



DEMOCRITUS  
UNIVERSITY  
OF THRACE

DEPARTMENT OF  
ELECTRICAL & COMPUTER  
ENGINEERING



# Όραση Υπολογιστών

Εργασία 4



Ον/μο Φοιτητή: Μπιτζίδης Χαράλαμπος

AM: 57424

## Περιεχόμενα:

- Σκοπός Εργασίας
- Παρουσίαση του σετ δεδομένων εκπαίδευσης
- Περιγραφή τεχνικής
- Μη προ-εκπαιδευμένο δίκτυο
  - Περιγραφή της αρχιτεκτονικής
  - Ανάλυση δικτύου
  - Ποσοτική εκτίμηση της επίδοσής τους στο σύνολο δοκιμής
  - Περιγραφή διαδικασία εκπαίδευσης
- Προ-εκπαιδευμένο δίκτυο
  - Περιγραφή της αρχιτεκτονικής
  - Ανάλυση δικτύου
  - Ποσοτική εκτίμηση της επίδοσής τους στο σύνολο δοκιμή
  - Περιγραφή διαδικασία εκπαίδευσης

# Σκοπός Εργασίας

Η συγκεκριμένη εργασία είναι η τέταρτη από μια σειρά εργασιών δόθηκαν στο πλαίσιο του μαθήματος Όραση Υπολογιστών και έχει ως στόχο την ταξινόμηση πολλαπλών κλάσεων (multi-class classification) χρησιμοποιώντας δεδομένα εικόνων. Για την επίλυση του προβλήματος, ζητείται να υλοποιηθεί αρχιτεκτονική συνελικτικού δικτύου σε Python με τη χρήση της βιβλιοθήκης **Keras-Tensorflow**.

Το σετ δεδομένων (dataset) αποτελείται από δύο βάσεις εικόνων, αυτές των φακέλων ‘**imagedb**’ και ‘**imagedb\_test**’, αντίστοιχα. Η πρώτη ‘**imagedb**’ θα χρησιμοποιηθεί για την εκπαίδευση του συστήματος και η δεύτερη ‘**imagedb\_test**’ για την δοκιμή και την αξιολόγησή του. Οι βάσεις αυτές αποτελούν υποσύνολο της βάσης εικόνων *Belgium Traffic Sign Dataset - Belgium TS Dataset*.

Στα πλαίσια της εργασίας έπρεπε να υλοποιηθούν και να υποβληθούν δύο αρχιτεκτονικές:

1. Ένα ΜΗ προ-εκπαιδευμένο δίκτυο το οποίο θα δημιουργύσταν αποκλειστικά για το τρέχον πρόβλημα ταξινόμησης, με τη διαδικασία που είχε αποτυπωθεί κατά το εργαστήριο του μαθήματος.
2. Ένα προ-εκπαιδευμένο δίκτυο προσωπικής επιλογής, που θα συνοδευόταν και από την αντίστοιχη ανάλυση του δικτύου που επιλέχθηκε.

Κάθε μια από τις παραπάνω αρχιτεκτονικές θα έπρεπε να συνοδευόταν από πλήρη περιγραφή της αρχιτεκτονικής και των επιπέδων που χρησιμοποιήθηκαν. Επιπλέον, θα έπρεπε να παρέχεται ποσοτική εκτίμηση της επίδοσής τους στο σύνολο δοκιμής, ως το ποσοστό επιτυχίας ταξινόμησης (accuracy). Τέλος, θα έπρεπε να περιγραφεί η διαδικασία εκπαίδευσης (εποχές, batch size, callbacks, προ-επεξεργασία, data augmentation, learning rate schedule).

## Παρουσίαση του σετ δεδομένων εκπαίδευσης

Το πρόβλημα προς επίλυση της παρούσας εργασίας ήταν η ταξινόμηση πολλαπλών κλάσεων (**multi-class classification**). Σκοπός του τελικού κώδικα ήταν να ταξινομήσει την κάθε εικόνα του σετ δοκιμής δεδομένων (**testing set**) σε μια από τις κλάσεις που υπήρχαν στα δεδομένα. Το σετ δεδομένων της παρούσας εργασίας, όπως αναφέρθηκε και παραπάνω ήταν ένα υποσύνολο του **Belgium TS Dataset**, το οποίο διαθέτει 62 κλάσεις. Τόσο το σετ εκπαίδευσης όσο και το σετ δοκιμής που δόθηκαν στο πλαίσιο της εργασίας αυτής αποτελούνται από 34 κλάσεις.

Το σετ **Belgium TS Dataset**, αποτελείται από εικόνες πινακίδων κυκλοφορίας. Οι εικόνες αυτές είναι σε μορφή **.ppm** (portable pixmap) και όχι στα συνηθισμένα format **.jpg** ή/και **.png**. Τα αρχεία PPM είναι χρωματικές εικόνες 24 bit σε μορφή κειμένου. Κάθε ένα τέτοιο αρχείο κειμένου αποθηκεύει την τιμή όλων των εικονοστοιχείων με έναν αριθμό από το 0 έως το 65536, δηλαδή το χρώμα του. Τα αρχεία PPM επίσης αποθηκεύουν τις διαστάσεις της εικόνας (ύψος, πλάτος) που αποθηκεύουν καθώς και άλλα δεδομένα όπως whitespace data και την τιμή του μέγιστου χρώματος (maximum color value).

Οσον αφορά το σετ δεδομένων δημιουργήθηκε ένα παραπάνω colab notebook, προκειμένου να γίνει μια σύντομη παρουσίαση του συγκεκριμένου υποσυνόλου του **Belgium TS Dataset**. Στο notebook αυτό, απλά φορτώνεται το σετ δεδομένων σε έναν τοπικό φάκελο στα προσωρινά αρχεία του colab και αφού χωριστεί σε σετ εκπαίδευσης και σετ δοκιμής, χρησιμοποιείται μια συνάρτηση (*load\_data*) που απλώς αποθηκεύει όλα τα αρχεία PPM σε εικόνες οι οποίες μπορούν να απεικονιστούν στο colab. Συγκεκριμένα επιστρέφει μια λίστα με τις εικόνες και μια λίστα με τα *labels* των εικόνων αυτών, καθώς επίσης δέχεται ως είσοδο τον φάκελο δεδομένων (είτε του σετ εκπαίδευσης είτε του σετ δοκιμής). Τα παραπάνω πραγματοποιήθηκαν με την βοήθεια τόσο της βιβλιοθήκης **skimage**, για το διάβασμα των αρχείων PPM, όσο και της βιβλιοθήκης **matplotlib** για την απεικόνισή τους. Αυτό γίνεται χωριστά για το σετ εκπαίδευσης και για το σετ δοκιμής.

Παρακάτω φαίνονται και ο αριθμός των εικόνων που υπολογίστηκε για τα δύο σετ δεδομένων:

```
Training set :  
Unique Labels: 34  
Total Images: 3056
```

Testing set :  
 Unique Labels: 34  
 Total Images: 2149

Στην συνέχεια χρησιμοποιείται μια άλλη συνάρτηση (*display\_classes*) η οποία παρουσιάζει όλες τις κλάσεις του σετ δεδομένων, απεικονίζοντας σε ένα πλέγμα την πρώτη εικόνα όλων των κλάσεων.

Παρακάτω φαίνονται τα αποτελέσματα της συνάρτησης *display\_classes* για τα δύο σετ δεδομένων:

#### Για το σετ δεδομένων εκπαίδευσης:



#### Για το σετ δεδομένων δοκιμής:



### Σχόλια - Παρατηρήσεις:

- Βεβαίως και οι κατηγορίες πινακίδων κυκλοφορίας είναι οι ίδιες στα δύο σετ. Με άλλα λόγια έχουμε τα ίδια labels στα δύο σετ δεδομένων.
- Δίπλα από τον αριθμό του label της κάθε κατηγορίας πινακίδας κυκλοφορίας βρίσκεται και το πλήθος των εικόνων που έχουν αυτό το label. Για παράδειγμα στο σετ δεδομένων εκπαίδευσης για την ρυθμιστική πινακίδα της υποχρεωτικής διακοπής πορείας (STOP) υπάρχουν 43 εικόνες που την απεικονίζουν.

Αυτό φαίνεται και παρακάτω:



Στην συνέχεια χρησιμοποιούνται κάποιες άλλες συνάρτησεις (`display_label_images_train()` για την εκπαίδευση και `display_label_images_test()` για την δοκιμή) οι οποίες παίρνουν ως είσοδο την λίστα των εικόνων `images` και ένα συγκεκριμένο αριθμό `label`. Ο σκοπός αυτών

των συναρτήσεων είναι να παρουσιάσουν κάποια δείγματα εικόνων των σετ δεδομένων για ένα συγκεκριμένο label.

Παρακάτω φαίνονται τα αποτελέσματα των δειγμάτων του **σετ εκπαίδευσης** για κάποιες περιπτώσεις label:

- Για την πινακίδα αναγγελίας κινδύνου λόγω εκτελούμενων εργασιών στην οδό:

Label 10 (21)



Οι εικόνες δείγματα φαίνονται παρακάτω:



- Για την πινακίδα αναγγελίας κινδύνου λόγω δύο διαδοχικών στροφών:

Label 5 (11)



Οι εικόνες δείγματα φαίνονται παρακάτω:

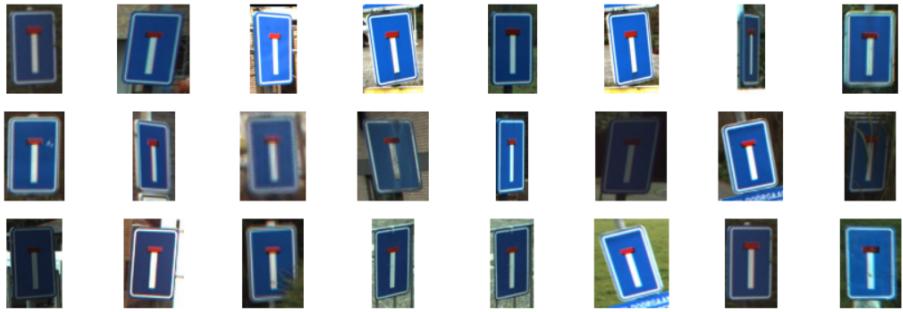


- Για την πληροφοριακή πινακίδα για αδιέξοδο:

Label 54 (118)



Οι εικόνες δείγματα φαίνονται παρακάτω:



Παρακάτω φαίνονται τα αποτελέσματα των δειγμάτων του **σετ δοκιμής** για κάποιες περιπτώσεις label:

- Για την ρυθμιστική πινακίδα “Απαγορεύεται η δεξιά στροφή”:

Label 30 (37)



Οι εικόνες δείγματα φαίνονται παρακάτω:



- Για την ρυθμιστική πινακίδα του μέγιστου ορίου ταχύτητας:

Label 32 (422)



Οι εικόνες δείγματα φαίνονται παρακάτω:



- Για την πληροφοριακή πινακίδα του χώρου επιτρεπόμενης στάθμευσης:

Label 45 (84)



Οι εικόνες δείγματα φαίνονται παρακάτω:

