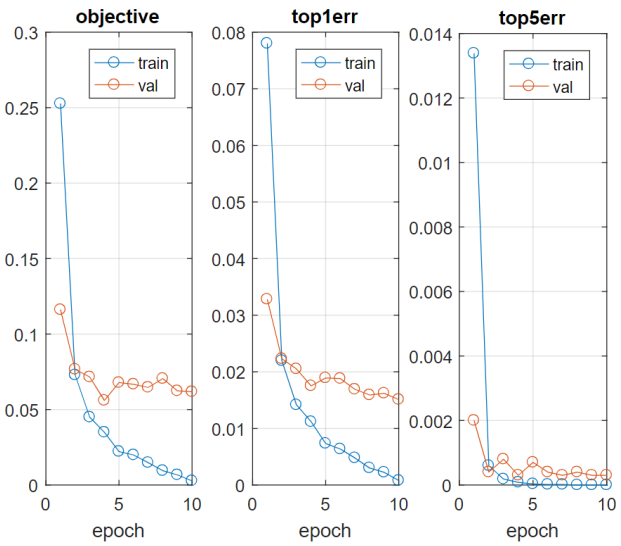
Πιθανά λάθη



Σφάλματα αρχιτεκτονικής δεν επιτρέπουν την εκκίνηση της εκπαίδευσης. Το σφάλμα εντοπίζεται στην **vl_nnconv** όπως φαίνεται δίπλα. Διαβάζουμε την πρώτη κόκκινη πρόταση. Μπορεί να οφείλεται σε:

- Μη συμβατό αριθμό των καναλιών του φίλτρου με το βάθος του προηγούμενου επιπέδου
- Λάθος διαστάσεις του φίλτρου
- Μη συμβατό αριθμό φίλτρων με αριθμό biases του επιπέδου
- κλπ

Αξιολογώντας την εκπαίδευση



Παρατηρούμε κυρίως τις **καμπύλες του loss** (**objective** στην matconvnet). Με μπλε απεικονίζονται οι καμπύλες των training data και με πορτοκαλί αυτές των validation data.

Το **top1err** υπολογίζει το ποσοστό των λάθος προβλέψεων ελέγχοντας στην έξοδο μόνο την πρόβλεψη του νευρώνα με την μεγαλύτερη τιμή. Αντιθέτως, το **top2err** υπολογίζει το ποσοστό των λάθος προβλέψεων θεωρώντας, ωστόσο, ως σωστή μία πρόβλεψη εφόσον στην έξοδο ο νευρώνας που αντιστοιχεί στην σωστή κλάση ανήκει στους πέντε νευρώνες με την μέγιστη τιμή.

Η απόδοση του δικτύου καθορίζεται από το **top1accuracy** των validation data, το οποίο υπολογίζεται ως:

$$top1accuracy(\%) = (1 - top1err) * 100\%$$

Όσο αυξάνεται η απόσταση του training και του validation loss, τόσο ενισχύεται το φαινόμενο του overfitting, με αποτέλεσμα σταδιακά το validation accuracy να μην βελτιώνεται άλλο. Αυτό πρακτικά σημαίνει ότι το δίκτυο παύει να "μαθαίνει" από τα training data χρήσιμες πληροφορίες για την επίλυση του προβλήματος που επιθυμούμε. Σε αυτήν την περίπτωση λέμε ότι το δίκτυο πάυει να γενικέυει.

Γενικότερα προσπαθούμε να σταματήσουμε την εκπαίδευση σε σημείο ούτως ώστε η απόσταση των δύο losses να μην έχει αυξηθεί υπερβολικά και το validation **top1accuracy** να είναι ικανοποιητικά υψηλό αναφορικά με το πρόβλημα που αντιμετωπίζουμε.

References

1] http://www.vlfeat.org/matconvnet/

2] https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec

3] https://en.wikipedia.org/wiki/Overfitting