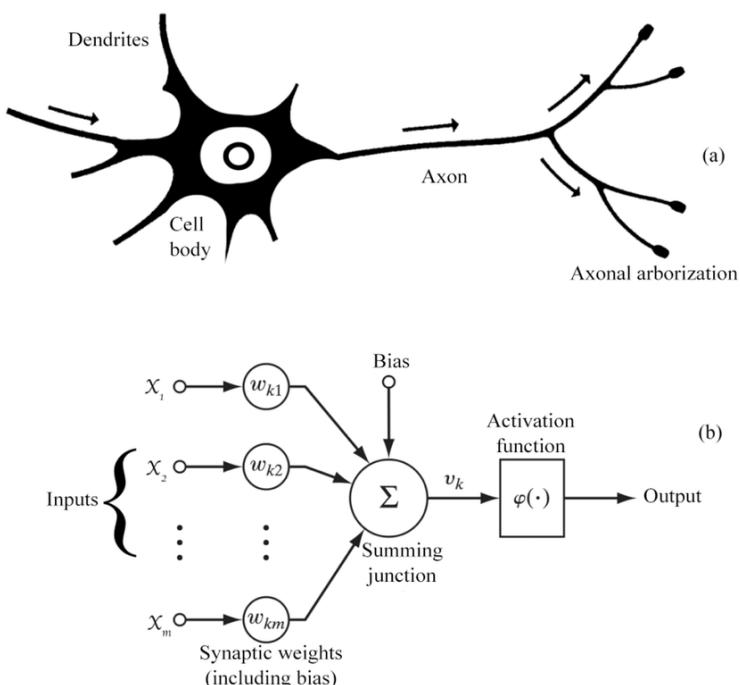


# Περιγραφή τεχνικής

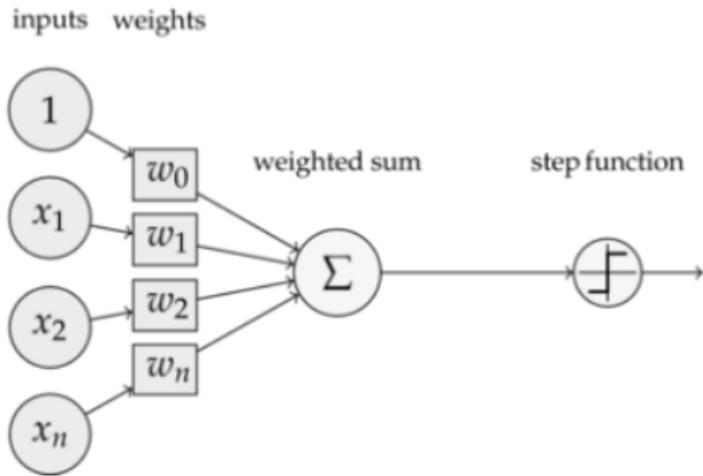
Η τεχνική που χρησιμοποιήθηκε σε αυτήν την εργασία είναι η βαθιά μάθηση (**Deep Learning**) και τα νευρωνικά δίκτυα (**Neural Networks**). Συγκεκριμένα για το πρόβλημα της ταξινόμησης πολλαπλών κλάσεων, χρησιμοποιήθηκε ένας συγκεκριμένος τύπος νευρωνικών δικτύων, αυτός των συνελικτικών (**CNN - Convolutional Neural Networks**). Αυτός ο τύπος νευρωνικών δικτύων είναι βασισμένος στην πράξη της συνέλιξης πινάκων. Παρακάτω ακολουθεί μια παρουσίαση των νευρωνικών δικτύων ξεκινώντας από απλούστερες δομές και καταλήγει στα συνελικτικά και τον τρόπο λειτουργίας τους.

## Η δομή Perceptron:

Η δομή Perceptron είναι η απλούστερη μορφή νευρωνικών δικτύων. Είναι ένας αλγόριθμος Μηχανικής Μάθησης με Επίβλεψη (**Supervised Machine Learning**) για δυαδική ταξινόμηση (**binary classification**). Είναι ένας τύπος γραμμικού ταξινομητή (**linear classifier**) και κάνει τις προβλέψεις βασιζόμενος σε μια γραμμική συνάρτηση πρόβλεψης. Το δίκτυο Perceptron ονομάζεται αλλιώς και νευρώνας (**neuron**), διότι έχει εμπνευστεί από τους βιολογικούς νευρώνες.



Παρακάτω απεικονίζεται το διάγραμμα ενός δικτύου Perceptron:



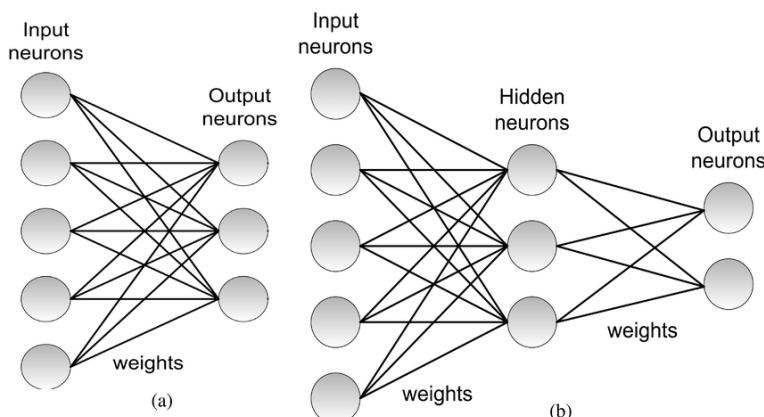
Τα βασικά στοιχεία του δικτύου Perceptron είναι:

- Το διάνυσμα εισόδου, το οποίο αποτελείται από η τιμές. Λέγεται αλλιώς και διάνυσμα χαρακτηριστικών (**feature vector**).
- Ο πίνακας των βαρών. Τα βάρη (**weights**) είναι οι τιμές οι οποίες στα νευρωνικά δίκτυα υπολογίζονται στην διαδικασία της εκπαίδευσης.
- Η τιμή bias, η οποία είναι μία παράμετρος που δεν επηρεάζεται από κάποιο βάρος. Είναι μία τιμή ειδικής εισόδου.
- Η συνάρτηση ενεργοποίησης (**activation function**), η οποία καθορίζει το αποτέλεσμα του δικτύου Perceptron. Ως είσοδο παίρνει το άθροισμα του διανύσματος εισόδου αφού προηγουμένως έχει πολλαπλασιαστεί αρχικά κατά στοιχείο με το διάνυσμα βαρών και έπειτα έχει προστεθεί στο αποτέλεσμα η τιμή του bias. Ως έξοδο η συνάρτηση ενεργοποίησης παράγει την πρόβλεψη γραμμικής ταξινόμησης του Perceptron. Ένας νευρώνας Perceptron μπορεί να λύσει μόνο γραμμικώς διαχωρίσιμα προβλήματα, επειδή η συνάρτηση ενεργοποίησης διαχωρίζει μόνο γραμμικά προβλήματα.

## Η δομή των Τεχνητών Νευρωνικών Δικτύων (Artificial Neural Networks):

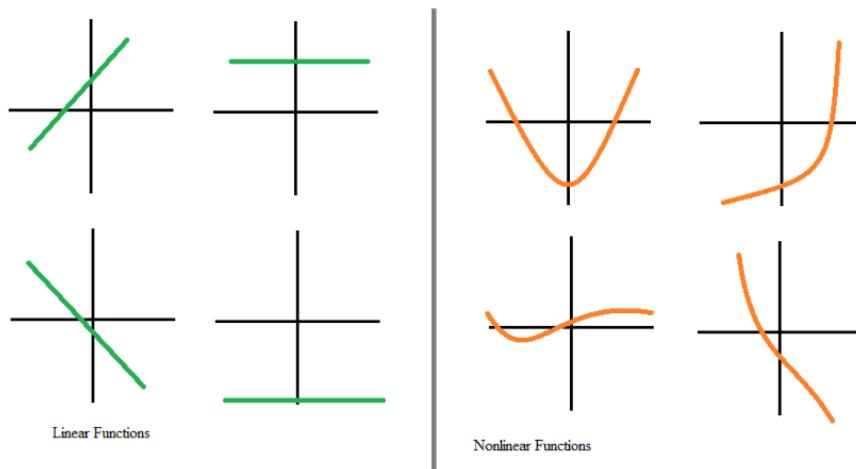
Τα νευρωνικά δίκτυα είναι δίκτυα το οποία αποτελούνται από νευρώνες. Οι νευρώνες συνδέονται σε επίπεδα (**layers**). Τα νευρωνικά δίκτυα μπορεί να έχουν ένα ή και παραπάνω επίπεδα από νευρώνες. Στην περίπτωση του ενός layer τα δίκτυα ονομάζονται **Single Layer Perceptrons**, ενώ στην περίπτωση των πολλαπλών layers, τα δίκτυα ονομάζονται **Multilayer Perceptrons**. Τα Multi Layer Perceptrons αποτελούνται από πολλά ενδιάμεσα επίπεδα νευρώνων, τα οποία ονομάζονται κρυφά (**hidden layers**) επειδή δεν επικοινωνούν με την είσοδο ή με την έξοδο του δικτύου.

Παρακάτω φαίνονται ένα Single Layer Perceptron και ένα Multi Layer Perceptron:



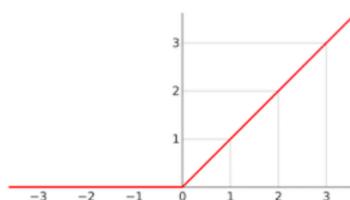
Τα Single Layer Perceptrons (SLP) μπορούν να λύσουν μόνο γραμμικώς διαχωρίσιμα προβλήματα. Σε αντίθεση με αυτά, τα Multilayer Perceptrons (MLP), μπορούν να λύσουν τόσο γραμμικώς, όσο και μη γραμμικώς διαχωρίσιμα προβλήματα. Ο λόγος για την ικανότητα αυτή των Multilayer Perceptrons είναι οι περισσότερο πολύπλοκες συναρτήσεις ενεργοποίησης που διαθέτουν. Ενώ τα Single Layer Perceptrons διαθέτουν συναρτήσεις ενεργοποίησης που διαχωρίζουν μόνο γραμμικά προβλήματα, τα Multilayer Perceptrons διαθέτουν πιο σύνθετες όπως ReLU(Rectified Linear Unit), Sigmoid, Tanh.

Παρακάτω φαίνονται τα διαγράμματα κάποιων συναρτήσεων ενεργοποίησης ενός SLP σε σχέση με αυτές των MLP:

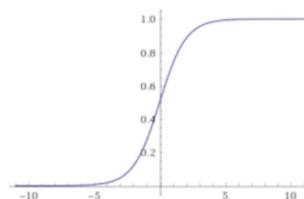


Παρακάτω φαίνονται τα διαγράμματα για συγκεκριμένες συναρτήσεις ενεργοποίησης:

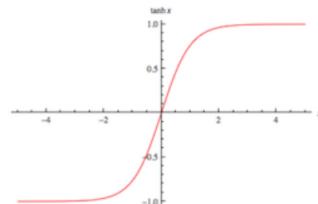
- **Rectified Linear Unit:**



- **Sigmoid:**



- **Tanh:**

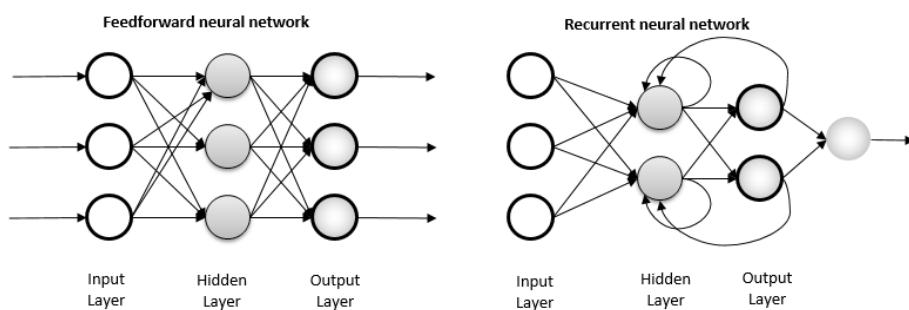


Στα τεχνητά νευρωνικά δίκτυα κάθε νευρώνας, δηλαδή κάθε δομή Perceptron, λειτουργεί ακριβώς όπως περιγράφηκε παραπάνω στην δομή Perceptron. Κάθε νευρώνας ενός επιπέδου ενώνεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Αυτός ο τρόπος σύνδεσης καθορίζει και το επίπεδο ως πλήρως συνδεδεμένο (**fully connected layers** ή **dense layers**). Οι νευρώνες του επιπέδου εισόδου έχουν ως είσοδο το διάνυσμα εισόδου

και μια τιμή bias. Οι νευρώνες κάθε κρυφού επιπέδου (για τα Multilayer Perceptrons) έχουν ως είσοδο ένα διάνυσμα που αποτελείται από τις εξόδους όλων των νευρώνων του προηγούμενου επιπέδου. Οι νευρώνες εξόδου έχουν ως έξοδο τις τιμές που προβλέπει το δίκτυο για το κάθε πρόβλημα.

Ένας άλλος διαχωρισμός των τεχνητών νευρωνικών δικτύων είναι με βάση την κατεύθυνσή τους. Χωρίζονται σε προσω τροφοδοτούμενα (**feedforward**) και ανατροφοδοτουμενα (**feedback**). Τα προσω τροφοδοτούμενα μεταφέρουν τα δεδομένα από την είσοδο στην έξοδο. Τα ανατροφοδοτούμενα νευρωνικά δίκτυα έχουν μονοτάτια ανάδρασης και μπορούν να μεταφέρουν τα δεδομένα και από την είσοδο προς την έξοδο αλλά και από την έξοδο προς την είσοδο.

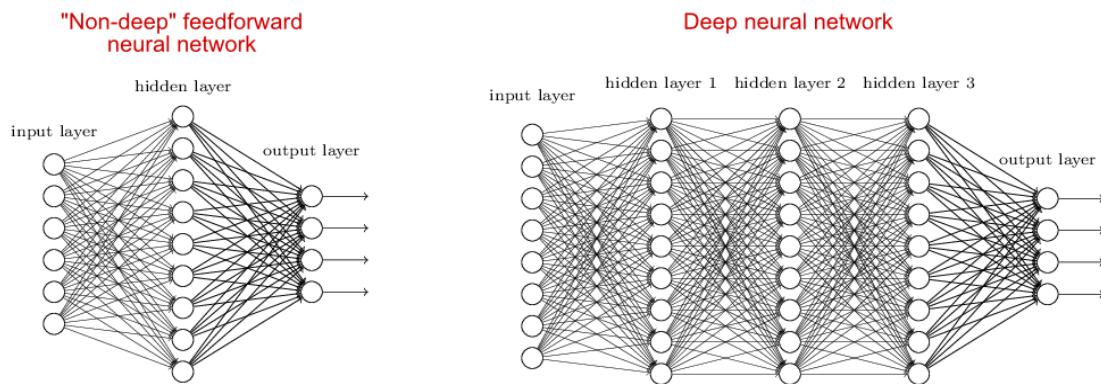
Παρακάτω φαίνεται ένα προσω τροφοδοτούμενο και ένα ανατροφοδοτούμενο νευρωνικό δίκτυο:



## Η δομή των Βαθέων Νευρωνικών Δικτύων (Deep Neural Networks):

Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ) που αποτελούνται από λίγα κρυφά επίπεδα παραμένουν στην κατηγορία των απλών τεχνητών νευρωνικών δικτύων. Τα δίκτυα που έχουν πολλά κρυφά επίπεδα εκτός από τεχνητά ονομάζονται και Βαθιά Νευρωνικά Δίκτυα (ΒΝΔ).

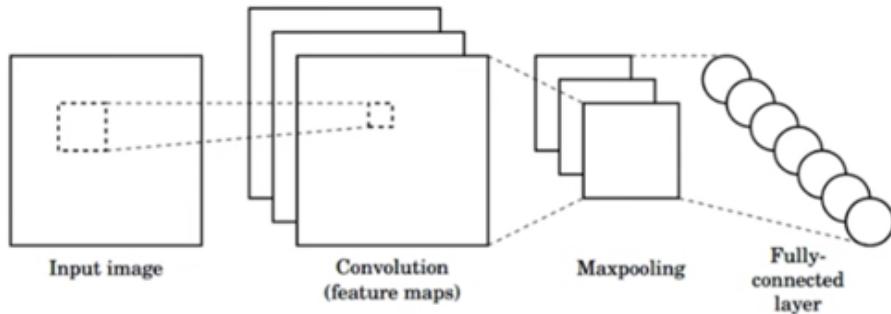
Παρακάτω φαίνονται ένα παράδειγμα ΤΝΔ και ένα παράδειγμα ΒΝΔ:



## Η δομή των Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks):

Τα συνελικτικά νευρωνικά δίκτυα (**CNN**) είναι μια κατηγορία βαθέων νευρωνικών δικτύων που χρησιμοποιούνται σε προβλήματα Επεξεργασίας Εικόνας και Όρασης Υπολογιστών όπως οπτική αναγνώριση (**visual recognition**), εντοπισμός αντικειμένου (**object detection**), κλπ. Τα νευρωνικά αυτά δίκτυα είναι βασισμένα στην μαθηματική πράξη της συνέλιξης.

Μια απλή αρχιτεκτονική ενός CNN φαίνεται παρακάτω:



Τα βασικά στοιχεία ενός CNN είναι τα παρακάτω:

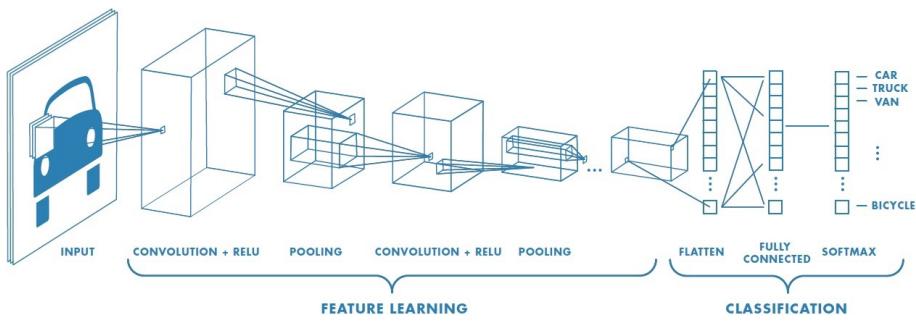
- Η πράξη της συνέλιξης (**convolution**), η οποία πραγματοποιείται με την βοήθεια φίλτρων για την παραγωγή χαρτών από χαρακτηριστικά (**feature maps**).
- Η μη γραμμικότητα (**non-linearity**) η οποία πραγματοποιείται με τη βοήθεια των συναρτήσεων ενεργοποίησης (πχ ReLU) με σκοπό την αντιμετώπιση μη γραμμικών δεδομένων.
- Η μέθοδος **Pooling**, η οποία είναι ένας τρόπος υποδειγματοληψίας στους χάρτες χαρακτηριστικών.

Στα συνελικτικά νευρωνικά δίκτυα, η εύρεση χαρτών χαρακτηριστικών είναι πολύ σημαντική. Με τους χάρτες χαρακτηριστικών, μπορεί το CNN να μάθει κάποια χαρακτηριστικά (**features**) της κάθε εικόνας και τελικά να κάνει την πρόβλεψη πάνω στο συγκεκριμένο πρόβλημα πχ αυτό της ταξινόμησης (**classification**).

Για την επίλυση του προβλήματος της ταξινόμησης πολλαπλών τάξεων (**multi-class classification**), το CNN αποτελείται από τα παρακάτω επιμέρους δίκτυα:

- Ένα δίκτυο που είναι υπεύθυνο για την Εκμάθηση χαρακτηριστικών (**feature learning**).
- Ένα δίκτυο που είναι υπεύθυνο για την επίλυση του συγκεκριμένου προβλήματος της ταξινόμησης πολλαπλών κλάσεων.

Παρακάτω φαίνονται τα δύο αυτά επιμέρους δίκτυα που ορίζουν ένα CNN σε απλή μορφή:

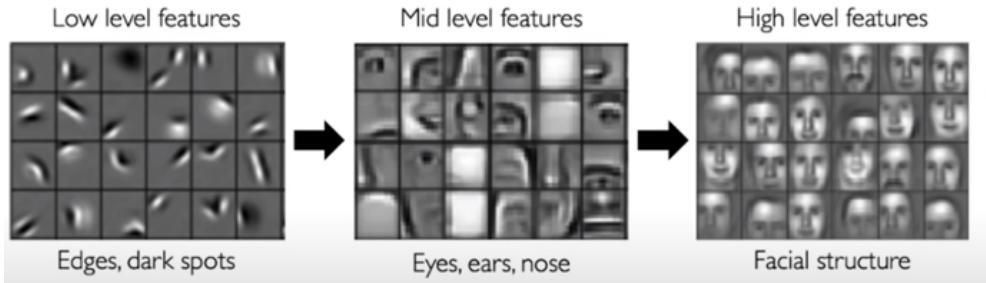


### Feature Learning και Convolutions:

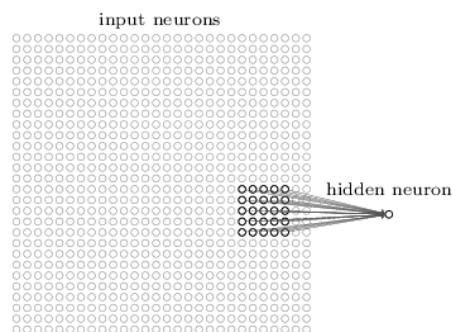
Το νευρωνικό δίκτυο αποτελείται από επίπεδα συνέλιξης (**convolutional layers**) και από επίπεδα pooling (**pooling layers**).

Τα επίπεδα συνέλιξης είναι το βασικότερο συστατικό των CNN. Κάθε επίπεδο συνέλιξης, αποτελείται από έναν αριθμό από φίλτρα (**filters**), στο καθένα από τα οποία εφαρμόζεται η πράξη της συνέλιξης μεταξύ αυτού και της εικόνας εισόδου και παράγεται ένα feature map. Στο πρώτο συνελικτικό επίπεδο, για την περίπτωση ενός φίλτρου το σήμα εισόδου είναι η εικόνα δεδομένων και η έξοδος είναι ο χάρτης χαρακτηριστικών του πρώτου επιπέδου. Για όλα τα επόμενα συνελικτικά επίπεδα (δεδομένου πάλι ότι όλα τα επίπεδα αυτά έχουν ένα φίλτρο), το σήμα εισόδου είναι ο χάρτης χαρακτηριστικών του προηγούμενου επιπέδου και η έξοδος είναι ένας νέος χάρτης χαρακτηριστικών, ο οποίος θα εισαχθεί στο επόμενο συνελικτικό επίπεδο. Στην πραγματικότητα τα συνελικτικά επίπεδα δεν αποτελούνται μόνο από ένα φίλτρο αλλά από πολύ περισσότερα (πχ 16, 32, 64, 128 κλπ). Όσα περισσότερα φίλτρα διαθέτει ένα τέτοιο επίπεδο, πάνω σε τόσα περισσότερα χαρακτηριστικά θα μπορεί να εκπαιδευτεί το συγκεκριμένο επίπεδο. Ο αριθμός των φίλτρων καθορίζει τον αριθμό των χαρακτηριστικών που μπορεί να μάθει το δίκτυο σε εκείνο το επίπεδο. Στα πρώτα επίπεδα τα χαρακτηριστικά που μπορεί να μάθει το δίκτυο είναι απλά, για παράδειγμα μπορεί να μάθει να αναγνωρίζει ακμές. Τα χαρακτηριστικά που μπορούν να μάθουν να αναγνωρίζουν τα επόμενα συνελικτικά επίπεδα κυρίως τα τελευταία είναι πιο σύνθετα, για παράδειγμα μπορεί τα δίκτυο να μάθει να αναγνωρίζει απλά σχήματα, στην συνέχεια πιο σύνθετα σχήματα και αργότερα πιο σύνθετες δομές (πχ ρόδα αυτοκινήτου, φανάρι αυτοκινήτου κλπ).

Αυτή η διαφορά στις δομές που μπορούν να μάθουν να αναγνωρίζουν τα διάφορα συνελικτικά επίπεδα με βάση την σειρά τους στο δίκτυο, φαίνεται παρακάτω:



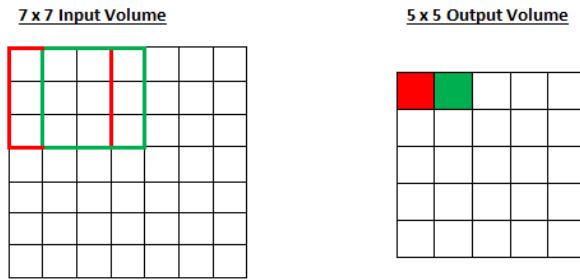
Μια διαφορά στα συνελικτικά νευρωνικά δίκτυα είναι ότι κάθε νευρώνας ενός επιπέδου δεν έχει ως είσοδο την έξοδο όλων των νευρώνων του προηγούμενου επιπέδου αλλά μόνο ένα μικρό κομμάτι αυτών, το οποίο εξαρτάται από το μέγεθος των φίλτρων που ορίζονται στο επίπεδο αυτό. Αυτό φαίνεται παρακάτω:



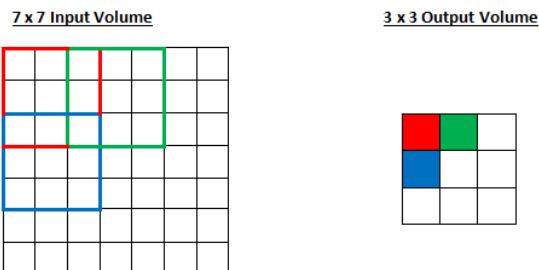
Οι διαστάσεις των feature maps εξόδου των συνελικτικών επιπέδων εξαρτώνται από:

- Το μέγεθος του feature map του προηγούμενου επιπέδου (ή το μέγεθος της εικόνας εισόδου για την περίπτωση του πρώτου συνελικτικού επιπέδου),(ας το ονομάσουμε **N**).
- Το μέγεθος του κάθε φίλτρου που επιλέγεται να συνελιχθεί με το feature map του προηγούμενου επιπέδου (ας το ονομάσουμε **F**).
- Κάποιους παραμέτρους της διαδικασίας της συνέλιξης (πχ **padding**, **stride**)
  - **Padding:** Αν θα υπάρξει επαύξηση των τιμών του feature map (με μηδενικά συνήθως) κατά δύο σειρές και στήλες στις δύο διαστάσεις του (ύψος, πλάτος).
  - **Stride:** Ο αριθμός της επικάλυψης διαδοχικών μετατοπίσεων του φίλτρου πάνω στο feature map στην διαδικασία της συνέλιξης.

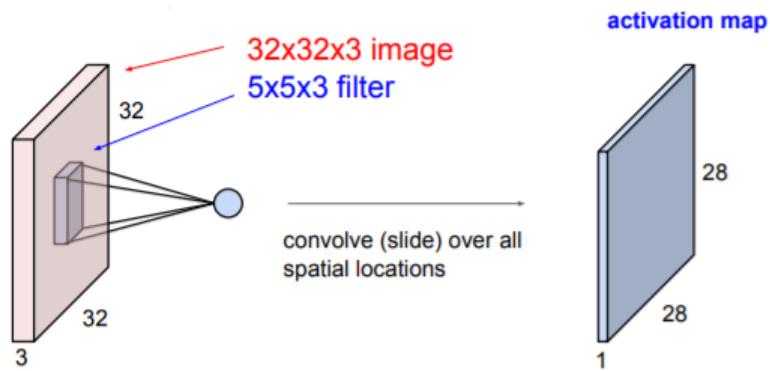
Για την περίπτωση του stride = 1:



Για την περίπτωση του stride = 2:



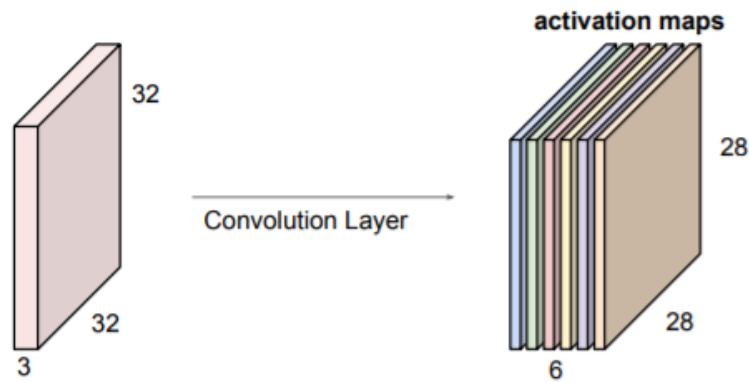
Παρακάτω φαίνεται ένα παράδειγμα για μια εικόνα εισόδου διαστάσεων (32,32,3) και για ένα μοναδικό φίλτρο διαστάσεων (5,5,3). Το παράδειγμα μελετά την περίπτωση εικόνας με 3 χρωματικά κανάλια. Το αποτέλεσμα της συνέλιξης μεταξύ της εικόνας εισόδου και του φίλτρου με τις συγκεκριμένες διαστάσεις και με την περίπτωση του stride=1, θα δώσει αποτέλεσμα ένα feature map διαστάσεων (28,28,1).



Η διάσταση του feature map εξόδου γενικά μπορεί να υπολογιστεί από τον παρακάτω τύπο:

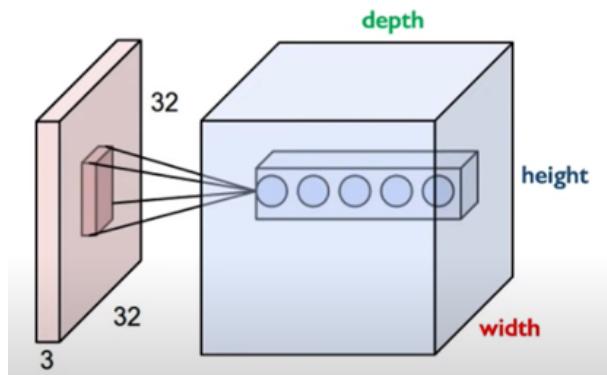
$$(N - F) / \text{stride} + 1$$

Για την περίπτωση παραπάνω από ένα φίλτρων, ισχύουν τα ίδια. Για την περίπτωση της εικόνα εισόδου διαστάσεων (32,32,3) και 6 φίλτρων διαστάσεων (5,5,3), το αποτέλεσμα θα είναι 6 feature maps διαστάσεων (28,28,1). Με άλλα λόγια θα έχουμε ένα feature map στην έξοδο του επιπέδου που θα έχει διαστάσεις (28,28,6).



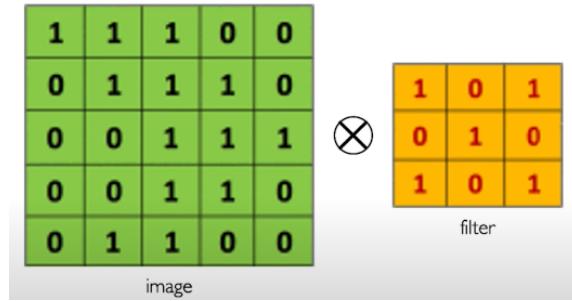
Από αυτό το παράδειγμα μπορεί να φανεί ότι τα feature maps που παράγουν τα συνελικτικά επίπεδα δεν είναι ένας δισδιάστατος χάρτης χαρακτηριστικών, αλλά ένας όγκος χαρακτηριστικών (**features volume**).

Παρακάτω φαίνεται η έξοδος ενός επιπέδου για την είσοδο μιας εικόνας (ή feature map) εισόδου:

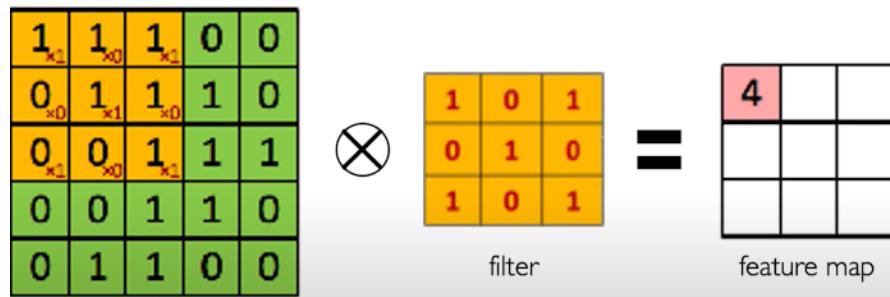


Στην παραπάνω εικόνα το width και το height είναι οι χωρικές διαστάσεις (**spatial dimensions**), οι οποίες εξαρτώνται όπως αναφέρθηκε προηγουμένως από παράγοντες όπως οι διαστάσεις του σήματος εισόδου ( $N$ ), οι διαστάσεις των φίλτρων του συγκεκριμένου επιπέδου ( $F$ ), και παραμέτρους όπως το stride και το padding. Το depth είναι ο αριθμός των φίλτρων με βάση τα οποία θα εκπαιδευτεί αυτό το συνελικτικό επίπεδο και ο αριθμός των χαρακτηριστικών που θα μάθει να μαθαίνει.

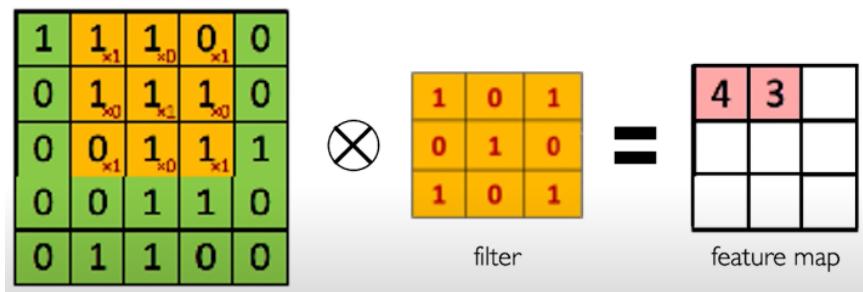
Όσον αφορά το κομμάτι των convolutions, παρακάτω φαίνεται ένα παράδειγμα συνελιξης σε δισδιάστατο επίπεδο για μια συγκεκριμένη εικόνα και ένα συγκεκριμένο φίλτρο:



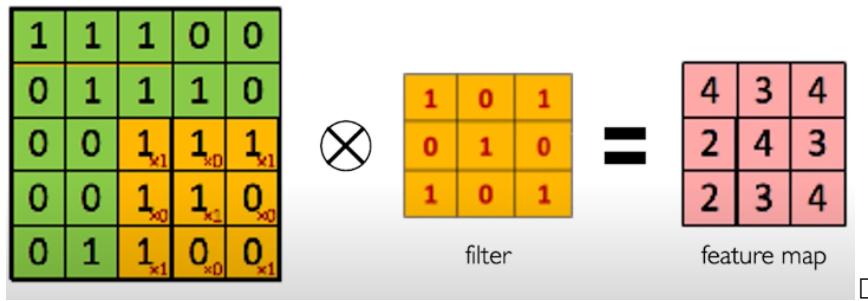
Αρχικά εφαρμόζεται συνέλιξη μεταξύ του φίλτρου και της περιοχής της εικόνας που φαίνεται παρακάτω. Το αποτέλεσμα της συνέλιξης αυτού του βήματος φαίνεται στην παρακάτω εικόνα και είναι το άθροισμα των πολλαπλασιασμών κατά στοιχείο των δύο πινάκων:



Στην συνέχεια αυτό συμβαίνει και για την επόμενη περιοχή της εικόνας, η οποία καθορίζεται από το stride, το οποίο επιλέγεται να είναι 1. Το αποτέλεσμα φαίνεται παρακάτω:



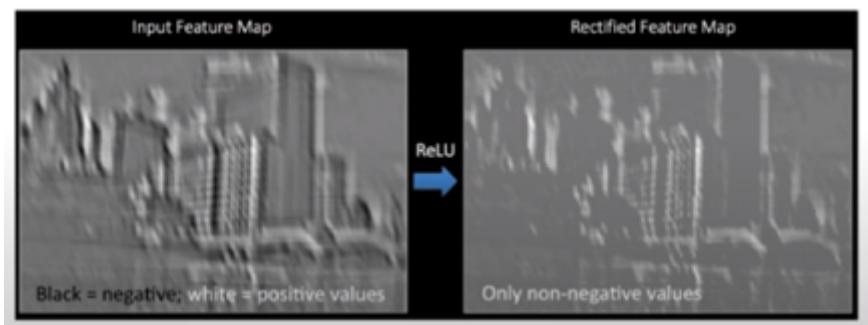
Αυτό συμβαίνει για όλη την εικόνα εισόδου με συνεχείς μετατοπίσεις (κατά σειρές και κατά στήλες) και τελικά το feature map που παράγεται στην έξοδο του επιπέδου με αυτό το φίλτρο είναι:



Το φίλτρο αυτό επειδή έχει 1 τόσο στις διαγώνιους του όσο και στο κεντρικό εικονοστοιχείο, θα μπορούσαμε να πούμε ότι εντοπίζει χαρακτηριστικά που είναι της μορφής “X”. Από το feature map μπορεί να εντοπίσει κανείς ότι στην εικόνα εισόδου υπάρχουν χαρακτηριστικά που μοιάζουν με το επιθυμητό χαρακτηριστικό “X”, στις περιοχές που “αποκαλύπτει” το feature map, δηλαδή στην πάνω και κάτω δεξιά περιοχή, στην πάνω αριστερά περιοχή, τόσο και στην κεντρική περιοχή της εικόνας εισόδου.

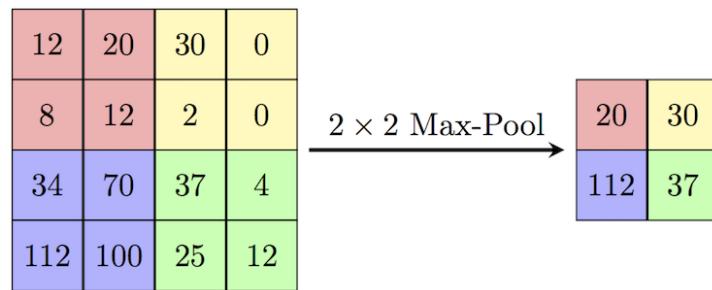
### Non - Linearity:

Στο τέλος της συνέλιξης στο κάθε επίπεδο, εφαρμόζεται ένα ακόμη στάδιο πριν περάσει το feature map στο επόμενο επίπεδο. Στο στάδιο αυτό το feature map περνάει από μια συνάρτηση ενεργοποίησης, όπως αναφέρθηκε προηγουμένως. Συγκεκριμένα κάθε τιμή του όγκου χαρακτηριστικών περνάει από την συνάρτηση αυτή και ένας νέος όγκος χαρακτηριστικών με ανανεωμένες τιμές είναι πλέον έτοιμος να περάσει ώς είσοδο στο επόμενο επίπεδο. Παρακάτω φαίνεται αλλαγή που προκαλεί η συνάρτηση ενεργοποίησης (συγκεκριμένα η ReLU) σε έναν χάρτη χαρακτηριστικών.



### Τεχνική Pooling:

Η τεχνική του pooling, εφαρμόζει μια υποδειγματοληψία στο feature map. Υπάρχουν διάφοροι τύποι pooling, όπως Average Pooling ή Max Pooling ή Min Pooling, συνήθως όμως εφαρμόζεται η τεχνική του Max Pooling. Παρακάτω φαίνεται ένα παράδειγμα, με μέγεθος pooling 2:



## Μη προ-εκπαιδευμένο δίκτυο:

Το πρώτο ζητούμενο αυτής της εργασίας ήταν η εκπαίδευση και η δοκιμή ενός δικτύου πάνω στο σετ δεδομένων που δόθηκε **BelgiumTS Dataset**. Το δίκτυο αυτό θα έπρεπε να μην ήταν προεκπαίδευμένο, αλλά να εκπαιδευτεί από την αρχή πάνω στο συγκεκριμένο σετ δεδομένων.

Για να ήταν επιτυχής η εκπαίδευση και η δοκιμή αυτού του δικτύου στο **BelgiumTS Dataset**, θα έπρεπε το δίκτυο να επιστρέψει υψηλά ποσοστά ακρίβειας (**accuracy**) στις δύο αυτές περιπτώσεις, δηλαδή υψηλό accuracy στο training και στο testing αντίστοιχα.

### Εκπαίδευση Δικτύου:

Προκειμένου το δίκτυο να πετύχει υψηλή ακρίβεια στο training set, θα έπρεπε να ληφθούν υπόψη διάφοροι παράγοντες. Μερικοί από τους παράγοντες αυτούς παρουσιάζονται παρακάτω:

- Πλήθος και τύπος επιπέδων (**layers**) εκπαίδευσης
- Πλήθος εποχών (**epochs**) εκπαίδευσης
- Μέγεθος **batch**
- Επαύξηση δεδομένων (**data augmentation**)
- Ρυθμός μάθησης (**learning rate**)

### Τύπος επιπέδων εκπαίδευσης:

- Επίπεδα Συνέλιξης (**Conv2D()**)

Τα συνελικτικά επίπεδα είναι ο βασικότερος τύπων επιπέδων για τα CNN. Είναι τα δίκτυα στα οποία εφαρμόζεται η πράξη της συνέλιξης. Στα επίπεδα αυτά ορίζονται κάποια στοιχεία. Αρχικά ορίζεται το πλήθος των φίλτρων που θα χρησιμοποιηθούν στην συνέλιξη αυτού του επιπέδου. Μαζί επιλέγονται και οι διαστάσεις των φίλτρων αυτών. Έπειτα ορίζεται ο τύπος της συνάρτησης ενεργοποίησης η οποία θα εφαρμοστεί στο feature map που προέκυψε από την συνέλιξη πριν περάσει στο επόμενο επίπεδο. Στην συνέχεια ορίζεται εαν

Θα εφαρμοστεί η μέθοδος padding ή όχι στο feature map εισόδου πριν αυτό περάσει στο στάδιο της συνέλιξης.

- Επίπεδα **MaxPooling2D()**

Τα επίπεδα αυτά είναι εξίσου σημαντικά στα CNN. Ο ρόλος τους είναι η μείωση των χωρικών διαστάσεων των feature maps που υπολογίζουν τα επίπεδα στο προηγούμενο στάδιο, τα οποία είναι συνήθως τα συνελικτικά. Και σε αυτά τα επίπεδα επιλέγονται κάποιες παράμετροι. Αρχικά ορίζεται το μέγεθος του παραθύρου, με το οποίο θα υποδειγματοληφθεί η πληροφορία από το feature map. Στην συνέχεια επιλέγεται εάν θα υπάρχει επικάλυψη (**strides**) μεταξύ διαδοχικών μετατοπίσεων του παραθύρου κατά γραμμές και κατα στηλες και πόσο μεγάλη θα είναι αυτή η μετατόπιση. Επίσης και εδώ ορίζεται αν θα εφαρμοστεί η μέθοδος του padding ή όχι.

- Επίπεδα **Flatten()**

Τα επίπεδα αυτά χρησιμοποιούνται με σκοπό την μετατροπή ενός τρισδιάστατου feature map σε ένα διάνυσμα χαρακτηριστικών, δηλαδή σε έναν μονοδιάστατο πίνακα. Είναι απαραίτητο αυτό το βήμα, έτσι ώστε να περάσει η πληροφορία από το κομμάτι του feature learning του CNN στο κομμάτι του Classification, όπου υπάρχουν **Fully Connected** επίπεδα.

- Επίπεδα **Dense()**

Τα επίπεδα **Dense** είναι τα **Fully Connected** επίπεδα, στα οποία ο κάθε νευρώνας του επιπέδου συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Τα επίπεδα αυτού του τύπου λειτουργούν όπως ακριβώς τα επίπεδα των απλών ANN, τα οποία μπορούν να λύσουν προβλήματα ταξινόμησης παλινδρόμησης κλπ. Το τελικό επίπεδο του δικτύου είναι ένα **Dense** επίπεδο που περιέχει μια συγκεκριμένη συνάρτηση ενεργοποίησης, την **Softmax()**. Η Softmax μετατρέπει τις τιμές του τελευταίου Dense επιπέδου, που βρίσκονται σε ένα διάνυσμα τέτοιων διαστάσεων όσο και το πλήθος των διαχωρίσιμων κλάσεων που υπάρχουν στο κάθε πρόβλημα, σε μια κατανομή πιθανοτήτων. Έτσι μια εικόνα-δείγμα έχει στο τέλος του δικτύου μια πιθανότητα να ανήκει στην κάθε κλάση. Η κλάση με την

μεγαλύτερη πιθανότητα είναι και η προβλεπόμενη κλάση για την συγκεκριμένη εικόνα-δείγμα.

Το πλήθος των συνελικτικών επιπέδων και των επιπέδων γενικότερα ενός CNN, θα πρέπει να είναι συγκεκριμένο. Θα πρέπει να είναι ικανοποιητικά μεγάλο έτσι ώστε το δίκτυο να μπορεί να μάθει τα χαρακτηριστικά του σετ δεδομένων εκπαίδευσης, αλλά όχι πολύ μεγάλο για να μην υπάρχει μεγάλη καθυστέρηση και μεγάλες απαιτήσεις μνήμης στο δίκτυο.

#### Πλήθος εποχών εκπαίδευσης:

Μια εποχή ορίζεται μία επανάληψη εκπαίδευσης end-to-end του δικτύου, δηλαδή ένα εμπρός πέρασμα του δικτύου για μια εικόνα-δείγμα, υπολογισμός του ποσοστού επιτυχίας (loss) και τέλος ένα πίσω πέρασμα του δικτύου και επαναπροσδιορισμός των παραμέτρων (weights και biases) με σκοπό την ελαχιστοποίηση του loss. Γενικά τα CNN χρειάζονται έναν αριθμό από εποχές για να εκπαιδευτούν, ο οποίος ποικίλει ανάλογα με το πρόβλημα και το σετ δεδομένων για την συγκεκριμένη εφαρμογή. Ωστόσο αυτός ο αριθμός από εποχές δεν θα πρέπει να είναι μεγαλύτερος από όσο χρειάζεται διότι το μοντέλο του CNN θα οδηγηθεί σε υπερπροσαρμογή (**overfitting**), κατάσταση στην οποία το μοντέλο μαθαίνει την πληροφορία απλά απομνημονεύοντας το training set χωρίς να μπορεί να γενικευει και έτσι αποτυγχάνει στο testing set.

#### Μέγεθος batch:

Υπάρχει δυνατότητα πολλές εικόνες- δείγματα να περαστούν μαζί στο δίκτυο κατα την διαδικασία της εκπαίδευσης. Αυτή η τεχνική βοηθάει στην ακρίβεια του δικτύου. Το μέγεθος batch ορίζει πόσες εικόνας-δείγματα θα περαστούν μαζί στο δίκτυο. Όσο μεγαλύτερο είναι το batch size, τόσο καλύτερα είναι τα αποτελέσματα εκπαίδευσης. Ωστόσο υπάρχει ένα όριο καθώς όσο αυξάνεται το batch size, η κάρτα γραφικών δεν επαρκεί, άρα μπορεί να επιλέγεται το μέγιστο δυνατό.

#### Επαύξηση δεδομένων (data augmentation):

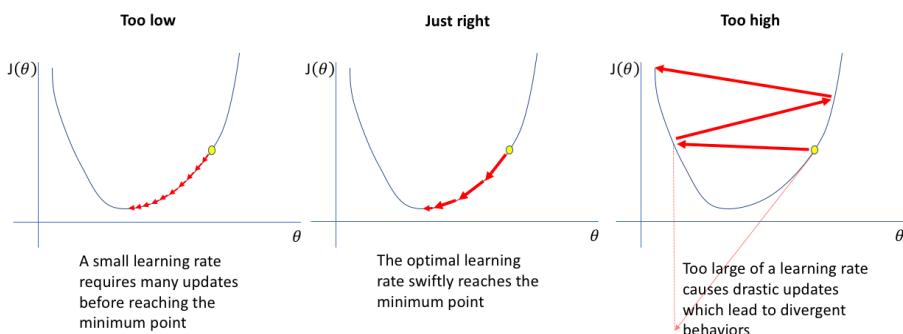
Τα CNN και γενικότερα τα ANN λειτουργούν καλύτερα όταν υπάρχουν πάρα πολλά δεδομένα. Η μέθοδος data augmentation παίρνει τις υπάρχουσες εικόνες-δείγματα και και

εφαρμόζει σε αυτές διάφορους μετασχηματισμούς εικόνων. Κάποια παραδείγματα αναφέρονται παρακάτω:

- Περιστροφή (**rotation\_range**)
- Οριζόντιος αντικατοπτρισμός (**horizontal\_flip**)
- Κάθετος αντικατοπτρισμός (**vertical\_flip**)
- Αλλαγή λαμπρότητας (**brightness\_range**)
- Μεγέθυνση (**zoom\_range**)

### Ρυθμός μάθησης (learning rate):

Η εκπαίδευση ενός νευρωνικού δικτύου, είναι ένα πρόβλημα εύρεσης των κατάλληλων παραμέτρων(weights και biases) με σκοπό την επίλυση ενός προβλήματος (πχ ταξινόμηση). Το πρόβλημα αυτό αποσκοπεί στην ελαχιστοποίηση του loss. Η παράμετρος learning\_rate καθορίζει πόσο γρήγορα το δίκτυο θα λύσει αυτό το πρόβλημα της ελαχιστοποίησης. Η τιμή του learning rate θα πρέπει να είναι τόσο μεγάλη έτσι ώστε να παρουσιάζεται βελτίωση της ακρίβειας του δικτύου από επανάληψη σε επανάληψη. Παρακάτω φαίνεται μια αναπαρασταση αυτού του προβλήματος:



### **Χρήση των Callbacks:**

Τα Callbacks είναι ένας τρόπος αυτοματοποιημένης παρέμβασης στην διαδικασία της εκπαίδευσης.

Κάποια παραδείγματα αναφέρονται παρακάτω:

- EarlyStopping
- ModelCheckpoint
- TensorBoard

#### EarlyStopping:

Πολλές φορές τα CNN μετά από ένα αριθμό από epochs έχουν φτάσει σε μια λύση και στις επόμενες epochs δεν βελτιώνουν την ακρίβεια του συστήματος. Σε εκείνο το σημείο μπορεί το EarlyStopping να σταματήσει εγκαίρως την εκπαίδευση με σκοπό την εξοικονόμηση χρόνου. Συγκεκριμένα ορίζεται μια παράμετρος που ονομάζεται patience και είναι το πλήθος των εποχών στις οποίες δεν θα παρατηρηθεί καμία βελτίωση στις μετρικές του συστήματος (accuracy ή loss) του επιθυμητού σετ δεδομένων (training ή validation)

#### ModelCheckpoint:

Σκοπός της δημιουργίας ενός CNN είναι η εξασφάλιση αυτών των παραμέτρων (weights και biases) που δίνουν την μέγιστη ακρίβεια. Κατα την διάρκει της εκπαίδευσης μπορεί το αποτέλεσμα να είναι το βέλτιστο αλλά μετά από κάποια εποχή να μειωθεί. Σε αυτήν την περίπτωση η αποθήκευση των παραμέτρων, κάθε φορά που η επιθυμητή μετρική είναι βέλτιστη, είναι απαραίτητη. Για την αποθήκευση αυτή φροντίζει το ModelCheckpoint.

#### TensorBoard:

Με την βοήθεια του TensorBoard, είναι εύκολη η οπτικοποίηση διαφόρων στοιχείων κατα την εκπαίδευση. Παρακάτω αναφέρονται κάποιες από τις δυνατότητες που παρέχει:

- Παρουσίαση μετρικών συστήματος με διαγράμματα
- Οπτικοποίηση γράφου εκπαίδευσης
- Ιστογράμματα ενεργοποίησης

#### **Προτεινόμενη Αρχιτεκτονική Δικτύου:**

Μετά από αρκετές δοκιμές, η αρχιτεκτονική του δικτύου η οποία έδωσε τα καλύτερα αποτελέσματα, δηλαδή υψηλή ακρίβεια στο σετ δεδομένων εκπαίδευσης (**training accuracy**) και δοκιμής (**testing accuracy**), φαίνεται παρακάτω :

```

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), padding="valid", activation='relu', input_shape=(256, 256, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), padding="valid", activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), padding="valid", activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), padding="valid", activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(34, activation='softmax'))

```

Όπως φαίνεται, επιλέχθηκαν τέσσερα συνελικτικά επίπεδα (**Conv2D()**), κάθε ένα από τα οποία ακολουθείται από ένα επίπεδο **MaxPooling2D()**. Στα τέσσερα αυτά συνελικτικά επίπεδα επιλέχθηκε ο αριθμός των φίλτρων να ισούται με τις τιμές: {32, 64, 64, 128}. Η αύξηση του πλήθους των φίλτρων είναι μια συνηθισμένη τεχνική στα CNN, έτσι ώστε το δίκτυο να μαθαίνει όλο και περισσότερα χαρακτηριστικά (features), όσο αυξάνεται το βάθος του. Το μέγεθος των φίλτρων παραμένει σταθερό. Η συνάρτηση ενεργοποίησης κάθε συνελικτικού επιπέδου επιλέχθηκε να είναι η ReLU. Επίσης δεν χρησιμοποιείται padding σε αυτά τα επίπεδα. Επιλέγεται οι εικόνες-δείγματα να προ επεξεργάζονται αλλάζοντας τις χωρικές διαστάσεις τους σε (256, 256).

Τα τέσσερα αυτά ζευγάρια συνελικτικών και pooling επιπέδων, ακολουθεί το Fully Connected κομμάτι του δικτύου. Αρχικά μετατρέπεται το τελικό feature map του προηγούμενου επιπέδου σε ένα διάνυσμα και στην συνέχεια η πληροφορία περνάει στο επίπεδο εξόδου περνώντας από πλήρως συνδεδεμένα επίπεδα. Επίσης υπάρχει ένα επίπεδο **Dropout()**, το οποίο μηδενίζει τυχαία την κάθε είσοδο του με μια πιθανότητα κάθε φορά που ορίζεται (εδώ 50%). Το τελικό επίπεδο διαθέτει 34 κόμβους, αριθμός ο οποίος είναι και το πλήθος των κλάσεων του συγκεκριμένου προβλήματος της ταξινόμησης. Το επίπεδο αυτό διαθέτει για συνάρτηση ενεργοποίησης την συνάρτηση **Softmax**, η οποία αντιστοιχεί μια πιθανότητα που ανήκει στο (0,1) για την κάθε κλάση. Η κλάση με την μεγαλύτερη πιθανότητα είναι και η πρόβλεψη του δικτύου για το πρόβλημα της ταξινόμησης πολλαπλών κλάσεων (multi-class classification).

Παρακάτω φαίνεται οι παράμετροι του δικτύου τόσο στο κάθε επίπεδο ξεχωριστά όσο και συνολικά :

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
mobilenet_0.50_224 (Functional)	(None, 512)	829536
dense (Dense)	(None, 1024)	525312
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 320)	328000
dropout_1 (Dropout)	(None, 320)	0
dense_2 (Dense)	(None, 34)	10914

Total params:	1,693,762
Trainable params:	1,334,498
Non-trainable params:	359,264

Επίσης χρησιμοποιήθηκε ο optimizer Adam με learning rate = 0.0001 καθώς και για την ελαχιστοποίηση του σφάλματος (**loss**), χρησιμοποιήθηκε η μέθοδος **Categorical Crossentropy**, η οποία είναι μια συνάρτηση σφάλματος που χρησιμοποιείται συχνά σε προβλήματα ταξινόμησης πολλαπλων κλασεων. Τα στοιχεία αυτά φαίνονται παρακάτω:

```
model.compile(optimizer=optimizers.adam_v2.Adam(learning_rate=1e-4),
              loss = 'categorical_crossentropy',
              metrics=['accuracy'])
```

Για την τροφοδοσία του δικτύου με τα δεδομένα χρησιμοποιήθηκε ο **ImageDataGenerator**, στον οποίο ορίστηκαν οι μετασχηματισμοί που εφαρμόστηκαν στα δεδομένα εκπαίδευσης με σκοπό την επαύξηση τους. Οι μετασχηματισμοί αυτοί φαίνονται παρακάτω:

```
train_datagen = ImageDataGenerator(rotation_range = 90,
                                    horizontal_flip=True,
                                    vertical_flip = True,
                                    brightness_range=[0.2,1.0],
                                    zoom_range=[0.5,1.0],
                                    validation_split=0.2)
```

Το **validation\_split** ορίστηκε στο 0.2 και αυτό σημαίνει ότι το σετ δεδομένων εκπαίδευσης χωρίστηκε σε ένα κομμάτι που προοριζόταν για την εκπαίδευση του δικτύου και ήταν το 80% του συνολικού και σε ένα 20% που προοριζόταν για την αξιολόγηση του δικτύου κατα την διάρκεια της εκπαίδευσης. Επίσης το **batch\_size** ορίστηκε στο 40, δηλαδή 40 εικόνες-δείγματα περνούσαν το δίκτυο κάθε φορά. Τα callbacks που χρησιμοποιήθηκαν ήταν το EarlyStopping και το ModelCheckpoint.

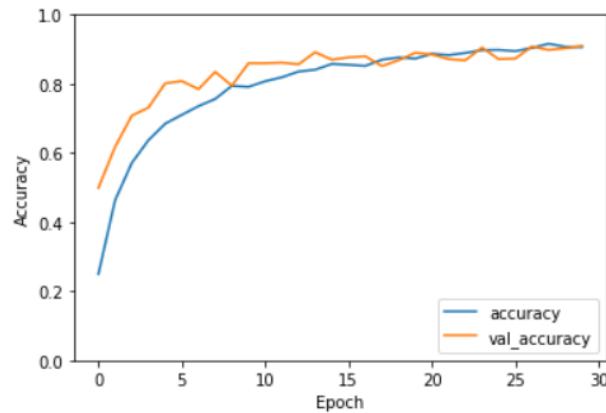
Το δίκτυο έτρεξε για 30 epochs και τα αποτελέσματα του στην εκπαίδευση ήταν:

```
Epoch 30/30
62/61 [=====] - ETA: 0s - loss: 0.3277 - accuracy: 0.9064
Epoch 00030: val_loss improved from 0.35954 to 0.35359, saving model to model_from_scratch_best.hdf5
61/61 [=====] - 46s 756ms/step - loss: 0.3277 - accuracy: 0.9064 - val_loss: 0.3536 - val_accuracy: 0.9098
```

και για την δοκιμή:

```
215/215 [=====] - 5s 20ms/step - loss: 1.5507 - accuracy: 0.7073
```

Παρακάτω φαίνεται και ένα γράφημα της ακρίβειας (**training** και **validation**) συναρτήσει των εποχών εκπαίδευσης:



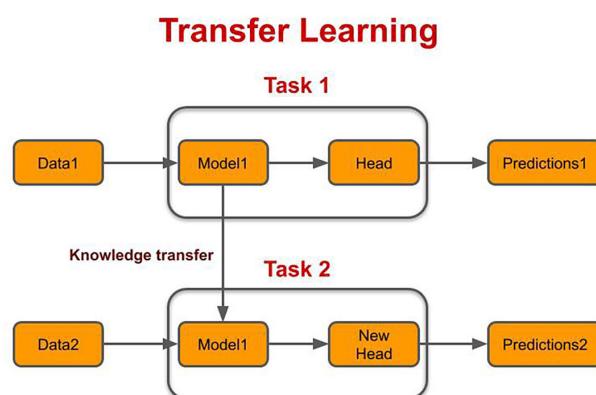
Όπως φαίνεται παραπάνω το δίκτυο πέτυχε ακρίβεια στο σετ δεδομένων εκπαίδευσης **0.9064**, δηλαδή **90.64%** ενώ στο σετ δεδομένων δοκιμής **0.7073** δηλαδή **70,73%**. Συνήθως ένα τέτοιο ζευγάρι για τις ακρίβειες ενός δικτύου παραπέμπει σε Overfitting, δηλαδή σε απομνημόνευση των ήδη γνωστών εικόνων από το σετ εκπαίδευσης και όχι γενίκευση τους. Ωστόσο η διαφορά αυτή των ακριβειών δεν είναι τόσο μεγάλη, οπότε μπορούμε να δεχτούμε ότι το δίκτυο έχει εκπαιδευτεί ικανοποιητικά καλά, με μια εγγύηση όμως ότι η ακρίβεια του σε άγνωστα δεδομένα θα φτάνει μέχρι το 70%.

## Προ-εκπαιδευμένο δίκτυο:

### Τεχνική Transfer Learning:

Στην εργασία αυτή, ζητούμενη ήταν η εφαρμογή ενός προεκπαιδευμένου δίκτυου στο σύνολο δεδομένων. Γενικά το προεκπαιδευμένα δίκτυα έχουν εκπαιδευτεί (διαδικασία training και validation) και δοκιμαστεί (διαδικασία testing) σε ένα πρόβλημα με ένα συγκεκριμένο σύνολο δεδομένων και φτάνοντας σε ένα σημείο υψηλών επιδόσεων έχουν αποθηκευτεί με σκοπό την επαναχρησιμοποίηση τους σε παρόμοια προβλήματα. Με άλλα λόγια, ζητούμενη ήταν η εφαρμογή της τεχνικής **Transfer Learning**. Κατά την επαναχρησιμοποίησή τους στο νέο πρόβλημα και συνεπώς στο νέο σύνολο δεδομένων, για τα προεκπαιδευμένα δίκτυα δεν είναι αναγκαίο να εκπαιδευτούν και πάλι από την αρχή. Αντιθέτως το μεγάλο πλεονέκτημα της τεχνικής Transfer Learning είναι η εφαρμογή του προεκπαιδευμένου δίκτυου στα επιθυμητά δεδομένα, ξεπερνώντας το στάδιο της εκπαίδευσης, το οποίο είναι πάντα το πιο χρονοβόρο.

Παρακάτω φαίνεται ένα παράδειγμα της τεχνικής Transfer Learning:



Το προεκπαιδευμένο δίκτυο (*Task1*) έχει εκπαιδευτεί πάνω σε ένα συγκεκριμένο σύνολο δεδομένων (*Data1*) με σκοπό την επίλυση ενός συγκεκριμένου προβλήματος πρόβλεψης (*Predictions1*). Το δίκτυο αυτό, όπως τα περισσότερα συνελικτικά νευρωνικά δίκτυα, αποτελείται από δύο μέρη. Το πρώτο μέρος της διαδικασίας εκμάθησης χαρακτηριστικών-feature learning (*Model1*) και το δεύτερο μέρος της διαδικασίας της ταξινόμησης-classification (*Head*). Το προεκπαιδευένο δίκτυο μπορεί να “μεταφέρει” (*transfer*) την “γνώση αυτή που έχει μάθει” (*learning*), σε ένα νέο σύστημα δεδομένων

(*Data2*) χρησιμοποιώντας μόνο το πρώτο μέρος (*Model1*) του δικτύου. Στην συνέχεια προστίθενται κάποια επιπλέον layers στο τέλος του προεκπαιδευμένου δικτύου (*New Head*), και έτσι μπορεί να γίνουν οι νέες προβλέψεις (*Predictions2*).

### Προ-εκπαιδευμένο δίκτυο MobileNet:

Στο πλαίσιο αυτής της εργασίας επιλέχθηκε να μελετηθεί το προ-εκπαιδευμένο δίκτυο **MobileNet**. Το MobileNet έχει προεκπαιδευτεί στο διάσημο σετ δεδομένων **IMAGENET**, οπότε μπορεί να χρησιμοποιηθεί και στο **BelgiumTS Dataset**. Το δίκτυο αυτό χρησιμοποιείται σε εφαρμογές κινητών συσκευών (**mobile devices**) και ενσωματωμένων συστημάτων (**embedded systems**) που σχετίζονται με το πεδίο της Όρασης Υπολογιστών (**computer vision**).

Παρακάτω φαίνονται κάποια παραδείγματα αυτών των εφαρμογών:



Για τον λόγο ότι τα δίκτυα αυτά θα τρέχουν σε τέτοιες συσκευές άρα και σε πραγματικό χρόνο, θα πρέπει να χαρακτηρίζονται από μικρό μέγεθος στην μνήμη και από μεγάλη ταχύτητα ή ισοδύναμα χαμηλή καθυστέρηση (**latency**). Παρακάτω φαίνεται μια σύγκριση ανάμεσα στο MobileNet και στο VGG16, ένα άλλο γνωστό προεκπαιδευμένο δίκτυο:

model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (GPU)
<b>VGG16</b>	528	0.713	0.901	138,357,544	23	4.16

<b>MobileNet</b>	16	0.704	0.895	4,253,864	88	3.44
------------------	----	-------	-------	-----------	----	------

Τα δίκτυα MobileNet (MobileNet και MobileNetV2) προκειμένου να έχουν χαμηλή καθυστέρηση, είναι βασισμένα σε μια αρχιτεκτονική που χρησιμοποιεί έναν διαφορετικό τύπο συνελίξεων που ονομάζεται Συνελίξεις διαχωριζόμενες κατά βάθος (**Depthwise Separable Convolution**).

### Τεχνική Depthwise Separable Convolution:

Η τεχνική **Depthwise Separable Convolution** είναι μια τεχνική η οποία μειώνει την απαίτηση των συνελίξεων των συνελικτικών δικτύων σε πολλαπλασιασμούς. Οι πολλαπλασιασμοί είναι μια πράξη μεγάλης πολυπλοκότητας για τον υπολογιστή, η οποία χρειάζεται αρκετό χρόνο και πόρους συστήματος. Οπότε η μείωση των πολλαπλασιασμών είναι ένα μεγάλο προτέρημα για τα CNN, τα οποία στηρίζονται στις συνελίξεις και άρα στην πράξη του πολλαπλασιασμού.

Η τεχνική **Depthwise Separable Convolution** χωρίζεται σε δύο βήματα:

1. Συνέλιξη κατά βάθος (**Depthwise Convolution**)
2. Συνελίξεις κατά σημεία (**Pointwise Convolutions**)

Για την εξήγηση της διαδικασίας, θεωρούμε για ένα επίπεδο:

- feature map εισόδου ( $D_F \times D_F \times M$ )
- διάνυσμα φίλτρων για την συνέλιξη ( $D_K \times D_K \times N$ )

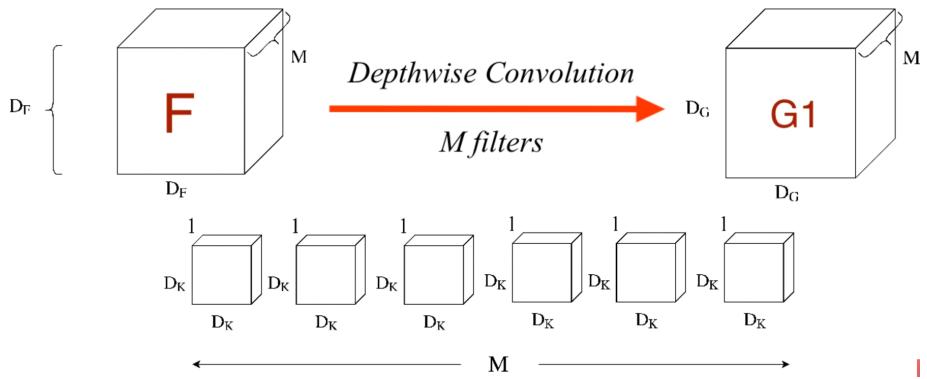
όπου  $D_F$  : οι χωρικές διαστάσεις του feature map εισόδου,  $M$  : το πλήθος των καναλιών (**channels**) του feature map εισόδου,  $D_K$  : οι χωρικές διαστάσεις του φίλτρου συνέλιξης,  $N$  : το πλήθος των φίλτρων ( $D_K \times D_K \times 1$ ).

#### Στο πρώτο βήμα (Depthwise Convolution):

Χρησιμοποιούνται  $M$  φίλτρα ( $D_K \times D_K \times 1$ ) και έχουμε  $M$  συνελίξεις. Μια συνέλιξη για κάθε ένα channel του feature map εισόδου ( $D_F \times D_F \times 1$ ). Το αποτέλεσμα αυτών των συνελίξεων

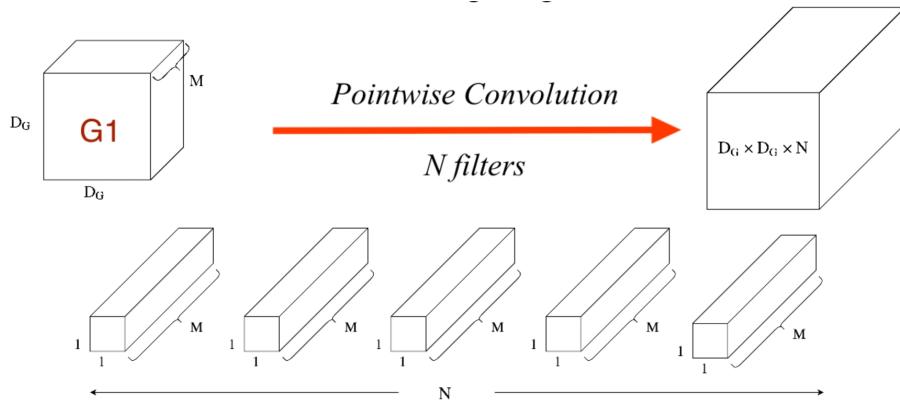
είναι  $M$  feature maps διαστάσεων  $(D_G \times D_G \times 1)$ , ή διαφορετικά ένας όγκος  $(D_G \times D_G \times M)$ .

Παρακάτω φαίνεται η διαδικασία **Depthwise Convolution**:



#### Στο δεύτερο βήμα (Pointwise Convolutions):

Χρησιμοποιούνται  $N$  φίλτρα  $(1 \times 1 \times M)$  και έχουμε  $N$  συνελίξεις. Μια συνέλιξη για κάθε ένα channel του feature map που παράχθηκε στο προηγούμενο βήμα  $(D_G \times D_G \times M)$ . Το αποτέλεσμα αυτών των συνελίξεων είναι  $N$  feature maps διαστάσεων  $(D_G \times D_G \times 1)$ , ή διαφορετικά ένας όγκος  $(D_G \times D_G \times N)$ . Παρακάτω φαίνεται η διαδικασία **pointwise convolutions**:



#### Σχόλια - Παρατηρήσεις:

- Όπως εξηγήθηκε από ένα feature map εισόδου διαστάσεων:  $(D_F \times D_F \times M)$ , μετά την **Depthwise Separable Convolution** με ένα διάνυσμα φίλτρων διαστάσεων:  $(D_K \times D_K \times N)$ , πρέκυψε ένα feature map διαστάσεων:  $(D_G \times D_G \times N)$ . Το αποτέλεσμα ήταν αναμενόμενο καθώς αυτή η τεχνική έχει το ίδιο αποτέλεσμα μια μια κανονική συνέλιξη με την μόνη διαφορά ότι γίνεται πιο βέλτιστα από την πλευρά των πράξεων.

- Η παραπάνω διαδικασία πραγματοποιείται για την περίπτωση των stride=1 και χωρίς padding. Για την περίπτωση με padding, το feature map εξόδου θα ήταν διαστάσεων:  $(D_F \times D_F \times N)$ .

### Συγκριση πράξεων Depthwise Separable Convolution και Standard Convolution:

Όπως αναφέρθηκε παραπάνω η τεχνική **Depthwise Separable Convolution** μειώνει τον αριθμό των πολλαπλασιασμών σε σχέση με μια κανονική συνέλιξη (**Standard Convolution**). Παρακάτω θα αναλυθεί αυτή η διαφορά.

#### Πλήθος πολλαπλασιασμών κατά Depthwise Separable Convolution :

- **Depthwise Convolution:**
  - για μια συνέλιξη με κάθε φίλτρο διαστάσεων  $(D_K \times D_K \times 1) : D_K^2$
  - για ένα κανάλι του feature map εισόδου:  $D_G^2 \times D_K^2$
  - για M κανάλια του feature map εισόδου:  $M \times D_G^2 \times D_K^2$
- **Pointwise Convolutions:**
  - για μια συνέλιξη με κάθε φίλτρο διαστάσεων  $(1 \times 1 \times M) : M$
  - για ένα κανάλι του feature map που παράχθηκε στο **Depthwise Convolution**:  $D_G \times D_G \times M$
  - για N φίλτρα διαστάσεων  $(1 \times 1 \times M) : N \times D_G \times D_G \times M$
- **Συνολικά Depthwise Convolution και Pointwise Convolutions:**

$$M \times D_G^2 \times D_K^2 + N \times D_G \times D_G \times M$$

Στην παρακάτω εικόνα φαίνονται τα συγκριτικά αποτελέσματα με μια standard convolution:

$$\frac{\text{No. Mults in Depthwise Separable Conv}}{\text{No. Mults in Standard Conv}} = \frac{M \times D_G^2 (D_K^2 + N)}{N \times D_G \times D_G \times D_K \times M}$$

$$\frac{\text{No. Mults in Depthwise Separable Conv}}{\text{No. Mults in Standard Conv}} = \frac{D_K^2 + N}{(D_K^2 \times N)} = \frac{1}{N} + \frac{1}{D_K^2}$$

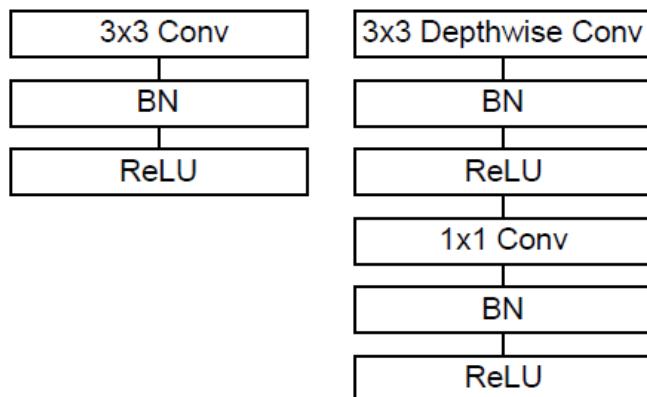
Για ένα παράδειγμα όπου  $N = 1024$  φίλτρα και  $D_K = 3$  :

$$\frac{\text{No. Mults in Depthwise Separable Conv}}{\text{No. Mults in Standard Conv}} = \frac{1}{1024} + \frac{1}{3^2} = 0.112$$

Δηλαδή το πλήθος των πολλαπλασιασμών της τεχνικής **Depthwise Separable Convolution** υπολογίζεται περίπου στο **11.2%** των πολλαπλασιασμών μιας **Standard Convolution**, διαφορά η οποία είναι μεγάλη και αρκετά σημαντική.

### Αρχιτεκτονική δικτύου MobileNet:

Το MobileNet έχει κατασκευαστεί με **Depthwise Separable Convolution** επίπεδα. Σημειώνεται ότι το πρώτο επίπεδο συνέλιξης ακολουθεί την κανονική συνέλιξη. Όλα τα επίπεδα συνέλιξης ακολουθούνται από ένα επίπεδο κανονικοποίησης **batch normalization** και ένα επίπεδο ενεργοποίησης με την συνάρτηση ενεργοποίησης ReLU. Παρακάτω φαίνεται η σύγκριση μεταξύ ενός επιπέδου κανονικής συνέλιξης ακολουθούμενο από τα παραπάνω επίπεδα σε σχέση με ένα επίπεδο **Depthwise Separable Convolution**:



Παρακάτω φαίνεται ολόκληρη η αρχιτεκτονική του MobileNet:

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool $7 \times 7$	$7 \times 7 \times 1024$
FC / s1	$1024 \times 1000$	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Αρχικά υπάρχει το επίπεδο κανονικής συνέλιξης με διάνυσμα φίλτρων ( $3 \times 3 \times 32$ ). Έπειτα βρίσκονται 13 επίπεδα **Depthwise Separable Convolution**, στα οποία αρχικά υπάρχει το επίπεδο της depthwise convolution και έπειτα το επίπεδο της pointwise convolution. Για αυτά τα 13 επίπεδα το διάνυσμα φίλτρων στο οποίο εφαρμοζόταν η depthwise convolution αποφασίστηκε να έχει διαστάσεις: ( $3 \times 3 \times N$ ), όπου το  $N$  παίρνει τις συγκεκριμένες τιμές:  $\{32, 64, 128, 128, 256, 256, 512, 512, 512, 512, 512, 1024\}$ . Στην συνέχεια ακολουθεί ένα επίπεδο υποδειγματοληψίας Average Pooling. Τέλος βρίσκεται ένα fully connected επίπεδο το οποίο “απλώνει” όλες τις τιμές του προηγουμένου feature map σε ένα μονοδιάστατο διάνυσμα, ακολουθούμενο από το τελικό επίπεδο ταξινόμησης, το οποίο διαθέτει την συνάρτηση ενεργοποίησης Softmax, η οποία αντιστοιχεί τις τιμές των κόμβων εξόδου σε μια κατανομή πιθανοτήτων η οποία προσδίδει την πρόβλεψη του δικτύου.

#### Η παράμετρος Width Multiplier του MobileNet:

Το μοντέλο που ορίζει η αρχιτεκτονική του MobileNet είναι ήδη βέλτιστο, δηλαδή υπάρχει μικρή καθυστέρηση και μικρό σε επίπεδα. Ωστόσο κάποιες φορές για συγκεκριμένες εφαρμογές υπάρχει ανάγκη να μειωθεί ακόμα πιο πολύ το μέγεθος του δικτύου, άρα και η

καθυστερηση του. Με άλλα λόγια υπάρχει η ανάγκη για την δημιουργία πιο λεπτών δίκτυων (**thinner networks**). Για την βελτίωση αυτή στα δίκτυα MobileNet, χρησιμοποιείται η υπερ-παράμετρος (**hyperparameter**) του πολλαπλασιαστή πλάτους (**Width Multiplier**). Ο ρόλος του **Width Multiplier** είναι να λεπταίνει το δίκτυο με ομοιόμορφο τρόπο σε όλα τα επίπεδα.

Με την χρήση του **Width Multiplier  $\alpha$** , αλλάζει το πλήθος των καναλιών του feature map εισόδου από  $M$  σε  $\alpha M$  στο στάδιο depthwise convolution και το πλήθος των φίλτρων ( $D_K \times D_K \times 1$ ) από  $N$  σε  $\alpha N$ . Παρακάτω φαίνεται το υπολογιστικό κόστος σε πολλαπλασιασμούς, μετά την εφαρμογή του **Width Multiplier  $\alpha < 1$** :

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

Ο πολλαπλασιαστής πλάτους  $\alpha$  παίρνει τιμές στο  $(0,1]$  με τυπικές τιμές τις  $\{1, 0.75, 0.5, 0.25\}$ . Για  $\alpha=1$  το δίκτυο είναι το MobileNet που περιγράφηκε παραπάνω χωρίς “thinning”. Για τις τυπικές τιμές  $\alpha < 1$ , το δίκτυο λεπταίνει κατά αυτό το ποσοστό. Η παράμετρος αυτή μπορεί να μειώσει το υπολογιστικό κόστος και τον αριθμό των παραμέτρων (weights και biases) τετραγωνικά δηλαδή κατά  $\alpha^2$ . Ο πολλαπλασιαστής πλάτους θα πρέπει να χρησιμοποιείται λαμβάνοντας υπόψη το trade off μεταξύ μείωσης καθυστέρησης και μεγέθους με μείωση ακρίβειας. Με άλλα λόγια το δίκτυο θα πρέπει να έχει χαμηλό μέγεθος και latency, έτσι ώστε να χρησιμοποιηθεί σε εφαρμογές πραγματικού χρόνου αλλά όχι τόσο χαμηλά, διότι τότε θα πεφτει και η ακρίβεια του, μέγεθος το οποίο είναι άκρως σημαντικό.

### Η παράμετρος **Resolution Multiplier** του MobileNet:

Η δεύτερη υπερ-παράμετρος που χρησιμοποιείται για να μειωθεί το υπολογιστικό κόστος του δίκτυου είναι ο πολλαπλασιαστής ανάλυσης (**Resolution Multiplier**)  $\rho$ . Η υπερ-παράμετρος αυτή εφαρμόζεται στην εικόνα εισόδου και σε όλα τα εσωτερικά επίπεδα, μειώνοντας την ανάλυση των feature maps κατά αυτό το ποσοστό  $\rho$ . Παρακάτω φαίνεται το υπολογιστικό κόστος σε πολλαπλασιασμούς, μετά την εφαρμογή του **Width Multiplier  $\alpha < 1$**  και την εφαρμογή του **Resolution Multiplier  $\rho < 1$** :

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$$

Ο πολλαπλασιαστης ανάλυσης ρ παίρνει τιμές στο  $(0,1]$ , έτσι ώστε η ανάλυση της εικόνας εισόδου να έχει μια από τις εξείς τιμές {224, 192, 160, 128}. Για  $\rho=1$  το δίκτυο είναι το MobileNet που περιγράφηκε παραπάνω χωρίς μείωση ανάλυσης. Για τιμές  $\rho < 1$ , στο δίκτυο εφαρμόζεται μείωση ανάλυσης κατά αυτό το ποσοστό, έτσι ώστε η τελική ανάλυση να έχει τις συγκεκριμένες παραπάνω τιμές. Η παράμετρος αυτή μπορεί να μειώσει το υπολογιστικό κόστος τετραγωνικά δηλαδή κατά  $\rho^2$ .

### Προτεινόμενη Αρχιτεκτονική Δικτύου MobileNet:

Μετά από αρκετές δοκιμές, η αρχιτεκτονική του δικτύου MobileNet η οποία έδωσε τα καλύτερα αποτελέσματα, δηλαδή υψηλή ακρίβεια στο σετ δεδομένων εκπαίδευσης (**training accuracy**) και δοκιμής (**testing accuracy**), φαίνεται παρακάτω :

```
model.add(mobilenet_conv)

model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(320, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(34, activation='softmax'))
```

Αρχικά τοποθετείται το βασικό μοντέλο MobileNet χωρίς το fully connected κομμάτι. Σην συνέχεια τοποθετούνται τρία dense επίπεδα τα οποία αποτελούνται από {1024, 320, 34} νευρώνες. Οι συναρτήσεις ενεργοποίησης τους είναι η ReLU εκτός από αυτή του τελευταίου επιπέδου που χρειάζεται να είναι η Softmax, έτσι ώστε να παράξει μια κατανομή πιθανότητας για τις 34 κλάσεις του προβλήματος. Επίσης υπάρχουν και τα επίπεδα **Dropout()**, τα οποία μηδενίζουν τυχαία την κάθε είσοδο τους με μια πιθανότητα κάθε φορά που ορίζεται στο 50%.

Παρακάτω φαίνεται οι παράμετροι του δικτύου τόσο στο κάθε επίπεδο ξεχωριστά όσο και συνολικά :

Model: "sequential"		
Layer (type)	Output Shape	Param #
mobilenet_0.50_224 (Functional)	(None, 512)	829536
dense (Dense)	(None, 1024)	525312
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 320)	328000
dropout_1 (Dropout)	(None, 320)	0
dense_2 (Dense)	(None, 34)	10914

Total params: 1,693,762  
Trainable params: 1,334,498  
Non-trainable params: 359,264

Επίσης χρησιμοποιήθηκε ο optimizer Adam με learning rate = 0.0001 καθώς και για την ελαχιστοποίηση του σφάλματος (**loss**), χρησιμοποιήθηκε η μέθοδος **Categorical Crossentropy**, η οποία είναι μια συνάρτηση σφάλματος που χρησιμοποιείται συχνά σε προβλήματα ταξινόμησης πολλαπλων κλασεων. Τα στοιχεία αυτά φαίνονται παρακάτω:

```
model.compile(optimizer=optimizers.adam_v2.Adam(learning_rate=1e-4),
              loss = 'categorical_crossentropy',
              metrics=['accuracy'])
```

Για την τροφοδοσία του δικτύου με τα δεδομένα χρησιμοποιήθηκε ο **ImageDataGenerator**, στον οποίο ορίστηκαν οι μετασχηματισμοί που εφαρμόστηκαν στα δεδομένα εκπαίδευσης με σκοπό την επαύξηση τους. Οι μετασχηματισμοί αυτοί φαίνονται παρακάτω:

```
train_datagen = ImageDataGenerator(
    rotation_range = 90,
    horizontal_flip=True,
    vertical_flip = True,
    brightness_range=[0.2,1.0],
    zoom_range=[0.5,1.0],
    preprocessing_function=mobilenet.preprocess_input,
    validation_split=0.2)
```

Το **validation\_split** ορίστηκε στο 0.2 και αυτό σημαίνει ότι το σετ δεδομένων εκπαίδευσης χωρίστηκε σε ένα κομμάτι που προοριζόταν για την εκπαίδευση του δικτύου και ήταν το 80% του συνολικού και σε ένα 20% που προοριζόταν για την αξιολόγηση του δικτύου κατα την διάρκεια της εκπαίδευσης. Επίσης το **batch\_size** ορίστηκε στο 40, δηλαδή 40

εικόνες-δείγματα περνούσαν το δίκτυο κάθε φορά. Τα callbacks που χρησιμοποιήθηκαν ήταν το EarlyStopping, το ModelCheckpoint και το TensorBoard.

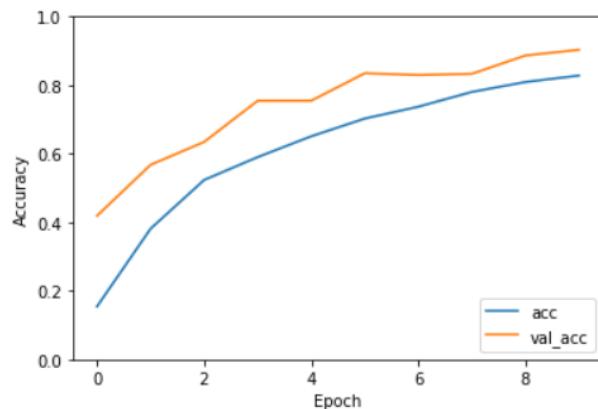
Το δίκτυο έτρεξε για 10 epochs και τα αποτελέσματα του στην εκπαίδευση ήταν:

```
Epoch 10/10
62/61 [=====] - ETA: 0s - loss: 0.6029 - acc: 0.8278
Epoch 00010: val_loss improved from 0.39931 to 0.35590, saving model to pretrained_MobileNet_batch_40_epochs_30.hdf5
61/61 [=====] - 42s 688ms/step - loss: 0.6029 - acc: 0.8278 - val_loss: 0.3559 - val_acc: 0.9032
```

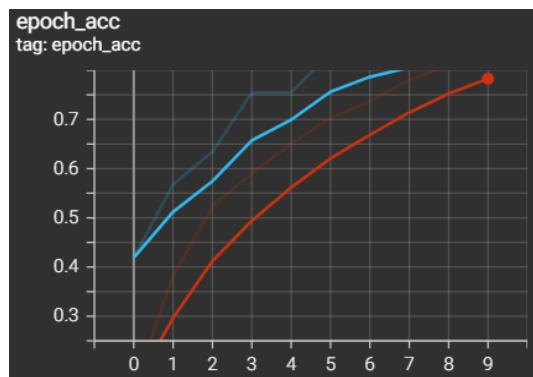
και για την δοκιμή:

```
215/215 [=====] - 5s 21ms/step - loss: 3.0853 - acc: 0.1936
```

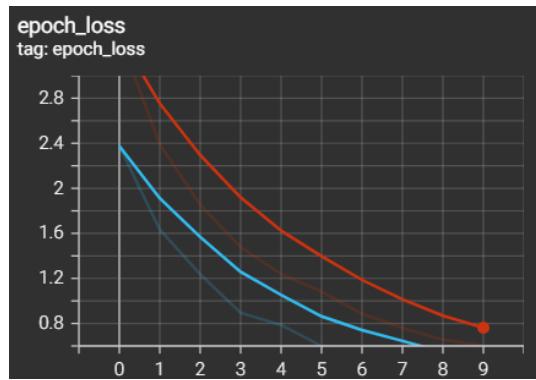
Παρακάτω φαίνεται και ένα γράφημα της ακρίβειας (**training** και **validation**) συναρτήσει των εποχών εκπαίδευσης:



Επίσης ένα γράφημα της ακρίβειας (**training** και **validation**) συναρτήσει των εποχών εκπαίδευσης, από το callback TensorBoard φαίνεται παρακάτω :



Επίσης ένα γράφημα του σφάλματος (**training** και **validation**) συναρτήσει των εποχών εκπαίδευσης, από το callback TensorBoard φαίνεται παρακάτω :



### Παρατήρηση:

Με γαλάζιο σημειώνεται η πρόοδος του validation set ενώ με κόκκινο η πρόοδος του training set

Όπως φαίνεται παραπάνω το δίκτυο πέτυχε ακρίβεια στο σετ δεδομένων εκπαίδευσης **0.8278**, δηλαδή **82.78%** ενώ στο σετ δεδομένων δοκιμής **0.1936** δηλαδή **19,36%**. Σε αυτήν την περίπτωση έχουμε σίγουρα **overfitting**, δηλαδή σε απομνημόνευση των ήδη γνωστών εικόνων από το σετ εκπαίδευσης και όχι γενίκευση τους. Αυτό μπορεί να οφείλεται σε κάποιες τιμές παραμέτρων του δικτύου. Μια λύση προκειμένου να βρεθεί ένα βέλτιστο αποτέλεσμα της εκπαίδευσης, θα ήταν η δοκιμή όλων των συνδυασμών από τιμές για τις παρακάτω παραμέτρους:

- batch\_size
- πλήθος epochs
- learning rate
- αριθμός νευρώνων στα dense επίπεδα
- ποσοστό dropout rate στα επίπεδα dropout